UNIVERSITÉ DE GENÈVE **FACULTÉ DES SCIENCES**

Institut Suisse de Bioinformatique **Professeur R.D. Appel**
Département d'Informatique
Département de Biologie Structurale et Bioinformatique

Institut Suisse de Bioinformatique **Docteur C. Hoogland**

Managing and Publishing Proteomics Data

The Make2D-DB II Tool, an Integrative Environment Applied to 2-DE Datasets

THÈSE

Présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention bioinformatique

par

Khaled Mostaguir

de
Genève (GE)

Thèse N° 4001

GENÈVE
Atelier de reproduction de la section de physique
2008

UNIVERSITÉ
DE GENÈVE

FACULTÉ DES SCIENCES

Doctorat ès sciences
mention bioinformatique

Thèse *de Monsieur Khaled MOSTAGUIR*

intitulée :

" Managing and Publishing Proteomics Data
The Make2D-DB II Tool, an Integrative Environment
Applied to 2-DE Datasets "

La Faculté des sciences, sur le préavis de Monsieur R. APPEL, professeur ordinaire et co-directeur de thèse (Département d'informatique), Madame Ch. HOOGLAND, docteur et co-directrice de thèse (Faculté de médecine – Institut Suisse de Bioinformatique), Madame F. LISACEK, docteur (Faculté de médecine – Institut Suisse de Bioinformatique), Messieurs Ch. PELLEGRINI, professeur ordinaire (Département d'informatique), et M. DUNN, professeur (University College Dublin – Conway Institut of Biomolecular & Biomedical Research – Dublin, Ireland), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 4 juin 2008

Thèse - 4001 -

[illegible]

Le Décanat

N.B. - La thèse doit porter la déclaration précédente et remplir les conditions énumérées dans les "Informations relatives aux thèses de doctorat à l'Université de Genève".

Nombre d'exemplaires à livrer par colis séparé à la Faculté : - 7 -

Abstract

Managing and Publishing Proteomics Data

The Make2D-DB II Tool: an Integrative Environment Applied to 2-DE Datasets

In a living organism, a genome generates the active entities that are responsible for carrying out every aspect of life, the proteins. Study of these entities is a fundamental sub-domain of molecular biology, called proteomics. Proteome analysis depends on separation techniques to reduce the complexity of the protein mixture. Once the proteins are separated, their properties and expressions can be compared. They can also be broken into smaller pieces, using mass spectrometers, to weigh the resulting fractions. One major technique to realise protein separation is through Two-Dimensional Polyacrylamide Gel Electrophoresis (2D-PAGE). In this technique, proteins are separated according to their masses and their charges over a two-dimensional area. Currently, 2D-PAGE is the only technique that can be routinely applied for parallel quantitative expression profiling of large sets of complex protein mixtures. Gels are highly reproducible and the quantification of spots and differential analysis are generally performed using image analysis software. Since analyses and characterisation results cannot be of great interest without being made accessible to the scientific community, 2D-PAGE experiments and analysis must also be reported. 2D-PAGE datasets generally cover experiment parameters, spots properties, identification techniques, identified proteins, specific annotations, and references to related external resources. However, 2-DE datasets are typically limited in their content and are often specific in their format to people that are producing them. When inspecting the many data resources available over the Internet, we rapidly notice the degree of autonomy and diversity of these resources when compared to each other. Besides, the majority of these resources are isolated datasets that are often unstructured or unrefined. Problems are ranging from the diversity of semantics, of data formats to technologies and interfaces used to access the data. Problems of incomplete information, or worse, of incompatibilities and conflicts between data are often encountered. Dealing with so many disparate resources may be confusing. Navigating in many different locations and processing query results from one resource before accessing another resource is extremely tedious. In addition, a significant part of 2D-PAGE data is either not available to the community, or is only published in the literature without being electronically accessible. This underlines the necessity of offering comprehensive solutions to manage and access 2D-PAGE data and to promote data integration between remote datasets.

The aim of the Make2D-DB II tool, which we have developed at the Swiss Institute of Bioinformatics, is to provide a flexible and easy-to-use distributed environment that creates, converts, processes, publishes, maintains and interconnects 2D-PAGE datasets. It aims at the establishment of highly reliable and easy-to-build 2-DE databases. The tool, which is compliant with current standards, converts text reports into an evolving and modular data representation. It is based on top of an extended, realistic and highly consistent data model that efficiently captures all aspects of 2D-PAGE analyses and related data. The environment ensures strong data reliability, automatic integration of various external data, and interoperability between remote 2-DE databases. It is designed to include as much experimental information as possible, in order to improve the quality of the database content. In addition, Make2D-DB II is provided with a rich query interface that is intuitive to use and that can access simultaneously any number of remote 2D-PAGE databases. Dynamic synchronisation between remote databases ensures that distant databases are up-to-date with regard to each others.

Make2D-DB II is a fully functional environment that can be used to build single databases, dynamic portals and public repositories. Since its first public release in 2004, the tool has established itself as a reference in data management, in data integration, and in data publication of gel-based proteomics resources. It has been continuously evolving and new releases have been made available from the renowned ExPASy server. The environment has been adopted by a large number of academic and private organisations, resulting in the expansion of a virtual 2-DE database with data distributed all over the world. Many world-famous institutions are currently managing and publishing their proteomics data using this environment.

Make2D-DB II is an open source project, which requires working constantly on the extension of its conception and functionalities in order to stay in tune with the evolution in proteomics techniques, data resources, and data representation. It is also necessary to stay tuned with the ongoing recommendations from the HUPO Proteomics Standards Initiative, which will guarantee future compatibility with other management systems and data resources.

Acknowledgments

I would like to express my sincere gratitude to all the reputable members of the Supervisory Committee for their support and encouragement to achieve the present work:

Professor **Ron D. Appel**
Professor **Michael Dunn**
Dr. **Christine Hoogland**
Dr. **Frédérique Lisacek**
Professor **Christian Pellegrini**

Ce travail n'aurait jamais vu le jour sans le soutien et l'aide de nombreuses personnes à qui j'aimerais exprimer toute ma gratitude. Tout d'abord ***Ron Appel****, qui m'a donné une chance extraordinaire de travailler dans son groupe et qui m'a permis tout au long de ces nombreuses années de découvrir et d'apprécier ses qualités certaines en tant qu'excellent directeur, mais également, et surtout, en tant que personne profondément humaine. Je tiens également à remercier* ***Frédérique Lisacek*** *pour son remarquable soutien professionnel et moral, pour sa patience et sa passion et pour avoir été toujours présente quand j'avais besoin d'elle. Je tiens également tout particulièrement à remercier* ***Christine Hoogland*** *de m'avoir assisté tout au long de ces années de travail. Pour toute la connaissance qu'elle m'a transmise, ainsi que sa gestion et tous ses précieux conseils sans lesquels ce travail n'aurait jamais pu voir le jour.*

Many thanks to professor ***Michael Dunn*** *and professor* ***Christian Pellegrini*** *for their support and encouragements. I am greatly thankful to them for all their precious time they are giving me by supervising my thesis, and for their advices regarding the present document.*

Tous mes collègues du SIB avec qui j'ai passé de très merveilleuses années, tant sur le plan professionnel que sur le plan de l'amitié. Leur connaissance m'a été profondément enrichissante, et j'en garderai toujours un formidable souvenir… je les remercie tous… ***Amos*** *pour son professionnalisme et sa passion,* ***Ivan****,* ***Karin*** *et* ***Salvo*** *pour leur soutien technique et surtout pour leur amitié,* ***Claudia*** *et* ***Véronique*** *pour leur sympathie très chaleureuse,* ***Laure*** *et* ***Dolnide*** *pour tout ce qu'elles font pour nous,* ***Patricia*** *et* ***Patricia*** *qui sont de formidables collègues,* ***Marc*** *pour son agréable compagnie,* ***David****,* ***Nadine****,* ***Céline*** *et* ***Grégoire*** *pour toute leur précieuse amitié,* ***Pierre-Alain*** *pour ses précieux conseils…*

De nombreuses autres personnes de Swiss-Prot m'ont également été d'une aide très appréciable. Tout particulièrement ***Nicole****,* ***Eric****,* ***Isabelle****,* ***Sandrine****,* ***Alexandre****,* ***Serenella*** *et* ***Edouard****, que je remercie sincèrement. Je remercie également tous les joueurs de tarot du SIB, des joueurs d'échecs et ceux qui ont participé à l'équipe de football du SIB Genève… de très bons moments partagés au sein du groupe.*

Mes remerciements à ***Marie-Laure*** *pour le formidable travail de correction et de suggestion de l'anglais utilisé tout au long de ce document.*

*Finalement, toute ma reconnaissance et mes sentiments à **Maman** et à **Nashwa**, pour leur soutien et leur amour sans fin… et à toi **Elisabeth**, pour tout ce que tu m'as donné et continues à me donner.*

A tous, merci pour tout…

A la mémoire d'un homme si extraordinaire, à Papa…

RÉSUMÉ EN FRANÇAIS

Gérer et publier des données protéomiques

L'outil Make2D-DB II : un environnement intégratif appliqué aux données 2-DE

Dans un organisme vivant, un génome génère des entités actives qui sont à l'origine de tous les aspects de la vie : les protéines. L'étude de ces entités constitue un sous-domaine fondamental de la biologie moléculaire, appelé la protéomique. L'analyse du protéome s'effectue grâce à des techniques de séparation qui permettent de réduire la complexité de la constitution du protéome. Une fois que les protéines sont séparées, on peut alors comparer leurs propriétés et leur expression. Elles peuvent également être cassées en de plus petits fragments qui peuvent être pesés au moyen de spectromètres de masse. L'une des principales techniques de séparation des protéines est l'électrophorèse bidimensionnelle sur gel de polyacrylamide (2D-PAGE ou 2-DE). Dans cette technique, les protéines sont séparées en fonction de leur masse et de leur charge sur une surface bidimensionnelle. En fait, la méthode 2D-PAGE est la seule technique qui peut être appliquée de manière systématique pour l'expression quantitative parallèle du profile des mélanges complexes de protéines. Les gels sont faciles à reproduire et la quantification des spots et l'analyse différentielle sont généralement réalisées grâce à l'utilisation de logiciels d'analyse d'images. Pour présenter un intérêt, les résultats d'analyse et de caractérisation doivent être accessibles à la communauté scientifique et c'est pourquoi les expériences 2D-PAGE doivent également faire l'objet de rapports détaillés. Les données relatives à une expérience 2D-PAGE concernent généralement les paramètres expérimentaux, les propriétés des spots, les techniques d'identification, les protéines identifiées, les annotations spécifiques et des références à des ressources externes liées. Cependant, le contenu de ces données est typiquement limité et leur format est souvent spécifique aux gens qui les produisent. Lorsque l'on étudie les nombreuses sources de données disponibles sur Internet, on s'aperçoit rapidement du degré d'autonomie et de la diversité de ces ressources lorsqu'on les compare les unes aux autres. En outre, la majorité de ces ressources est constituée de données isolées, souvent brutes et sans structure. Les problèmes vont de la diversité de la sémantique, des formats de données, aux technologies et interfaces utilisées pour y accéder. On rencontre souvent des difficultés en raison d'informations incomplètes, ou pire encore, d'incompatibilité ou de conflits entre les données. Avoir à faire face à tant de ressources aussi disparates peut s'avérer compliqué. Naviguer sur de multiples sites et devoir traiter les résultats d'une requête provenant d'une source avant de pouvoir accéder à une nouvelle source est extrêmement fastidieux. De plus, une part importante des données 2D-PAGE est inaccessible à la communauté ou bien est seulement publiée dans un format papier et donc électroniquement inaccessible. Tous ces éléments mettent en évidence la nécessité d'offrir des solutions complètes permettant de gérer et d'accéder aux données 2D-PAGE et de promouvoir l'intégration des données entre les différentes bases de données distantes. Le but de l'outil Make2D-DB II que nous avons développé au sein de l'Institut suisse de bioinformatique consiste à fournir un environnement flexible et facile à utiliser qui crée, convertit, traite, publie, entretient et interconnecte des bases de données 2D-PAGE. Il vise à créer des bases de données fiables et faciles à construire. Cet outil, conforme aux normes de standardisation actuelles, convertit de simples rapports écrits en une représentation de données évolutive et modulaire. Il repose sur un modèle de données étendu, réaliste et hautement cohérent qui saisit de manière efficace tous les aspects des analyses 2D-PAGE et des données liées. L'environnement garantit une haute fiabilité des données, l'intégration automatique de diverses données externes et une interopérabilité entre les bases de données distantes. Il présente une interface de requête d'utilisation intuitive et qui peut accéder simultanément à un nombre illimité de bases de données 2D-PAGE distantes. Une synchronisation dynamique entre les bases de données garantit que les bases de données distantes sont à jour les unes par rapport aux autres.

Made2D-DB II est un environnement totalement fonctionnel. Depuis son premier lancement public en 2004, l'outil s'est positionné comme une référence en matière de gestion de données, d'intégration de données et de publication de ressources protéomiques d'électrophorèse bidimensionnelle. Il a continuellement évolué et il est possible d'accéder aux nouvelles versions à partir du serveur ExPASy. Notre environnement a été adopté par un grand nombre d'organisations académiques et privées qui ont créé et contribué à l'expansion d'une base de donnée 2D-PAGE virtuelle, avec des données distribuées partout dans le monde.

Make2D-DB II est un projet *open source* qui requiert un travail permanent pour l'extension de sa conception et de ses fonctionnalités afin de répondre à l'évolution que connaissent les techniques protéomiques, les ressources et la représentation des données. Il est également nécessaire qu'il demeure conforme aux recommandations de l'HUPO (Proteomics Standards Initiative) qui garantissent sa future comptabilité avec d'autres systèmes de gestion de données et d'autres sources de données.

Organisation de ce document

Ce document décrit les concepts qui sous-tendent Make2D-DB II, son développement et son environnement intégratif. Le chapitre ***A*** est une introduction en matière. Le chapitre ***B*** donne un aperçu des trois sous-catégories fondamentales de la biologie moléculaire que sont : la génomique, la transcriptomique et la protéomiques. Il décrit ensuite la protéomique et les techniques en matière de séparation de protéines, leur caractérisation et leur identification. Ce chapitre est principalement consacré aux méthodes 2D-PAGE et aux analyses liées. Le chapitre ***C*** dépeint la grande diversité des données trouvées dans les plus importantes ressources en matière de classification et de caractérisation des protéines en lien avec notre travail. Les aspects techniques liés à la structure des données et leur gestion sont présentés dans le chapitre ***D*** qui fait également état des principales approches en matières d'intégration de données dans les sciences de la vie. Certains systèmes d'intégration de données connus sont examinés à la fin de ce chapitre. Le chapitre ***E*** explique les raisons qui nous ont conduit à conceptualiser et développer Make2D-DB II. Ce chapitre examine méticuleusement tous les détails du modèle de données qui constitue l'élément central de cet outil. La mise en œuvre de Make2D-DB II et son environnement intégratif sont révélés dans le chapitre ***F***. Ce dernier décrit le fonctionnement physique de l'outil, l'interconnexion entre les systèmes distribués et l'intégration des données. Le fonctionnement des interfaces Web et la manière dont les données sont échangées entre les installations distantes sont les thèmes abordés dans le chapitre ***G***. Le chapitre ***H*** évalue la contribution de Make2D-DB II dans la création d'une base de donnée virtuelle consacrée aux données de l'électrophorèse bidimensionnelle. De nombreuses bases de données construites au moyen de notre outil sont énumérées dans ce chapitre. Les deux derniers chapitres, ***H*** et ***I,*** présentent les principales perspectives d'avenir à court et long termes de l'outil. Le manuscrit se referme en traitant de la place de l'intégration des données dans la biologie moléculaire dans un proche avenir.

Chapters

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Glossary

AJAX: AJAX (Asynchronous JavaScript and XML) is a group of inter-related web development techniques used for creating interactive web applications. A primary characteristic is the increased responsiveness and interactivity of web pages achieved by exchanging small amounts of data with the server. This is intended to increase the web page's interactivity, speed and functionality.

API: An Application Programming Interface is a source code interface that a computer system or program library provides to support requests for services to be made.

BLOB: Binary Large Objects are binary data stored as a single entity in a database management system. BLOBs are typically images, audio or other multimedia objects.

Bottom-up approach: The individual base elements of the system are first specified in detail. These elements are then linked together to form larger subsystems until a complete top-level system is formed. The beginnings are small, but eventually grow in complexity and completeness. However, elements and subsystems are developed in isolation, and are subject to local optimisation as opposed to meeting a global purpose.

CSS: Cascading Style Sheets. CSS is a style language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML. It is designed primarily to enable the separation of document content from document presentation.

CVS: The Concurrent Versions System (CVS) is an open-source version control system that keeps track of all work and all changes in a set of files, typically the implementation of a software project, and allows collaboration. CVS has become popular in the open source software world.

CORBA: The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.

Garbage collection: The automatic detection and freeing of memory or storage areas that are no longer in use.

Gel electrophoresis: Gel electrophoresis is a widely used technique for separating electrically charged molecules. It is a central technique in proteomics to separate and purify proteins, so they can be studied individually. Gel electrophoresis is often followed by staining or blotting procedures followed by various protein identification techniques.

Hierarchy: A classification of relationships in which each item except the top one (known as the root) is a specialised form of the item above it. Each item can have one or more items below it in the hierarchy.

HTTP: Hyper Text Transfer Protocol. The Internet protocol, based on TCP/IP, used to fetch hypertext objects from remote hosts.

Instance: A class represents a set or collection of objects called instances. Each instance must be uniquely identifiable and distinct from all other instances.

IUBMB: International Union of Biochemistry and Molecular Biology.

MVC: The Model-View-Controller (MVC) is a design pattern for the architecture of complex web applications. It is a widely adopted pattern, across many languages and implementation frameworks, whose purpose is to achieve a clean separation between three main components in most web applications: the model (business logic and processing), the view (user interface and data presentation) and the controller (control of the separation between the model and the view). MVC ensures that changes to the user interface do not affect data handling, and that the data can be reorganized without changing the user interface.

Ontology: "An ontology is an explicit specification of some topic. It is a formal and declarative representation, which includes the vocabulary for referring to the terms in that subject area and the logical statements that describe what the terms are, how they are related to each other, and how they can or cannot be related to each other. Ontologies therefore provide a vocabulary for representing and communicating knowledge about some topic and a set of relationships that hold among the terms in that vocabulary".

ORDBMS: Object-Relational Database Management System. This system simply puts an object oriented front end on a relational database (*cf.* RDBMS).

Portal: *cf.* Web portal.

Protein: A large molecule composed of one or more chains of amino acids in a specific order determined by the base sequence of nucleotides in the DNA coding for the protein. Proteins are required for the structure, function, and regulation of the body's cells, tissues, and organs.

Proteomics: “Proteomics aims at quantifying the expression levels of the complete protein complement (the proteome) in a cell at any given time. While proteomics research was initially focussed on two-dimensional gel electrophoresis for protein separation and identification, proteomics now refers to any procedure that characterises the function of large sets of proteins. It is thus often used as a synonym for functional genomics.”

RDBMS: Relational Database Management System. A type of database management system that stores data in the form of related tables.

REST: Representational State Transfer (REST) refers to a collection of network architecture principles that outline how resources are defined and addressed. The term is often used in a looser sense to describe any simple interface that transmits domain specific data over HTTP without an additional messaging layer such as SOAP. An important concept in REST is the existence of resources, each of which can be referred to using a global identifier (a URI). In order to manipulate these resources, components of the network (clients and servers) communicate via a standardized interface (*e.g.*, HTTP) and exchange representations of these resources.

RSS: Really Simple Syndication (RSS) is a family of Web feed formats used to publish frequently updated content. RSS makes it possible for people to keep up with Web sites content in an automated manner that can be transmitted to special programs or filtered displays.

Rule (RDBMS): Rules in a database offer a way to rewrite some specific SQL queries, or to automatically add some additional SQL instructions to an initial one.

Relational model: The relational model for database management is a database model based on predicate logic and set theory.

Semi-structured: The semi-structured model is a database model. In this model, there is no separation between the data and the schema, and the amount of structure used depends on the purpose. It provides a flexible format for data exchange and viewing. While the schema can be easily modified, a semi-structure model may suffer significantly in consistency and constraints' definition.

Sequence (RDBMS): Sequences are special tables used for generating integer sequences. Typically, they are used to create a unique record ID (or key) for each row in a table.

SOAP: The Simple Object Access Protocol (SOAP) is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. SOAP forms the foundation layer of the web services protocol stack providing a basic messaging framework upon which abstract layers can be built.

TCP/IP: Transmission Control Protocol based on IP. This is an Internet protocol that provides for the reliable delivery of streams of data from one host to another.

Timeout: A network parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. Typically, related to HTTP connection termination.

Top-Down approach: An overview of the system is first formulated, specifying but not detailing first-level subsystems. Each subsystem is then refined in more details until the entire specification is reduced to base elements. A top-down model is often specified with the assistance of "black boxes". However, black boxes may fail to elucidate elementary mechanisms or be detailed enough to realistically validate the model.

Trigger (RDBMS): Triggers are activation processes that are associated with some function or procedure and that are "fired" before or after a specific operation is attempted on a row.

Two-D PAGE / 2D-PAGE / 2-DE: Two-dimensional polyacrylamide gel electrophoresis, *cf.* Gel electrophoresis.

URI: A Uniform Resource Identifier is a string of characters used to identify or name a resource. The identifier enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols (usually HTTP for Web access).

URL: A Uniform Resource Locator is a standard for writing a text reference to an arbitrary piece of data in the World Wide Web. **Logical URLs** are descriptive (self-explanatory) locators, as opposed to **physical URLs**, which reflect the architecture of a directory tree.

Vertices: *Sing.* vertex, are the fundamental units or nodes out of which graphs are formed. Vertices are the indivisible objects that are connected by edges or arcs in a graph.

Web portal: A web portal is a site that functions as a point of access to information on the World Wide Web. Portals present information from diverse sources in a unified way, providing a pathway to other content. A Web portal is designed to use distributed applications, different numbers and types of middleware and hardware to provide services from a number of different sources.

Web service: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. Web services are frequently Web APIs that can be accessed over a network, and executed on a remote system hosting the requested services. Commonly, the term refers to clients and servers that communicate using XML messages that follow the SOAP standard. It is assumed that there is also a machine-readable description of the operations supported by the server written in the Web Services Description Language (WSDL).

Wrapper: A wrapper in computer science is a piece of code that allows classes to work together that normally would not due to interfacing incompatibility. The wrapper acts as an interface between its caller and the wrapped code. This may be done, for example, if the wrapped code is in a different programming language.

XML: The Extensible Markup Language (XML) is a general-purpose specification for creating custom markup languages. It is classified as an extensible language since it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Web, and it is used both to encode documents and to serialize data.

COMMONLY USED ABBREVIATIONS

2D-PAGE: Two-Dimensional Polyacrylamide Gel Electrophoresis.

API: Application Programming Interface.

CSS: Cascading Style Sheets.

CVS: Concurrent Versions System.

EMBL: European Molecular Biology Laboratory.

GUI: Graphical User Interface.

HTTP: Hyper Text Transfer Protocol

HUPO: Human Proteome Organisation.

IUBMB: International Union of Biochemistry and Molecular Biology.

LIMS: Laboratory Information Management System.

NCBI: National Center for Biotechnology Information.

ORDBMS: Object-Relational Database Management System.

PSI: Proteomics Standards Initiative.

RDBMS: Relational Database Management System.

REST: Representational State Transfer.

SIB: Swiss Institute of Bioinformatics.

SOAP: Simple Object Access Protocol.

TCP/IP: Transmission Control Protocol based on IP.

URI: Uniform Resource Identifier.

URL: Uniform Resource Locator.

C h a p t e r **A**

CHAPTER A. INTRODUCTION AND MOTIVATION

Managing scientific data nowadays is a crucial task in *proteomics* and, more generally speaking, in all life science fields. Over the last two decades, there has been a real outburst of data, which still goes on in every biomolecular domain. As a result, many scientists have become or relabelled themselves bioinformaticians, rather than simply geneticists, biologists, or computer scientists. Bioinformatics extends over many interrelated areas of biology, computer science and information technology to merge them into one single discipline. This covers almost the entire domain of biology, combined with data organisation, management and storage, data analysis, mathematics, statistics, detection algorithms, modelling and data representation, data mining, linguistics, physics, chemistry, and, of course, every related computer science development required for the design and the implementation of all related tools.

A.I. Managing data in bioinformatics

A suitable definition of bioinformatics, a moderately new discipline, is stated in the following terms:

"*Bioinformatics is the field of science in which biology, computer science, and information technology merge to form a single discipline. The ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be discerned.*"[1]

Important sub-disciplines of bioinformatics are[2]:

- The analysis and interpretation of different types of data (*e.g.*, nucleotide and amino acid sequences, protein structure, protein domains, etc).
- The development of algorithms and statistics to assess relationships among members of large datasets.
- The development and implementation of tools to access and manage the heterogeneous types of information.

The driving force in biological discovery today relies on the transformation of the multitude of heterogeneous, vast, and complex data on hand into useful organised information, with the ultimate goal to converge this organised information into a systematised knowledge. A representative information-driven discovery process is illustrated in Figure A.I-1 (Chung, Wooley 2003). This figures shows how experimentally generated data must be combined with data derived from other resources using computational data analysis models for a better global interpretation. Data organisation and infrastructure, together with advances in experimental methods, should lead to a better understanding of life science domains.

[1] NCBI, "Just the Facts: A Basic Introduction to the Science Underlying NCBI Resources." (November 2002)

[2] http://www.epa.gov/comptox/glossary.html

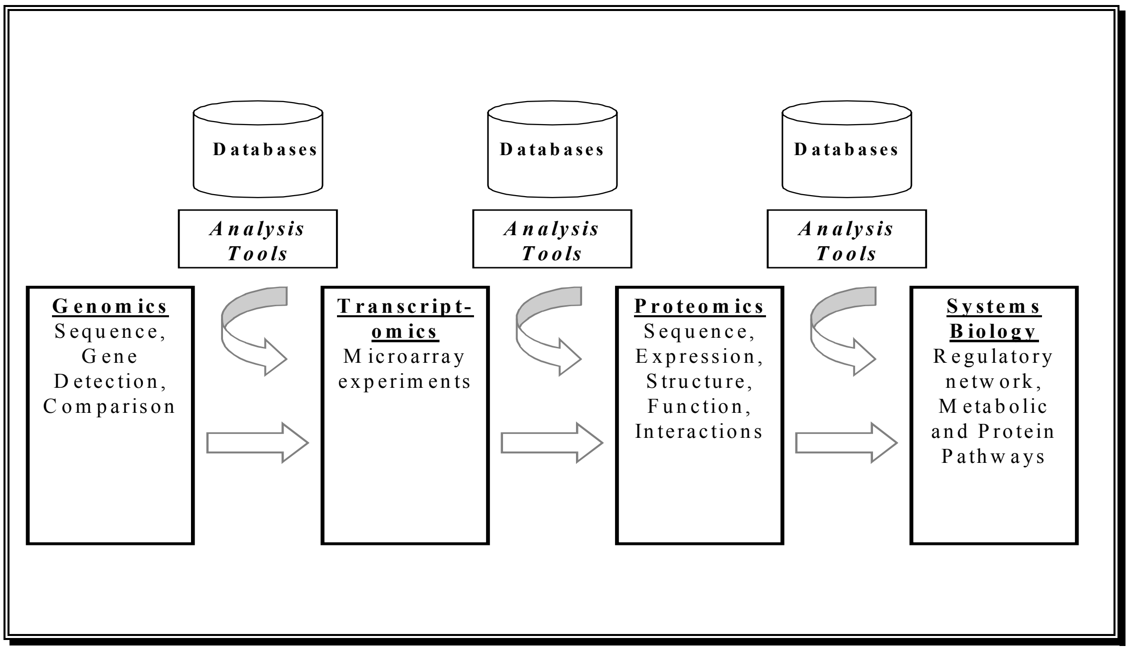

Databases
Databases
Databases
Analysis Tools
Analysis Tools
Analysis Tools
Genomics
Sequence, Gene Detection, Comparison
Transcript-omics
Microarray experiments
Proteomics
Sequence, Expression, Structure, Function, Interactions
Systems Biology
Regulatory network, Metabolic and Protein Pathways

Figure A.I-1: Information-driven discovery (Chung, Wooley 2003).

A.II. Proteomics data and 2-DE datasets

The objective of *genomics* and all related disciplines is to endorse our understanding of the function and evolution of the genomes of living systems. This understanding allows us to formulate problems and to face challenges in life science, medicine and related domains. In a living organism, a genome generates the active entities that are responsible for carrying out every aspect of life, the *proteins*. Study of these entities and their interactions is a fundamental and vast sub-domain of molecular biology, called *proteomics*. As it is the case for many other sciences, experimental data and related analysis are central for the study and elucidation of the considered domain.

Molecular biology data resources

At the present time, hundreds of different molecular biology data resources are publicly available all over the Internet. The content of those databases varies greatly and spreads over the many disciplines and sub-disciplines of life science fields. The *Journal of Nucleic Acid Research* (Galperin 2007) publishes each year a fairly extended collection of biological resources "*that are freely available to the public and may be useful to the molecular biologist*". The 2007 update includes already 968 public data resources. It is significant that this number has tripled since 2002 and that between 2006 and 2007 only, some 110 additional data resources have made their appearance in the list.

In the present document, the term "resources" will be conventionally used to designate databases or datasets (structured data), as well as any collection of less structured data that presents sequential and organised records. Only academic and publicly available data will be highlighted.

Proteome analysis and 2-DE resources

Proteome analysis depends on separation techniques to reduce the complexity of the protein mixture. A proteome generally contains hundreds of thousands of proteins. It is therefore essential to separate and organise those different types of proteins.

One major technique to realise this separation is through **2-Dimensional Polyacrylamide Gel Electrophoresis** (2-DE or 2D-PAGE) (O'Farrel 1975; Gorg et al. 2000). Since proteins differ from each other in terms of mass and charge, it is possible to separate them according to both properties over a two-dimensional area, a gel, using gel electrophoresis techniques. Data regarding the 2-DE separation procedure and the protein identification methods and results is typically reported in disparate resources. 2-DE resources are usually limited in their contents and are often specific in their formats to people and laboratories that are producing them.

One of our main concerns was the amount of 2-DE resources that are generated by numerous laboratories without being made available to the proteomics community. A large number of isolated 2-DE datasets is produced in many different formats, ranging from highly structured formats to plain text reports, and with dissimilar levels of annotations. Besides, a significant part of this data is either not available to the community, or is only published in the literature without being electronically accessible.

Even when access to these resources is possible, researchers may still be confused. Navigating in many different locations and processing query results from one resource before accessing another resource is a tedious task. This underscores the need to propose comprehensive and intelligent integration solutions to uniformly access such types of resources.

A.III. What is the Make2D-DB II package

Managing and publishing 2-DE resources exhaustively and efficiently was the challenge we tried to take up when we started to conceive our project. We wanted to provide researchers with the necessary tools to manage and publish their data. The fact that the same management system would be shared between remote databases directed our work towards the conception of a large-scale environment: a federated environment in which 2-DE resources are distributed while still being able to interact and exchange data.

From the "Make2D-DB II" tool site[1], an environment that was initially announced in 2003 (Mostaguir et al. 2003), a short and non-technical description is put forward to the tool users:

"*Make2D-DB II is…* [an environment] *to create, convert, publish, interconnect and keep up-to-date 2-DE databases. Being open source, it is distributed (...) free of charge.*

With this tool, one can easily convert any existing personal federated 2-DE database (including databases built with the first Make2ddb package or following the SWISS-2DPAGE conventions) into a more reliable format. The tool also handles XML exports from Melanie / ImageMaster^{TM} 2D Platinum, common spreadsheets (e.g., Excel / CSV reports) as well as simple text lists. It is also possible to create new relational databases from scratch. It runs on most UNIX-based operating systems (Linux, Solaris/SunOS, IRIX). Being continuously developed, the tool is evolving in concert with the current Proteomics Standards Initiative of the Human Proteome Organisation (HUPO).

Make2D-DB II is designed to ensure high consistency of data. It allows dynamic interconnection between any number of similar remote databases and offers many other features, including automatic data updates related to external data, dynamic cross-references to similar databases, intuitive search engine and data visualisation combined with exports in various formats.

In addition, the data model extends the concept of 2-DE databases to cover a more elaborate description of proteomic experiments and analysis. Currently, it can integrate various annotation documents (e.g., sample and gel preparation), as well as many common analysis file formats (e.g., common mass spectrometry formats, mzData, etc.). References to other remote repositories can also be easily integrated (e.g., PRIDE for MS/MS related data). Users can also define their own types of annotations and easily integrate them within their data representation.

Even with no local data, any laboratory can easily build up an intuitive Web portal accessing as many remote 2-DE resources as desired.

The Web interface appearance can be personalised for everyone's taste. Data can be marked to be public, as well as fully or partially private. An administration Web interface, highly secured, makes external data integration, data export, data privacy control, database publication and version control a very easy task to perform."

A.IV. Achievements

In the last few years, Make2D-DB II has established itself as a reference in data management, in data integration, and in data publication of gel-based proteomics resources. The tool has been adopted by a large number of academic and private

[1] http://world-2dpage.expasy.org/make2ddb/

organisations, resulting in the expansion of a virtual 2-DE database with data distributed all over the world.

An important constituent of our project is the recent World-2DPAGE Constellation. A large virtual resource including a portal to access many distributed data resources at once, as well as a public standards-compliant repository aiming to host gel-based proteomics data and to support laboratories that do not have the means of publishing and giving access to their data on the Web.

Like many other integration and management systems, Make2D-DB II is a contribution that aims to provide for a better understanding of the complexity of life science domains.

A.V. Organisation of this document

This document describes the concepts and the development of Make2D-DB II and its integrative environment. Chapter ***B*** gives an introductive overview of the three fundamental subcategories of molecular biology: genomics, transcriptomics and proteomics. It then describes proteomics and related techniques in protein separation, characterisation and identification. The chapter focuses principally on 2D-PAGE methods and related analyses. Chapter ***C*** depicts the variety of data found in the most important resources in protein classification and characterisation with regard to our work. Technical aspects related to data structure and management are presented in chapter ***D***, along with the principal data integration approaches in life science. Some known data integration systems are examined at the end of this chapter. Chapter ***E*** explains the motivations that drove us to conceptualise and develop Make2D-DB II. This chapter meticulously inspects all the details concerning the data model, which is the central element of the tool. The implementation of Make2D-DB II and its integrative environment are revealed in chapter ***F***. This chapter is a reference on how the tool is physically working, how the distributed systems interconnect, and how data is integrated. The functioning of the Web interfaces and the way data is exchanged between the remote installations are the topics of chapter ***G***. Chapter ***H*** evaluates the contribution of Make2D-DB II in establishing a virtual worldwide gel-based database. Many datasets and resources that have been constructed using our tool are listed in this chapter. The last two chapters, ***H*** and ***I***, present the most important short and long-terms perspectives regarding the future of the tool. The place of data integration in molecular biology in the near future is the final point discussed at the end of this manuscript.

Alongside the principal document, a supporting set of appendices have been supplied as a complementary material and reference covering a variety of subjects in relation to our work.

Chapter B

CHAPTER B. PROTEOMICS: DEFINITION AND TECHNIQUES

By analogy with the term *genomics*, the study of the genes, the term *proteomics* has been proposed to define the study of *proteins*. Proteomics study is a fundamental and vast sub-domain of molecular biology. Its primary concern is the exploration and the characterisation of proteins, of their structure, functions and interactions. Studying proteomics is much more complex than genomics, mostly because a *proteome* differs from cell to cell and changes through its interactions with the genome and the environment.

Researchers in the life science fields are assisted in their work by numerous molecular biology data resources. To be able to benefit from all these data resources in protein investigations, one needs to have a good understanding of proteomics. But a good understanding of proteomics and all related techniques and tools implies a good master of all the basic components of the domain.

In this chapter, we will first give an overview of *genomics*, *transcriptomics*, *proteomics*, and the "*omics*" evolving sciences. We will then describe in detail proteomics and the different protein characterisation and identification techniques. A supporting description of the fundamental elements in molecular biology, the genetic material, can be consulted from Appendix I.

B.I. Genomics, transcriptomics and proteomics

B.I.1 Genomics

Genomics is the discipline that studies the DNA sequences in the chromosomes of organisms. One important objective is to detect the genes within the different genomes. The very first genome to be entirely sequenced was realised in 1977 by Frederick Sanger for the bacteriophage phi-X174 (Sanger et al. 1977). In 1995, Haemophilus influenzae became the first free-living organism to be sequenced. Since then, many genome projects for several species have been initiated all over the world and at a rapid pace. A rough draft of the human genome was completed by the "Human Genome Project" in 2001. Current estimates place the human genome at around 3 billion base pairs and about 25 thousand genes[1].

There are several sequence databases of different scope and organisation. The International Nucleotide Sequence Database Collaboration[2] (INSDC) is a unified comprehensive database that contains publicly available raw DNA sequences with some basic annotations for approximately 165 000 organisms, obtained mostly through submissions from individual laboratories and from large-scale sequencing projects. It is the outcome of an association between the three major independent DNA sequence databases: GenBank at NCBI, the National Center for Biotechnology Information (Benson et al. 2006); EMBL Nucleotide Sequence Database at the European Molecular Biology Laboratory (Kulikova et al. 2007; Cochrane et al. 2006); and DDBJ, the DNA Data Bank of Japan (Okubo et al. 2006). The EnsEMBL[3] project (Hubbard et al. 2007) offers a broad and integrated source of annotation of chordate genome sequences (33 available genomes in 2007), while Entrez Gene[4] is a representation of gene-specific curated and automatically annotated information available from NCBI (Maglott et al. 2005; Geer, Sayers 2003). Some other important species-specific annotated resources include GDB for Human[5], AceDB, originally for C.elegans[6], FlyBase for drosophilia (The FlyBase Consortium 2003), MGD for mouse (Eppig et al. 2005), MaizeDb[7] and EcoGene for E.coli[8]. Many of these data resources were built by means of specifically adapted data management solutions.

---

[1] Estimates from the "International Human Genome Sequencing Consortium"

[2] http://www.insdc.org/page.php?page=home

[3] http://www.ensembl.org/

[4] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene

[5] http://www.gdb.org/

[6] http://www.acedb.org/

[7] http://www.maizegdb.org/

[8] http://ecogene.org/index.php

In addition to DNA sequences, large collections of different resources are available and cover many specific sub-domains of genomics (gene structure, introns and exons, splice sites, transcriptional regulator sites and factors, etc.). Genome sequence records may share some common standard formats for data representation, but conflicts still exist on adopted terms, especially for gene names and nomenclatures, which may lead sometimes to great confusion. In all these collections of genes and DNA sequences, the concept of unique stable identifiers (USI) is essential. In this USI concept, a specific "object", *e.g.*, a definite coding sequence, has one stable and unique identifier, at least within one database. The same concept extends widely to almost all datasets in life science. Use of Life Science Identifiers (LSIDs), a mechanism for retrieving data and metadata across different life science databases, is therefore a regular concept. Such identifiers are commonly called **accession numbers**.

B.I.2 Transcriptomics

When a gene is active, its coding sequence undergoes the transcription process, producing an RNA copy of the gene's information. The *transcriptome* is the set of RNA transcripts (*e.g.*, messenger mRNA) produced by the genome at one time in a given organism or in a particular cell type. It can broadly vary depending on external environmental conditions. The study of the transcriptome is termed **transcriptomics** and is widely used in cellular differentiation studies. Beside a large collection of mRNA databases, many other resources are also specific to non-coding RNA sequences (tRNA, rRNA) and can be listed from the NAR databases Web site.

The technique used to detect in a sample RNA, that may or may not be translated into active proteins, is called *expression analysis*. Expressed genes are frequently examined using techniques based on EST sequencing (Expressed Sequence Tag), SAGE (Serial Analysis of Gene Expression) (Velculescu et al. 1995) or on cDNA microarray technology (Schena et al. 1998). An example of a distributed access database to store raw and normalised data from microarray experiments is the "Stanford Microarray Database"[1] (Ball et al. 2005). ArrayExpress[2] at the European Bioinformatics Institute (EBI) stores submissions from users with details covering experimental protocols and sample preparation in a standard format, while RAD (RNA Abundance Database) is another public gene expression resource at the university of Pennsylvania, which includes data from various techniques, like SAGE, in addition to microarray data[3]. At the NCBI, the Gene Expression Omnibus (GEO) acts as a large public repository for a wide range of data and is integrated in Entrez, the integrative "Life Science Search Engine".

The microarray community has been a pioneer in the life science fields to set up concrete standards for its data representation. The Microarray Gene Expression Data (MGED) society is the international organisation established in 1999 to facilitate sharing of functional genomics, and to focus on proteomics array data (Ball et al. 2002; Ball, Brazma 2006). The standardisation efforts promoted by MGED are well adopted

[1] http://genome-www.stanford.edu/microarray/

[2] http://www.ebi.ac.uk/arrayexpress/

[3] http://www.cbil.upenn.edu/RAD/

by researchers to exchange and re-analyse gene expression microarray data. There are three main components proposed by MGED:

- MIAME, the "Minimum Information About a Microarray Experiment", a document that outlines the minimum information to be reported to unambiguously interpret and reproduce a microarray experiment
- MAGE, composed by "The Microarray Gene Expression Object Model" (MAGE-OM), an XML-based document exchange format (MAGE-ML) and MAGEstk (supporting toolkits)
- MGED Ontology (MO), a set of terms and annotation rules to ensure no loss of meaning among the community[1]

Such efforts have led the way for other communities to establish their own standards for data exchange. This is currently true with the ongoing efforts undertaken by the proteomics community.

B.I.3 Proteomics

Knowledge of the genomic sequence is only the first step towards prediction of the behaviour of gene products (Wojcik, Schachter 2000). **Proteomics** is defined as the post-genomic discipline through which biologists identify and quantify the proteins and characterise their functions, structures, and interactions. It should be pointed out that protein inventory is only one of the major goals of proteomics analysis. More should be learnt about principles of protein-protein interactions, regulation of their concerted functioning, and post-translational modifications. The term *proteome* itself has been introduced in the mid nineties to designate the collection of proteins produced by an organism, a tissue or a cell type (Wilkins et al. 1996). The proteome is much more variable than the genome because of the interactions proteins may have with the genome and with each other, as well as the secondary modifications they undergo. Besides, a proteome is time-dependent as it differs strongly depending on its location, on the conditions and on the stage of life cycle (Englbrecht, Facius 2005).

Proteomics studies have many objectives, among which the quantification of protein expression, the comparison between normal and disease protein patterns, the detection of diagnostic markers, the design of antibodies from antigens, the discovery of drugs and toxicology markers, etc. Proteomics involves a wide range of studies related to protein sampling, separation and characterisation (sections B.II and B.III). In simple terms, proteomics studies reveal a set of acting proteins that have to be characterised. Supporting information on subcellular location, tissue specificity, functional structure, but also supporting studies on phylogenetics, data mining and statistical analysis, are all of assistance for this characterisation. Time is needed to isolate, analyse, quantify, and identify the precise form of the protein, its functions and interactions with other entities. Many proteins are abundant and may mask other proteins in lower concentrations, while the sensitivity of current methods does not allow easy detection of proteins below a certain concentration threshold. Besides, some of the techniques may even be inadequate for some types of proteins, or may not be adapted in certain chemical

[1] http://mged.sourceforge.net/ontologies/index.php

conditions at all. As a result, combining different techniques is often necessary to achieve satisfactory performances. As far as proteomics is concerned, and maybe more than in any other field, the way any previous experimental or deduced data is reported, organised and made accessible is highly determinant. Proteomics data resources are highly heterogeneous, ranging from sequence and *knowledge bases*[1], from theoretical, calculated or extrapolated data, to databases and repositories of experimental separation and identification results. Because of the importance these resources represent in all attempts to interpret any proteomics study, and in relation to our work, we will focus on the more representative of them later in next chapter (section C.II).

The Human Proteome Organisation

The Human Proteome Organisation[2] (HUPO) is a major player in today's proteomics research. A worldwide consortium was launched in 2001 which mainly aimed at making an inventory of all human proteins, at creating a molecular protein atlas of cells, organs, tissues, schemas of protein-protein interactions, at developing special informational databases, and searching for specific markers of pathological processes (Hanash, Celis 2002). The organisation has already launched several main projects based on international collaboration and is active in facilitating the sharing and the exchange of proteomics data ([No authors listed] 2005). Discovery of potential diagnostic markers and therapeutic targets, one of the main goals of HUPO, will heavily rely on proteomics expertise techniques, such as 2-D gel electrophoresis separation and mass spectrometry identification.

B.I.4 "Omics"

Like in gen*omics*, transcript*omics* and prote*omics*, the "*-omics*" suffix is commonly used in many other areas of molecular biology. However, some of these areas are not entirely distinct and may overlap, while others are unclear. While "pseudo-*omics*" tend to vanish, the well established other *omics* will hopefully cooperate and complete each other.

The term **interactomics** is now adopted to describe protein-protein interactions and networks studies, while the term **metabolomics** has been introduced to represent the collection of all metabolites involved in a biological system, such as metabolic intermediates, reactants, hormones and other signalling molecules (Lindon et al. 2005). This covers the products of gene expression as a whole. "*Metabolomics is the systematic study of the unique chemical fingerprints that specific cellular processes leave behind*" (Daviss 2005). Metabolic profiling, using mass spectrometry and NMR techniques, intends to give an instantaneous snapshot of the physiology of the cell.

Systems biology emerges from the integration of proteomics, transcriptomics, metabolomics, and other "*omics*" to move one step further and materialise a more complete picture of living organisms (Lisacek et al. 2006b). Integrative and "cross-*omics*" approaches are increasingly incorporated in the analysis workflows (Perco et al.

[1] A term rather used among life science communities to designate databases of human expertise.

[2] http://www.hupo.org/

2006). This can be seen as "*an attempt to structure knowledge into hierarchical levels: from gene products to whole organisms*" (Lisacek, Appel 2007).

B.II. Protein characterisation in proteomics

"*To really understand biological processes, we need to understand how proteins function in and around cells since they are the functioning units*"[1].

How can proteins mixed in various abundances in a proteome be characterised? One way is by separating them. Many separation methods are available, among which chromatography-based and electrophoresis-based techniques. When necessary, multiple separation procedures can be applied successively in what is referred to as multi-dimensional separation. Once the proteins are separated, their physico-chemical properties and their amino acid compositions can be compared. They can also be broken into smaller pieces, using mass spectrometers, to weigh the resulting fractions. Each type of amino acid has a unique mass, making identification "relatively" straightforward. By identifying the smaller pieces or by deducing parts of their amino acid sequence, the different proteins can be identified.

In this section, and as an illustration, we are going to describe one of the many possible paths combining different techniques to achieve protein identification.

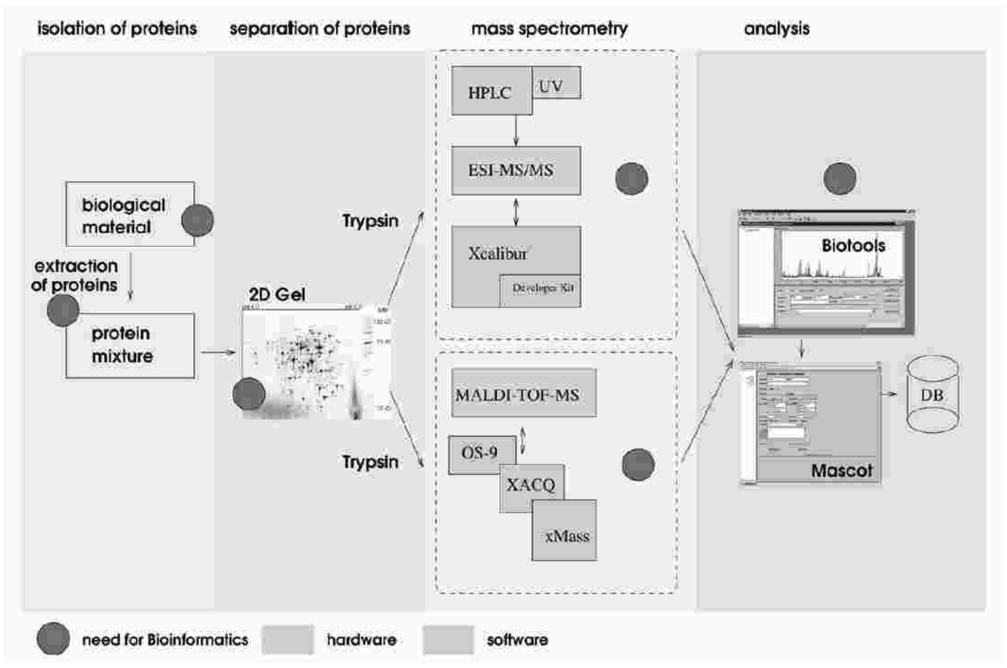


Figure B.II-1: Protein characterisation - example of a workflow (Wilke et al. 2003).

[1] Hanno Steen, director of the Proteomics Center at Children's Hospital Boston

B.II.1 Sample preparation and isolation of proteins

The first step in identifying proteins that are contained in a cellular extract or in a set of cells (*e.g.*, a biopsy or body fluid) consists in generating a large quantity of proteins to work with. The cells are left to grow in a container using a growth medium, a source of nutrition for the cells allowing the cells to feed and multiply. The cell culture is divided and allowed to multiply again, as many times as necessary, in order to obtain hundreds of millions of copies of the original cell. The culture is then placed in a buffer solution and a detergent is added to dissolve the outer membrane of the cells. This results in a solution of proteins mixed with cell remains. Centrifugation is then applied to definitely isolate those proteins from the cell debris.

Sampling of biological material is primarily an intrinsic part of technical laboratory skills. It significantly influences the quality of proteomics studies, and thus should be addressed with care (Govorun, Archakov 2002). Examples of sample preparation and solubilisation methods, especially intended for 2-D electrophoresis (see next sub-section), can be found at *http://www.expasy.org/ch2d/protocols*, as well as in many of the next following listed references.

B.II.2 Protein separation using 2-D electrophoresis

Proteome analysis depends on separation techniques to reduce the complexity of the protein mixture. The proteins produced by the cells, generally hundred of thousands of different types, are to be sorted out. The next step is to separate and organise those different types of proteins.

One major technique to realise this separation is through **2-Dimensional Polyacrylamide Gel Electrophoresis** (2-DE or 2D-PAGE). This technique being at the core of our work, it will be the focus of a dedicated section (B.III). At this point, we only need to know that the separation is performed over a special gel strip where the proteins, through the application of a voltage, are separated along a line (the first dimension), and then separated further over an area (the second dimension). The separation performed over the first axis is based on the disparity of **pI** (isoelectric point) between the different proteins. Differences between their molecular weight **Mw** (which is more or less proportional to their size) make the separation along the second axis feasible (Figure B.II-2). A 2-DE-based approach however presents one major drawback, as some subsets of proteins are not amenable to 2-DE, such as membrane and basic proteins.

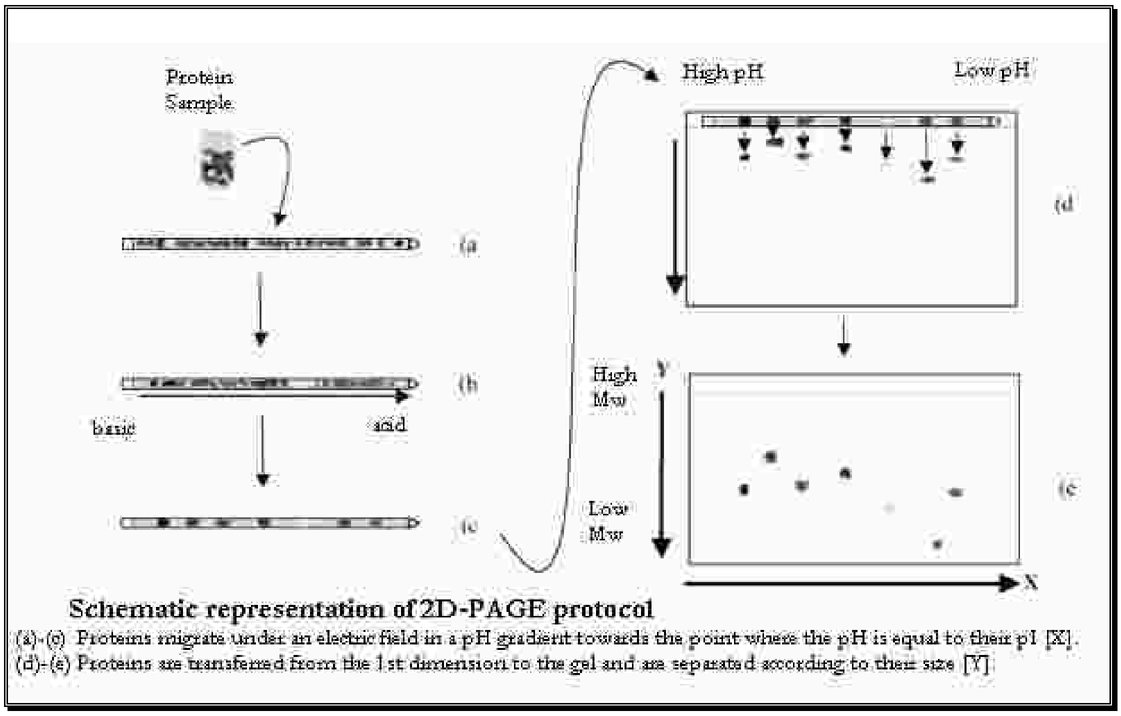

Protein Sample
(a
(b
basic
acid
(c
High pH
Low pH
(d
High Mw
Low Mw
Y
X
(e
Schematic representation of 2D-PAGE protocol
(a)-(c) Proteins migrate under an electric field in a pH gradient towards the point where the pH is equal to their pI [X].
(d)-(e) Proteins are transferred from the 1st dimension to the gel and are separated according to their size [Y]

Figure B.II-2: Schematic representation of 2D-PAGE protocol.

Once distributed over the gel, forming what we commonly call spots, the proteins are visually detected, immobilised or "blotted", and their location is recorded. The final step of the 2-DE separation process usually consists in cutting out or isolating the detected spots for further investigations. The material contained within the spots is then generally transferred into plates or test tubes. This may be done either manually or automatically using a computer-controlled robotic arm.

B.II.3 Protein cleavage and ionisation for mass spectrometry

Mass spectrometry (**MS**) has been playing a major role in proteomics for a long time. For the time being, this technique is widely employed in protein identification and analysis. The technique is not only useful for protein identification in high-throughput workflows. It also offers the necessary refinement to bring out more subtle characteristics, like post-transitional modifications or differential comparison using isotope labels. More details will be given in the next section.

Having performed the separation process over a protein sample, and in order to facilitate their identification, the separated proteins are often cleaved (cut) into smaller pieces. Enzymes, such as proteases, digest (fractioning) the amino acid chain into smaller pieces, ranging only from 6 to 30 amino acids on average. Those generated pieces, called **peptides**, are much easier to handle with the different analysis methods, especially by mass spectrometry.

Several techniques help feeding the cleaved peptides into the mass spectrometer. **Liquid chromatography** (LC) is one of the major techniques used to separate mixtures in biochemistry and in analytical chemistry. HPLC, more specifically high performance liquid chromatography, is a variant of classical LC that forces the mixture to be separated through the column under very high pressure. LC ensures a steady but fast stream of the different peptides, which are then uniformly separated and spread out from a column. The method involves passing the peptide mixture dissolved in a *mobile phase* (solvent) through a *stationary phase* located within the column. The stationary phase is composed of small particles, beads, with some binding properties for proteins. The peptides interact with those particles based on their charge, relative solubility or *adsorption*, which then causes the peptides to be *retained* dissimilarly. Varying chemical conditions in the solution (*e.g.*, with a pH gradient) influences the retention time of the different types of proteins, causing them to elute separately down the column. HPLC has the advantage of reducing considerably the time of retention in comparison to classic LC, thus making the technique appropriate when used in combination with MS analysis. For more details on chromatography, a helpful online tutorial is available from "Library 4 Science" at *http://www.chromatography-online.org/*.

At the end of the column, the peptides and the solvent reach a cone shape. When they emerge from the tip, an applied strong electric field ionises them and disperses them into an aerosol of highly charged droplets. The solvent evaporates, and the peptides, left to retain the positive charge, accelerate towards the negatively charged opening of the mass spectrometer. This is called the Electrospray Ionisation, **ESI** (Nilsson, Davidsson 2000). Another important technique of ionisation is the Matrix-Assisted Laser Desorption/Ionisation, **MALDI** (Karas, Hillenkamp 1988; Zaluzec et al. 1995), sometimes combined with pre LC separation technique experiments, though not in a continuous and direct flow like with ESI. The following section will deal with the MS techniques in more detail.

Let us note that many other variants of protein cleavage and ionisation, implying or not LC pre-separation, are quite common. We can point out for example the molecular scanner, which is an automated process. All proteins of a 2-DE gel are first simultaneously digested proteolytically within the gel and electro-transferred for extraction onto an appropriate membrane (PVDF). The membrane is then directly scanned by mass spectrometry (Binz et al. 1999).

B.II.4 Mass spectrometry

The purpose of mass spectrometry is to measure the mass-to-charge ratio of gas-phase ions (m/Z), where *m* represents the molecular mass of the ion, and *Z* its effective charge. The spectrometer generates a mass spectrum representing the masses of fragmented elements. In proteomics, the spectrometer measures the masses of the individual peptides and their fragments. This information is used to find out the identity of the parent peptides, thus identifying the original proteins (Aebersold, Mann 2003). Developments of technology and methodology in the field of mass spectrometry have been rapid over the last few years, providing improved and novel strategies for high-throughput analysis of proteins (Guerrera, Kleiner 2005).

Two major mass spectrometry approaches are used in protein identification. The **Peptide Mass Fingerprinting** (PMF), where proteins are cleaved into smaller peptides, the masses of which are measured, and the **tandem mass spectrometry**

(MS/MS), involving multiple steps of mass selection and analysis. Both techniques are described in Appendix II. More details are given in Ashcroft A. tutorial on mass spectrometry[1].

Many projects aiming to provide expandable repositories for MS-derived proteome information have been initiated. These data storage resources open many opportunities, not only to ensure data verification, but also to allow data comparison across different experiments and platforms. The most prominent of these projects are PeptideAtlas (Desiere et al. 2006), PRIDE, the "PRoteomics IDEntifications database" (Jones et al. 2006b), as well as OPD (Prince et al. 2004). Each of these projects has slightly different scopes and goals. Besides, we can also mention other ongoing developments of various management systems to store and analyse mass spectra, *e.g.*, ProDB (Wilke et al. 2003); or to search and exploit data obtained from publicly available data analysis servers, *e.g.*, GPM, the "Global Proteome Machine" (Craig et al. 2004).

B.II.5 Other separation and identification techniques

Multiple separation procedures, based on phyisco-chemical properties of proteins and peptides, can be successively applied on a sample to achieve better efficiency. Depending on the purpose of the investigation and the available resources, a laboratory may opt for one strategy or another. Different approaches are now routinely applied in many laboratories. Alternative multi-dimensional gel-free methods are sometimes preferred to overcome some issues on analytical sensitivity. It is important to note that it is not habitually rewarding to study a whole project based on a single biological sample and identification strategy (Lambert et al. 2005). Statistical methodologies are solicited to complement the interpretations derived from the various experiments.

Multi-dimensional separation by liquid chromatography

We have already introduced chromatographic methods used in combination with ESI-MS analysis. **Multi-dimensional liquid chromatography** (Multi-LC), with the so-called gel-free methods (with no gel electrophoresis separation), is another separation alternative that is easily automated, by opposition to 2-DE which is a kind of "art" rather than a routine technique. Multi-LC works well for some hydrophobic, acid, basic, very small, very large, and low abundance proteins that may be difficult to analyse using more traditional separation techniques. Multiple columns, with different stationary phases, are coupled orthogonally (Figure B.II-3). Fractions collected from one column are selectively transferred to another column for further separation. At the same time, new techniques based on isotope labelling of peptides handle well quantitative comparison of control and experimental samples using MS, without requiring 2-D electrophoresis. Isotope coded affinity tags (ICAT) is one of those methods (Gygi et al. 1999).

[1] http://www.astbury.leeds.ac.uk/facil/MStut/mstutorial.htm

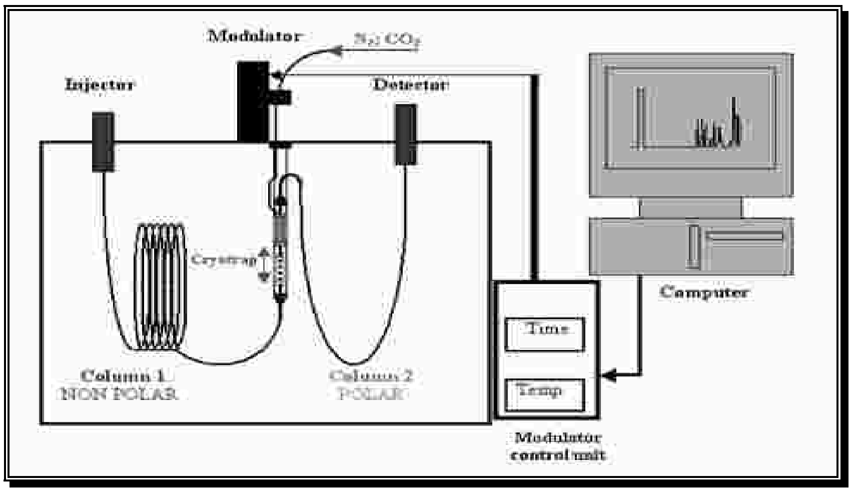


Figure B.II-3: An illustration of a two-dimensional LC installation.[1]

Gel matching in assigning proteins

Under similar experimental conditions, the position of proteins on a 2-D gel and the shape of their spots may be compared to similar characterised gels - often designed as reference maps - thus allowing the determination of the proteins according to their spot location. More details on **gel matching** will be discussed in section B.III.

Other identification techniques

In addition to the widespread use of mass spectrometry in protein identification, many other traditional identification methods are still in use. Although some of them present the inconvenience of being rather slow, labour intensive or relatively expensive, some identification strategies may still require their application, depending on available resources and the non-requirement of a strict high-throughput approach. These methods include, among many others, **microsequencing, immunobloting**, **comigration**, and **amino acid composition** (Wilkins, Gooley 1997). Generally, the identification of proteins involves the characterisation of some attributes (*e.g.*, pI, apparent mass, amino acid composition, sequence tags, etc.) that can be matched against protein databases in various manners. Those attributes may be used individually or in combination to identify any particular protein.

Protein chips, or protein microarray (MacBeath, Schreiber 2000), is a relatively recent technique similar to DNA microarray. A common application of the technique is to study protein-protein interactions. The most common protein microarrays are the antibody microarrays. Thousands of different specific proteins are fixed individually on a chip. A solution of the protein whose partners are looked for inundates the chip. A fluorescent tag, like with DNA microarray, reveals all proteins in the chip that are potential binding partners.

[1] Copyright RMIT Applied Sciences.

3D Structure studies may be the most appropriate to discern information about any protein's function, as they represent the protein as closely as possible to its real active configuration and spatial conformation. Structures are based on experimental X-ray crystallography, NMR, or cryoelectron microscopy data. Unfortunately, technical limitations due to protein crystallisation and other crystallography procedures restrain the number of studied proteins to only a few dozens of thousands. Protein Data Bank[1] (or PDB), the most notable 3D structure collection currently available, indexes about 38'000 known structures that have been deposited until 2006. These defined structures are however useful to predict other proteins' structures by similarity alignment.

As already pointed out, and given the wide range of study strategies, researchers rely more and more in combining results from different techniques, which also include non-proteomics experiments. This enables a better understanding and a more confidence with protein identification and functional assignment.

B.III. 2-Dimensional polyacrylamide gel electrophoresis (2D-PAGE)

B.III.1 Separation techniques

"(Although there are) *many alternative and complementary technologies (e.g., multidimensional protein identification technology, stable isotope labelling, protein or antibody arrays) that have emerged recently, 2-DE is currently the only technique that can be routinely applied for parallel quantitative expression profiling of large sets of complex protein mixtures*" (Gorg et al. 2004).

Since proteins differ from each other in terms of mass and charge, it is possible to separate them according to both properties over a two-dimensional area. Using gel electrophoresis in two perpendicular directions provides a maximum separation of the mixture. Once separated, the proteins may be "visually" revealed using staining agents. This results in a "map" in which **spots** (stains) can be individually quantified, isolated, and analysed by identification techniques.

[1] http://www.rcsb.org/pdb/home/home.do

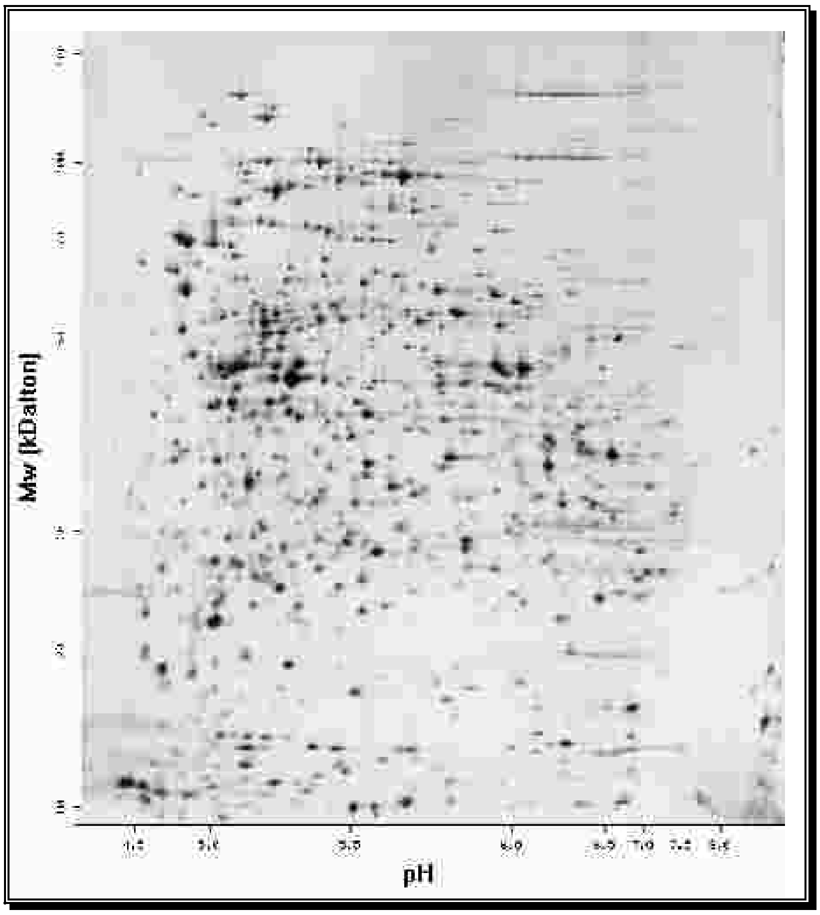


Figure B.III-1: A caption of a stained 2-DE gel where the proteins are distinctively separated in the two dimensions.

Currently, no other technique can match 2D-PAGE in terms of resolution, and sensitivity, in the analyses and comparison of large mixtures of proteins. It is highly reproducible, which allows the comparison of experiments between laboratories. On the one hand, it is used in combination with identification methods, *e.g.*, mass spectrometry, to systematically identify proteins present in a sample. On the other hand, the technique is particularly powerful to compare related samples, such as healthy versus pathological tissues. Comparative 2D-PAGE can also be used to look for proteins whose expression varies similarly or oppositely by changing environmental conditions (and thus may have related functions). All these aspects make gel electrophoresis the most popular separation method in proteomics. A PubMed search in March 2007 with the keywords “two dimensional polyacrylamide gel electrophoresis” revealed more than 17’000 papers.

Special pre-treatment of samples and methods of enrichment of rare polypeptides before 2D-PAGE application may be needed (Herbert et al. 1997). The separation of the treated sample is then performed over a special gel where the proteins are separated according to two of their properties:

- First dimension: according to their isoelectric point **pI**.

- Second dimension: according to their size / molecular weight **Mw**.

There are a large number of proteins in a cell. Some of them may differ in abundance by six orders of magnitude. 2D-PAGE is not sensitive enough to detect the rare proteins and many proteins will not be resolved. Splitting the sample into different fractions is often necessary to reduce the complexity of protein mixtures prior to 2D-PAGE application.

A gel strip of pH gradient ranging from acidic to alkaline (immobilised pH gradient **IPG**), coupled with an applied voltage, makes the proteins migrate over the first dimension: this is called isoelectric focusing or **IEF** (Gianazza et al. 1983). Proteins carry a negative, positive, or zero net charge depending on their amino acid composition and covalent modification (such as phosphorylation, nitrosylation, sulphation and glycosylation), and on the pH of their environment. According to its isoelectric point - the pH at which a protein carries no net electric charge - the protein advances until an electric balance within the gradient is reached (the position at which its charge is the same as the surrounding pH). It is important for the proteins to be well solubilised and reduced before running this procedure; to break up any interaction between them.

Proteins can then be separated furthermore according to their **electrophoretic mobility** (which is proportional to their size/length, their mass or molecular weight, their higher order protein folding, and other factors). A sodium dodecyl sulfate (SDS) solution is added to give the proteins a negative charge proportionally to their size. The solution works also as an agent to denature (unfold) their secondary and tertiary structure. The strip is transferred to a sheet of gel within a tray. A perpendicular voltage is applied this time to make the negatively charged proteins migrate downward in direction of the positive side of the gel located at the other end of the gel area. Proteins make their way through the gel depending on their size. The smaller a protein is, the faster it migrates throughout the gel. While IEF shotgun separation without the second dimension is widespread, 1-D gel electrophoresis techniques (SDS-PAGE) only based on the electrophoretic mobility of proteins are also commonly used (Figure B.III-2).

At the end, proteins are spread all over the gel according to both their pI and their size, in what is commonly called **spots**. Several forms of proteins, when sharing approximately the same pI and mass, may overlap over the same spot. Spots can be visually detected by a variety of means; the most frequently used being silver, coomassie (blue dye) and fluorescent dyes staining (Figure B.III-1). Comparing the proteins' positions to biomarkers, serving as landmarks, lets us estimate the effective values of their pI and Mw. The proteins can then be extracted for further investigation. Several techniques help transferring the proteins out of the gel onto an immobilizing membrane ("blotting"). For more experimental details, a highly detailed tutorial by Gorg including well-described protocols on 2-DE separation with IPGs (IPG-Dat) is available on-line[1].

[1] http://www.weihenstephan.de/blm/deg/manual/manfrm.htm

Alternatively, in comparative studies, some typical approaches used in microarray transcriptomics are promoted: difference gel electrophoresis (DIGE) is the labelling of protein extract with two or three fluorophores, Cy2, Cy3 and/or Cy5, exhibiting different fluorescent spectra (Figure B.III-3). This allows to separate distinct extracts in one gel and to evaluate relative quantitative changes in protein content (Unlu et al. 1997).

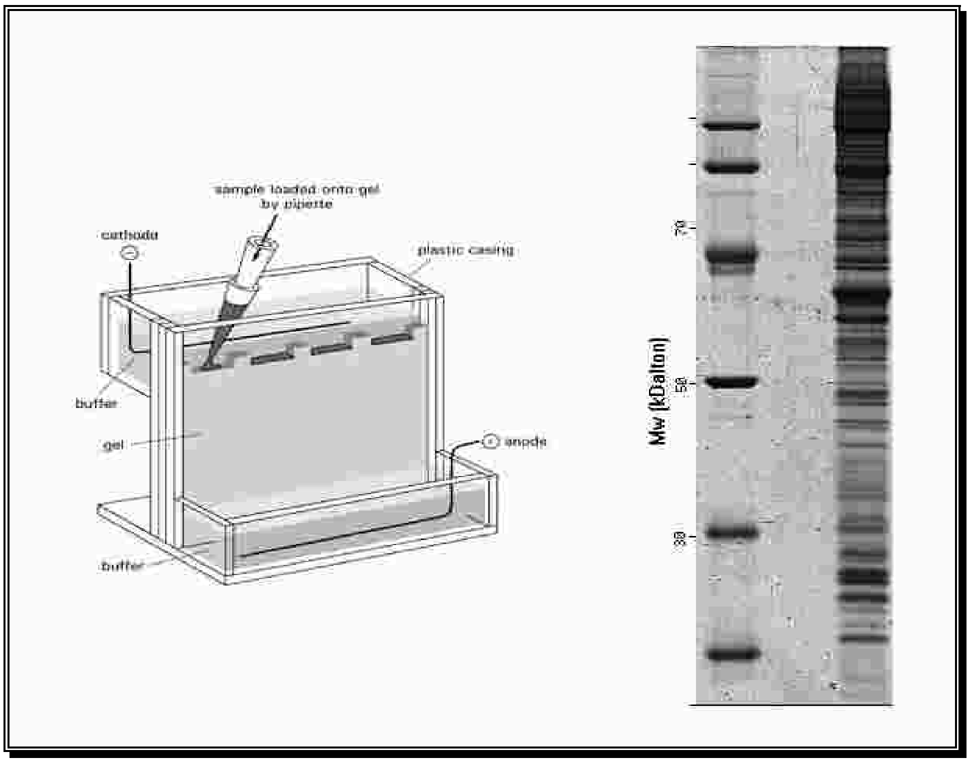


Figure B.III-2: A SDS-PAGE apparatus (left) producing a one dimensional SDS strip (right).

In most cases, the gel is scanned into an image representing a **map** of spots (or of bands in case of 1-D). 2-D gel analysis software then intervenes to detect and quantify the relative abundance of the separate and distinct spots. Experts also commonly use manual visual detection and comparison of spots, especially to adapt potential incorrectness of automatic algorithms' detection.

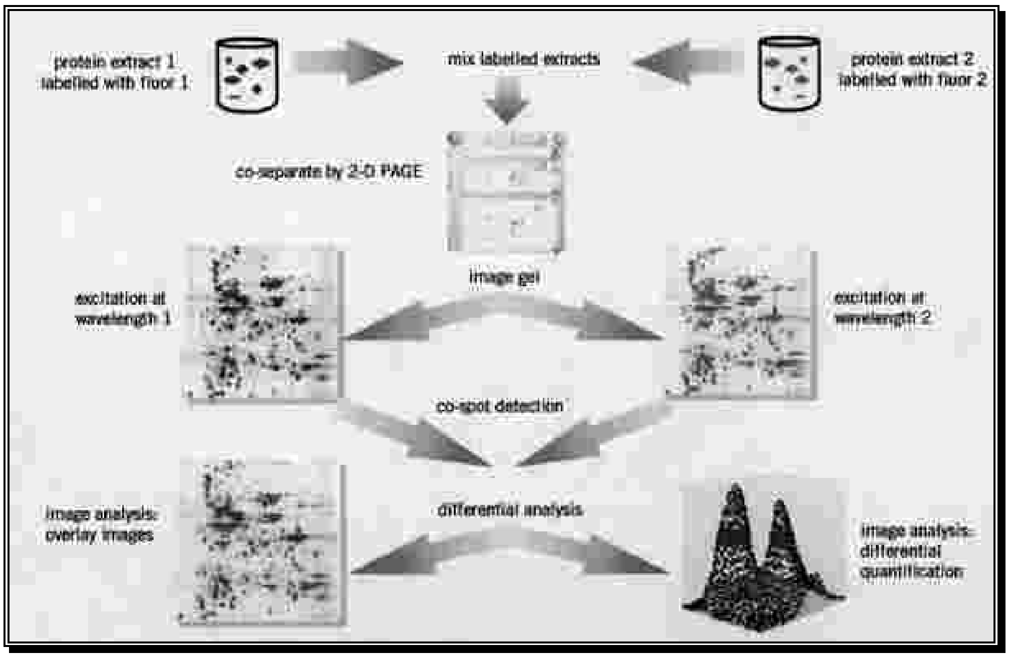

protein extract 1 labelled with fluor 1
mix labelled extracts
protein extract 2 labelled with fluor 2
co-separate by 2-D PAGE
image gel
excitation at wavelength 1
excitation at wavelength 2
co-spot detection
differential analysis
image analysis: overlay images
image analysis: differential quantification

Figure B.III-3: 2-D DIGE technology.[1]

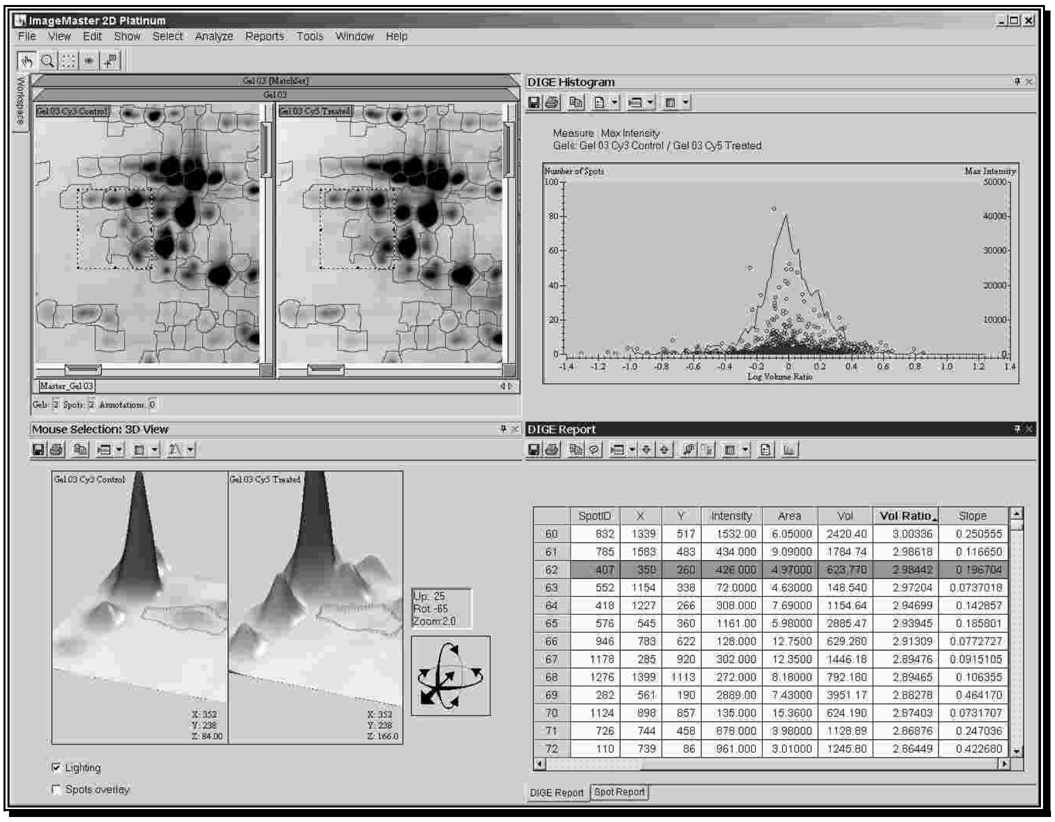
2-DE separation technique has limitations: highly hydrophobic proteins cannot be solubilised; heavy proteins do not migrate uniformly due to encumbrance; and as already mentioned its little sensitivity to low abundant proteins. The issue of incompletely separated (overlapping) proteins, which obstructs quantitative comparison, is a constraint. Still, this last issue may be partially dealt with by the use of narrow range pH gels to increase resolution. At the analysis level, the problem of detecting, quantifying and comparing spots, remains a time-consuming process causing the most significant bottleneck in any attempt for 2-DE automation.

B.III.2 Gel-Informatics

Analysing a scanned gel often requires a number of manual procedures and to use a specialised 2-DE image analysis software tool. For a summarizing list of some available 2-DE software, we may refer to Govorun and Archkov's review on proteomics technologies (Govorun, Archakov 2002). These analysing tools rely primarily on image processing algorithms to detect spot borders and to quantify their relative intensities. Using well-known biomarkers on the gel makes achievable the mapping of the gel, in the sense that coordinates of any point on the gel can be estimated in terms of both pI and Mw axis. In addition, many of these tools have the ability to match spots between diverse similar gels, thus allowing a quantitative comparison of protein expression among many samples. Many of them also attempt to correct running differences between gels by image wrapping. However, as already pointed out, several challenges must be overcome on image mining or in statistical approaches based on multiple gel runs. This is also true for automation of the image analysis process which is currently one of the main bottleneck in 2-DE technology

[1] Credit: Amersham Pharmacia Biotech, Life Science News, 7, 2001.

(Dowsey et al. 2003). Recent attempts to avoid such restrictions include the ProteomeGRID[1], a grid enabled high-throughput infrastructure for full automation of proteomics pipeline from 2-DE gel submission to protein identification and dissemination, with special focus on image-centric analysis for large-scale mining and statistical cross-validation (Dowsey et al. 2004). The project also aims to contribute to the development of standards for 2-DE gel ontology, in concert with the Proteomics Standards Initiative (PSI) (Jones, Gibson 2007). The ProteomeGRID encompasses a 2-DE gel matching algorithm: RAIN (Robust Advanced Image Normalisation) in which image intensity distributions, rather than selected features, are considered (Dowsey et al. 2006).



Measure : Max Intensity
Gels: Gel 03 Cy3 Control / Gel 03 Cy5 Treated

	SpotID	X	Y	Intensity	Area	Vol	Vol Ratio	Slope
60	832	1339	517	1532.00	6.05000	2420.40	3.00336	0.250555
61	785	1583	483	434.000	9.09000	1784.74	2.98618	0.116650
62	407	350	260	426.000	4.97000	623.770	2.98442	0.196704
63	552	1154	338	72.0000	4.63000	148.540	2.97204	0.0737018
64	418	1227	266	308.000	7.69000	1154.64	2.94699	0.142857
65	576	545	360	1161.00	5.98000	2885.47	2.93945	0.185801
66	946	783	622	128.000	12.7500	629.280	2.91309	0.0772727
67	1178	285	920	302.000	12.3500	1446.18	2.89476	0.0915105
68	1276	1399	1113	272.000	8.18000	792.180	2.89465	0.106355
69	282	561	190	2889.00	7.43000	3951.17	2.88278	0.464170
70	1124	898	857	135.000	15.3600	624.190	2.87403	0.0731707
71	726	744	458	878.000	3.98000	1128.89	2.86876	0.247036
72	110	739	86	961.000	3.01000	1245.80	2.86449	0.422680

Figure B.III-4: Melanie / ImageMaster™ 2D Platinum, an example of a 2-DE image software performing gel matching and statistical analysis[2].

B.IV. Accessing proteomics data

Genome and protein sequencing, as well as analyses and characterisation studies and interpretation, cannot be of great interest without being made accessible to the whole scientific community. Complementarities and extrapolation of investigations are

[1] http://www.proteomegrid.org/

[2] http://www.expasy.org/melanie/

essential. Therefore, accessing others' exploration data is vital. The increase of available data is however exponential, in both quantity and diversification, and this amount of data cannot be exploitable without being also well organised and finely interpreted. For the same reason, some guiding rules are necessary when reporting experiments (Bradshaw et al. 2006; Wilkins et al. 2006).

The availability of data resources is the primary sine qua non condition for proteomics investigations. The next chapter introduces some of the most important public and Web-based data resources in proteomics.

C h a p t e r **C**

CHAPTER C. PROTEOMICS DATA RESOURCES

This section describes the nature of data found in some of the most important resources in protein classification and characterisation with regard to our work.

C.I. Introduction

Protein sequence databases are historically the oldest resources ever created in molecular biology. "Atlas of Protein Sequence and Structure" was first published in 1965 by Dayhoff, and contained about a hundred sequences of uncharacterised proteins (Dayhoff et al. 1965). A few years later, PDB, the Protein Data Bank (Berman et al. 2007) was created as a repository for three-dimensional structure data of experimentally determined biological macromolecules. Since 1977, it has become much easier to obtain DNA sequences than protein sequences, and at the time being, many genomes have been entirely sequenced. Gene prediction processes, aiming to locate functional sequences on a genome, have also progressed. Translated protein sequences have started to be automatically derived from CDS sequences, based on translation and coding rules. Consequently, an explosion of the amount of available data has occurred, resulting in an exponential growth of protein sequence databases. In the meanwhile, and with the evolution of protein separation and characterisation techniques, as well as the rapid development of communication and exchange facilities promoted by the Internet, a large array of general and specific data resources, covering all the many aspects of proteomics, emerged all over the world. A valuable survey in the "Proteome Research" book, though not entirely up-to-date by the time of this writing, can be consulted (Bairoch 1997).

In this section, we describe the nature of data found in some of the most important resources in protein classification and characterisation with regard to our work. Category labelling is principally brought in to facilitate the reading of this chapter. However, boundaries between categories should not be considered rigid.

We will not go through details concerning technical aspects of data structure or management at this stage, as this will be carried out lately in a dedicated chapter. Proteomics databases dedicated to protein separation and identification methods, in particular 2-DE databases, will be presented on their own in sections C.IV and C.V. Being at the heart of our investigation, 2-DE datasets will particularly benefit from our special attention.

C.II. Protein sequence resources

As already mentioned, most of the comprehensive source of information on proteins is contained within the protein sequence databases, as a result of the huge amount of translated coding genes. Sequence databases are either universal resources covering as many various organisms and types of proteins without explicit distinction, or more specific databases centred on a specific organism or a protein family. The universal resources are either highly curated databases or simple repositories containing DNA translated sequences, while the specific databases always require domain specialists to annotate and process their content.

C.II.1 UniProtKB/Swiss-Prot

The UniProtKB/Swiss-Prot protein knowledgebase[1] was, since 1986 until quite recently, only referred to as Swiss-Prot. Maintained by both the Swiss Institute of Bioinformatics (SIB) and the European Bioinformatics Institute (EBI), it is by far one of the most mature and most important protein sequence databases (Bairoch et al. 2004). Data within Swiss-Prot is not just a collection of translated sequences. The non redundant sequences it contains are extensively annotated, with high quality information about biological functions, structure, isoforms, PTMs, protein family, bibliographic references, etc. (Boeckmann et al. 2005). It also provides many external links (cross-links) to a variety of other resources. This kind of annotation is inferred or directly imported from publications or submissions, and is performed by domain experts. Due to its robustness and richness, Swiss-Prot has become since long a reference in protein designation. The term ‘knowledgebase’ itself is quite appropriate, as it shows the main characteristic of the database: being a comprehensive compilation of contributions made by a body of knowledge.

Some in-house controlled vocabularies employed by the database annotators have gained acceptance. Besides, to designate a specific protein, scientists worldwide commonly use the Swiss-Prot **accession numbers**, the identifiers assigned to label proteins in Swiss-Prot. Those identifiers habitually constitute what we consider the **main index**, an index that allows unambiguous designation of individual proteins. This is even reinforced by the recent developments that have seen all protein orthologs totally demerged within the Swiss-Prot database.

At the same time, Swiss-Prot also maintains many specific projects, like Human Protein Initiative HPI, HAMAP for microbial protein families’ annotation, the ENZYME nomenclature database, or the NEWT taxonomy database. Information in Swiss-Prot is built and is generally presented in text records, called entries. Each protein has a distinct text entry, where information is listed in separated keyword/value(s) sections. Different types of lines, each with their own format, are used to record the various data that make up the entry. Table C.II-1 is an example of an annotated entry. It begins with technical information about the entry (identifiers and dates of modifications), the origin of the protein (descriptive name, gene of origin and organism), a bibliographic section and a comments section on biological facts. Cross-references to other external resources and keywords are listed, as well as a dedicated section (the feature table) for the annotation of the sequence and regions of interest. At the end of the entry, a consensus sequence of the protein - usually the sequence of the longest isoform - is given in standard IUPAC one-letter codes. Historically, entries are presented sequentially one after another in a single text file, called a **flat file**, for distribution purpose. Release 52.2 / April 2007 of Swiss-Prot contains more than 263’000 distinct entries.

[1] http://www.expasy.org/swiss-prot

Table C.II-1: A UniProtKB/Swiss-Prot protein entry, <P22222>, in raw text format.[1]

```
ID   E13B_CELCE              Reviewed;         548 AA.
AC   P22222;
DT   01-AUG-1991, integrated into UniProtKB/Swiss-Prot.
DT   01-AUG-1991, sequence version 1.
DT   31-OCT-2006, entry version 46.
DE   Glucan endo-1,3-beta-glucosidase precursor (EC 3.2.1.39) ((1->3)-beta-
DE   glucan endohydrolase) ((1->3)-beta-glucanase).
OS   Cellulosimicrobium cellulans (Arthrobacter luteus).
OC   Bacteria; Actinobacteria; Actinobacteridae; Actinomycetales;
OC   Micrococcineae; Promicromonosporaceae; Cellulosimicrobium.
OX   NCBI_TaxID=1710;
RN   [1]
RP   NUCLEOTIDE SEQUENCE [GENOMIC DNA], AND PROTEIN SEQUENCE OF 37-63.
RX   MEDLINE=91093212; PubMed=1985933;
RA   Shen S.-H., Chretien P., Bastien L., Slilaty S.N.;
RT   "Primary sequence of the glucanase gene from Oerskovia
RT   xanthineolytica. Expression and purification of the enzyme from
RT   Escherichia coli.";
RL   J. Biol. Chem. 266:1058-1063(1991).
CC   -!- FUNCTION: Lysis of cellular walls containing beta-1,3-glucans.
CC       Implicated in the defense against fungal pathogens.
CC   -!- CATALYTIC ACTIVITY: Hydrolysis of 1,3-beta-D-glucosidic linkages
CC       in 1,3-beta-D-glucans.
CC   -!- SUBCELLULAR LOCATION: Periplasm.
CC   -!- SIMILARITY: Belongs to the glycosyl hydrolase 64 family.
CC   -!- SIMILARITY: Contains 1 ricin B-type lectin domain.
DR   EMBL; M60826; AAA25520.1; -; Genomic_DNA.
DR   PIR; A39094; A39094.
DR   HSSP; P26514; 1KNM.
DR   GO; GO:0042973; F:glucan endo-1,3-beta-D-glucosidase activity; IEA:EC.
DR   InterPro; IPR000772; Ricin_B_lectin.
DR   InterPro; IPR008997; RicinB_like.
DR   Pfam; PF00652; Ricin_B_lectin; 1.
DR   SMART; SM00458; RICIN; 1.
DR   PROSITE; PS50231; RICIN_B_LECTIN; 1.
KW   Cell wall; Direct protein sequencing; Glycosidase; Hydrolase; Lectin;
KW   Periplasmic; Signal.
FT   SIGNAL        1     36       Potential.
FT   CHAIN        37    548       Glucan endo-1,3-beta-glucosidase.
FT                                /FTId=PRO_0000012235.
FT   DOMAIN      422    548       Ricin B-type lectin.
FT   REGION       37    430       Possess beta-glucanase activity, but is
FT                                unable to lyse viable cells.
FT   REGION      472    548       Essential for the lytic activity, but not
FT                                for the beta-glucanase function.
SQ   SEQUENCE   548 AA;  58089 MW;  412B5A4AA24C048D CRC64;
     MPHDRKNSSR RAWAALCAAV LAVSGALVGV AAPASAVPAT IPLTITNDSG RGPIYLYVLG
     … … …
     GTADGTAVWI YTCNGTGAQK WTYDSATKAL RNPQSGKCLD AQGGAPLRDG QKVQLWTCNQ
     TEAQRWTL
//
```

C.II.2 UniProtKB/TrEMBL

TrEMBL (TRanslation of EMBL nucleotide sequence database) is, as its name suggests, a collection of translated sequences originating from the EMBL database. Translation of the coding nucleotide sequences is done automatically, and entries contain only computer-generated annotations. TrEMBL was created in 1996 as a complement to Swiss-Prot for protein sequences that are not yet analysed, annotated and incorporated in Swiss-Prot. TrEMBL entries follow strictly the format of Swiss-

[1] The amino acid sequence has been shortened and the copyright text removed to save space.

Prot entries. Sequences are kept in TrEMBL until they are analysed, annotated, and moved to Swiss-Prot. The process of incorporation of entries in TrEMBL entails translation of CDS from EMBL with redundancy removal performed to some extent. A standardised transfer of corresponding annotations from Swiss-Prot to non-annotated TrEMBL entries that belong to already defined protein groups is carried out (Apweiler et al. 2004). This is done by similarity between homologous proteins using InterPro, the integrated resource of protein families, domains, and functional sites (Mulder et al. 2007). TrEMBL Release 35.5 / April 2007 contains more than 423 thousands entries.

C.II.3 PIR-PSD

The Protein Sequence Database (PSD) is the main database of the three different databases forming the Protein Information Resource[1] (PIR) and is developed at the Georgetown University Medical Center (Wu et al. 2003). PIR-PSD is characterised by the clustering of sequences in non-overlapping super-families, and by an efficient bibliography information system. PIR-PSD has been fully integrated in UniProtKB (*cf.* next description).

C.II.4 UniProtKB: The Universal Protein database

UniProt is a consortium formed by the previous three databases. The UniProt Knowledgebase[2] merges Swiss-Prot, TrEMBL and PIR-PSD to provide a central universal database of protein sequences with annotations and functional information (Bairoch et al. 2005). In practice, and in addition to all Swiss-Prot and TrEMBL entries, suitable PIR-PSD sequences missing from the other two databases have been incorporated within the knowledge database. Conversely, bi-directional cross-references have been added to ensure the tracking of PIR-PSD entries. As it was mentioned before, the distinct separation between highly annotated and non-annotated entries is kept within UniProtKB: this explains why we still distinctly cite UniProtKB/Swiss-Prot and UniProtKB/TrEMBL.

In addition to the central UniProt Knowledgebase, the consortium maintains two other databases. One is the UniProt Archive (UniParc), a complete and non-redundant collection of the all publicly available protein sequences, but with no annotations. The other is UniProt Reference (UniRef), a non-redundant data collections based on the clustering of UniProt Knowledgebase and UniParc. Clustering is made at three identity levels (100%, >90% and > 50%) and is helpful in fast sequence similarity search.

C.II.5 Some notes on major nucleotide sequence databases

In contrast to how annotations are carried out in UniProtKB, it is important to know how annotations are produced in the original nucleotide sequence database source from where most protein sequences are translated. Entries that are submitted to the International Nucleotide Sequence Database Collaboration (GeneBank / EMBL / DDJB, *cf.* B.I.1) are the property of their submitters. They contain annotations that no one except the original submitter can modify, which strongly differ from UniProtKB

[1] http://pir.georgetown.edu/

[2] http://www.uniprot.org

policy. As a result, potential errors, redundancy, and heterogeneity of used terms often occur and persist. A non-linearity between the UniProtKB and the original nucleotide Universal databases is therefore expected. Table C.II-2 gives the EMBL nucleotide sequence entry that has been translated into the Swiss-Prot <P22222> entry.

Table C.II-2: Entry <M60826> from the EMBL database.[1]

```
ID   M60826; SV 1; linear; genomic DNA; STD; PRO; 2697 BP.
XX
AC   M60826; M38734;
XX
DT   07-FEB-1991 (Rel. 27, Created)
DT   17-APR-2005 (Rel. 83, Last updated, Version 4)
XX
DE   O.xanthineolytica beta-1,3-glucanase gene, complete cds.
XX
KW   beta-1,3-glucanase.
XX
OS   Cellulosimicrobium cellulans
OC   Bacteria; Actinobacteria; Actinobacteridae; Actinomycetales;
OC   Micrococcineae; Promicromonosporaceae; Cellulosimicrobium.
XX
RN   [1]
RP   1-2697
RX   PUBMED; 1985933.
RA   Shen S.-H.H., Chretien P., Bastien L., Slilaty S.N.;
RT   "Primary sequence of the glucanase gene from Oerskovia xanthineolytica:
RT   Expression and purification of the enzyme from Escherichia coli";
RL   J. Biol. Chem. 266(2):1058-1063(1991).
XX
FH   Key             Location/Qualifiers
FH
FT   source          1..2697
FT                   /organism="Cellulosimicrobium cellulans"
FT                   /mol_type="genomic DNA"
FT                   /db_xref="taxon:1710"
FT   CDS             463..2109
FT                   /codon_start=1
FT                   /transl_table=11
FT                   /gene="beta-1,3-glucanase"
FT                   /product="beta-1,3-glucanase"
FT                   /note="putative"
FT                   /db_xref="GOA:P22222"
FT                   /db_xref="InterPro:IPR000772"
FT                   /db_xref="InterPro:IPR008997"
FT                   /db_xref="UniProtKB/Swiss-Prot:P22222"
FT                   /protein_id="AAA25520.1"
FT                   /translation="MPHDRKNSSRRAWAALCAAVLAVSGALVGVAAPASAVPATIPLTI
FT                   TNDSGRGPIYLYVLGERDGVAGWADAGGTFHPWPGGVGPVPVPAPDASIAGPGPGQSVT
FT                   … … …
FT                   CLDAQGGAPLRDGQKVQLWTCNQTEAQRWTL"
XX
SQ   Sequence 2697 BP; 402 A; 1004 C; 954 G; 337 T; 0 other;
     ggatcccgag caccggggcg tcggtggtgc cggtgacgac catcttcgcc ttgttgcgga        60

     … … …

     cgtccatcac cgcgttctcc accagcaccg gcacgaagtc gccacgtgcg cggcctg         2697
//
```

[1] The DNA sequence and the translated amino acid sequence have been shortened to save space.

C.II.6 NCBI sequence repositories

Entrez Protein[1] is another sequence database built at NCBI (National Centre of Biotechnology Information). Proteins being slightly or not at all curated, the database is more of a repository of translated nucleotide sequences and a compilation from other protein sequence databases than a value-added protein database. Redundancy is significant. RefSeq[2] is a collection with a better approach, grouping non-redundant sets of nucleotide, transcript, and protein sequences with some level of annotation.

NCBI protein identifiers are widely adopted by researchers similarly to UniProtKB identifiers. It is significant to note that mapping between NCBI protein sequence databases and UniProtKB is not a straightforward task, especially when dealing with the relatively unstable NCBI GenInfo identifiers (GI numbers). Most importantly, and due to the fact that relation between a protein sequence and its "original" coding gene is not necessarily a one to one relation, gene assignment between UniProtKB and NCBI systems diverges very much. Unfortunately, this situation causes difficulties when trying to automatically unify or integrate proteomics resources. An interesting solution has been recently proposed by Babnigg through the Sequence Globally Unique Identifier (SEGUID) based on the generation of identifiers from digested proteins' primary 1-D sequences (Babnigg, Giometti 2006). Unfortunately, this solution is limited by the fact that not all primary sequences are in fact derived from a direct one to one genome translation process, *e.g.*, submitted sequences and isoform annotations for the same protein in UniProtKB/Swiss-Prot. Nevertheless, and without abandoning the LSID concept, such an approach could generate an additional protein property of practical interest in integration processes.

C.II.7 Organism-specific protein sequences

There are a large number of species-specific protein sequence databases. Those databases are generally highly annotated, due to the specificity of their purpose. An example is YPD, the Yeast Protein Database, which is the first annotated database for a complete proteome (Payne, Garrels 1997). Organism-specific protein sequences databases are important resources of information with high quality data provided by domain specialists. The main inconvenience is the heterogeneity of their formats and contents, which makes it difficult to uniformly query them or to compare their content.

C.III. Other categories

Numerous resources covering many areas of protein characterisation are available. Several of these areas have an importance - either directly or indirectly - for the purpose and the evolvement of our work. We present here a little survey of some of these areas.

[1] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Protein

[2] http://www.ncbi.nlm.nih.gov/RefSeq/

Protein classification

The Gene Ontology Annotation group (GOA) at EBI[1] is developing GO[2] functional annotations to proteins using a hierarchical arrangement (Camon et al. 2004; Lee et al. 2005). The annotations consist of three structured, controlled vocabularies (ontologies) that describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner. The use of GO terms by several collaborating databases facilitates uniform queries across them. The controlled vocabularies are structured so that they can be queried at different levels.

Metabolic and enzyme databases

Computerizing knowledge of molecular and cellular biology systems in terms of pathways information is not a straightforward task. KEGG Pathway, which is part of the integrative KEGG database from the Kyoto Encyclopedia of Genes and Genomes[3], consists of graphical diagrams of metabolic pathways, and is actually not specific to any particular organism (Kanehisa et al. 2006). BRENDA[4] (BRaunshweig ENzyme DAtabase) is a collection of enzyme and metabolic information with specific species and enzyme sources or tissues classification (Schomburg et al. 2004). ENZYME[5], developed at SIB, is based on recommendations of IUBMB[6] and it contains identity and activity information on characterised enzymes that are assigned an enzyme commission code: EC ([No authors listed] 1999). The database is helpful in the development of metabolic databases (Bairoch 2000). A more integrative platform that allows the storage, management, and analysis of interrelated proteins, genes, interactions, protein families and cellular pathways, is developed at Cornell University: the Biozon platform (Birkland, Yona 2006a). Biozon warehouses existing published data from sources such as PDB, Genbank, Uniprot, KEGG, and BIND and integrates them using in-house derived approach.

Pattern and profiling databases

Particular regions of proteins may play a role, or simply are responsible for a specific activity or function of the aforementioned proteins. Relationships between different proteins from a set can be revealed through clustering by looking for conserved regions at the sequence and the structure level. This leads to the discovery of patterns and profiles that help in identifying members belonging to the same family.

We have already introduced the InterPro integrative system[7] as an organised and documented collection of protein families. It interrelate protein families, domains and functional sites from the trustworthiest pattern and profiling databases, such as the well-

[1] http://www.ebi.ac.uk/GOA/

[2] http://www.geneontology.org

[3] http://www.genome.jp/kegg/

[4] http://www.brenda.uni-koeln.de/

[5] http://www.expasy.org/enzyme/

[6] Nomenclature Committee of the International Union of Biochemistry and Molecular Biology.

[7] http://www.ebi.ac.uk/interpro/

known Pfam[1], PROSITE[2], Prints, ProDom or TIGRFAMs. Entries are organised in these systems by type (family, domain, repeat and PTM, active and binding sites) with a hierarchical parent-child relationship. Other significant profiling databases include Blocks[3] and Prints[4].

Analysis covering these domains is based on alignment identity and is often referred to as 'domain architecture'. The information contained in pattern and profiling databases is essentially statistical and predicted information.

Molecular interaction databases

Understanding proteomics, and more globally the cell machinery, presupposes a better portrayal of protein interactions and linked networks. Protein-protein interactions (PPI) - and more generally molecular interactions (MI) datasets - are often the result of curation and reporting publications covering the subject, and thus can be hardly automated. Before the recent efforts of standardisation in proteomics and the rapid advances made in molecular interactions formulation standards (Hermjakob et al. 2004), it was scarcely possible to compare to each other the many existent protein-protein interactions datasets, such as IntAct (Kerrien et al. 2007), MINT (Chatr-aryamontri et al. 2007) or BIND (Bader et al. 2003). Since the last few years, retrieval of relevant data from different datasets and data comparison can be performed consistently. More insights on the standardisation efforts are provided in next chapter.

Immunohistochemistry

Proteins may be directly spotted in cells of a tissue section exploiting specific binding of antibodies to antigens. A particular protein can be viewed under a microscope using specific fluorescent-labelled or radioactive antibodies. Studying the localisation, the distribution, and measuring the intensity of target proteins can then engender significant functional information about their characteristics. The technique can even be modified to visualise more than one protein alongside. Recently, the "Human Protein Atlas"[5] knowledge base for normal and cancer tissues has been released with more than 400 thousands annotated images corresponding to more than 700 antibodies toward human proteins (Uhlen et al. 2005). The atlas offers links to the principle protein databases. This represents an innovative and interesting approach, as it connects directly perceived systems biology observations with formulised and annotated data related to the actor sub-units involved in the biological process. This joins the so called '*inductive*' research approach in which data is generated first and then analysed to look for interesting models and to find out new hypothesis. An approach that is particularly applicable to systems biology where the whole system "*is viewed as a delicately balanced interplay of a multitude of processes, which can be studied in a global fashion using different types of technologies*" (Suresh et al. 2005).

[1] http://www.sanger.ac.uk/Software/Pfam/

[2] http://www.expasy.org/prosite/

[3] http://blocks.fhcrc.org/

[4] http://www.bioinf.manchester.ac.uk/dbbrowser/PRINTS/

[5] http://www.proteinatlas.org/

C.IV. 2D-PAGE (2-DE) datasets

C.IV.1 Many 2-DE datasets are in loose formats

Sequence databases and repositories are typically large and centralised datasets that are habitually presented in a well defined or agreed on format. By opposition, 2-DE datasets, as well as many similar collections of experimental data, are usually limited in their contents and are often specific in their formats to people and laboratories that are producing them. They result in many small datasets, often isolated, all over the world. Besides, many of those resources are in loose formats and are often unstructured and unrefined. Nevertheless, some resources of significant importance are also available on the Web and offer practical solutions for parsing and accessing data. Many of those resources are collections of protein reference maps that are very helpful in repeated or related analysis due to the high reproducibility of the separation method. Without being exhaustive, and without focusing on how data is internally organised or managed, we are going to list here some of the major 2-DE datasets that are available on the Web.

C.IV.2 The early federation approach

In 1996, in an attempt to federate 2-DE datasets with some level of integration, Appel proposed a simple and efficient way to federate and interconnect remote 2-DE datasets (Appel et al. 1996). Those rules can be summarised as follows[1]:

Rule 1: Individual entries in the database must be remotely accessible by keyword search. Other query methods, such as full text search, for example, are possible but not required.

Rule 2: The database must be linked to other databases through active hypertext cross-references, so that through a simple mouse click on a cross-reference, the user automatically is connected to the corresponding WWW site, and the cross-referenced document is then retrieved and displayed. This simple mechanism links together all related databases and combines them into one large virtual database. Database entries must have such a cross-reference to at least the main index (see Rule 3).

Rule 3: In addition to individually searchable databases, a main index that provides a means of querying all databases through one unique entry point has to be supplied. Bi-directional cross-references must exist between the main index and the other databases. Presently, the UniProtKB/Swiss-Prot knowledgebase identifiers define this main index.

Rule 4: Individual protein entries must be accessible through clickable images. That is to say that 2-DE images must be provided on the WWW server and that as a response to a mouse click on any identified spot on an image, the user must obtain the database entry for the corresponding protein. This method enables a user to easily identify proteins on a 2-DE image.

[1] http://expasy.org/ch2d/fed-rules.html

Rule 5: 2-DE analysis software, that have been designed for use with federated databases, must be able to directly access individual entries in any federated 2-DE database. For example, when displaying a 2-DE reference map with a 2-DE computer program, the user must be able to select a spot and remotely obtain the corresponding entry from the given database.

Though very simple, these conventions proved their efficiency and they were adopted by many databases that made their appearance on the Web. The first of them was the well-known SWISS-2DPAGE database. A database that is central to our work.

C.IV.3 SWISS-2DPAGE

General facts

The first and one of the largest databases of 2-DE reference maps on the Web is SWISS-2DPAGE which is maintained by our group, the Proteomics Informatics Group at the Swiss Institute of Bioinformatics (SIB), as well as the Biomedical Proteomics Research Group (BPRG) of the Geneva University Hospital (Hoogland et al. 2004). It was established and made available for the first time as far back as 1993, and is accessible from the address *http://world-2dpage.expasy.org/ch2d/*[1] (Appel et al. 1993). With more than 1000 hits per day, it is the second most accessed database on the ExPASy proteomics server (Gasteiger et al. 2003) after the UniProtKB/Swiss-Prot.

Due to the proximity of its developers to the Swiss-Prot developers, the 2-DE database has adopted from its very beginning a format analogous to the one used by the sequence database: a keyword/value(s) list of annotations for individual proteins identified on the various maps. The distribution of the database is also largely supported by means of a flat file, a plain text format listing sequentially all the identified protein entries one after another. Annotation in SWISS-2DPAGE is of high quality standard at both descriptive and experimental levels. It includes many reference maps and a high number of identified proteins. Images show the experimental location of proteins and information on spots' properties is displayed. The images can also be clicked to access the various protein entries. Theoretical location area over the different maps can be computed for any known protein sequence based on its amino acid composition. All this features makes the database a relevant resource in protein investigation and biomarker discovery.

In its current version, release 18.6 / January 2008, SWISS-2DPAGE contains 1265 protein entries, identified from 3976 spots over 36 reference maps and covering 7

[1] Search engine at: http://www.expasy.org/swiss-2dpage/

different species (mainly Homo sapiens, Mus musculus and Escherichia coli). Documents describing statistical facts, experimental information about protocols, chemicals and apparatus, as well as 75 bibliographic references, support the database content. A variety of methods is used to identify spots (Table C.IV-1). Several identification methods are often commonly applied to the same spot, making characterisation much more trustworthy.

Table C.IV-1: Identification methods distribution on SWISS-2DPAGE

Identification Method	Abbreviation	Number of Spots	Frequency (%)
Amino acid composition	Aa	179	4.50
Comigration	Co	88	2.21
Gel matching	Gm	1594	40.09
Immunobloting	Im	680	17.10
Microsequencing	Mi	527	13.25
Peptide mass fingerprinting	PMF	1108	27.87
Tandem mass spectrometry	MS/MS	741	18.64

Despite its high level of annotation, SWISS-2DPAGE can sometimes lack in providing full details on identification processes. Although some of this data may be present in the related bibliographic publications, some deficiency may be sensed when accessing the database itself looking for more details. For example, mass spectrometry selected peak values are listed without their intensities, and barely any detailed justification on protein assignment is given. Users need then to further investigate to judge of the quality of the assignment. Many of such identification data are internally accessible to the database maintainers (in diverse formats). However, they ought to be integrated within the database. The current management system of the database makes it straightforward to integrate this missing data.

At the core of our development

Until recently, SWISS-2DPAGE was essentially a collection of independent text entries assembled sequentially in one plain text file of linear structure, the flat file, with autonomous additional supporting text documents. Annotations were performed individually on each entry. Redundancy and rigidity are extremely high in such a format. Furthermore, time was needed to ensure consistency and error-free annotations. Moreover, and due to the linearity of relations, some data types and associations had to be represented via the application of strict writing rules. For example, associating peptide sequences with the MS/MS peak values from which they have been deduced could hardly be done straightforwardly. If such situations could sometimes be manageable, it was however obvious that a more structured format was needed. Therefore, restructuring SWISS-2DPAGE into an appropriate data model and using a more efficient management system was one of our main initial motivations, in addition to ensuring that any external information is valid and up-to-date. Even though our project is intended to be fairly generic and flexible, SWISS-2DPAGE has regularly been at the core of the project conception. The database has then greatly influenced and inspired our data model in its initial phases, though without being its sole and exclusive point of focus. Since late 2006, the official public version of SWISS-2DPAGE has fully adopted our new data representation and management system (system version: 2.50.2 / October 2006).

An entry description

As already mentioned, a protein entry, when presented to users in its most basic and raw view, is a list of two-character codes (data types) each followed by corresponding annotations that are readable by humans. Some data types are mandatory, some others are optional, and some may be repeated over several lines. Almost all annotations follow a precise predefined syntax.

Those views were originally the principal components that were directly used 'as is' to build up the database body. An extensive description of annotations' syntax intended for this format is given at[1]:

- *http://www.expasy.org/ch2d/manch2d.html*

Due to the inheritance of many SWISS-2DPAGE keywords from UniProtKB/Swiss-Prot, more information is described in the sequence database user manual at:

- *http://www.expasy.org/sprot/userman.html*

We are presenting hereafter some of the most important features of an entry in its original row format, because of their importance to understand part of the logic in our development choices. Some notions are introduced here but will be developed when depicting the data model that we have adopted. We chose a representative entry - <P0AB71> - that contains some of the elements common to all entries (Table C.IV-2).

Table C.IV-2: SWISS-2DPAGE entry <P0AB71>, previously <P11604>, release 18.3.[2]

```
ID   ALF_ECOLI; STANDARD; 2DG.
AC   P0AB71; P11604;
DT   01-SEP-1997, integrated into SWISS-2DPAGE (release 6).
DT   15-MAY-2003, 2D annotation version 4.
DT   14-NOV-2006, general annotation version 11.
DE   Fructose-bisphosphate aldolase class 2 (EC 4.1.2.13)
DE   (Fructose-bisphosphate aldolase class II) (FBP aldolase).
GN   Name=fbaA; Synonyms=fba, fda; OrderedLocusNames=b2925, JW2892;
OS   Escherichia coli.
OC   Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriales;
OC   Enterobacteriaceae; Escherichia.
OX   NCBI_TaxID=562;
MT   ECOLI, ECOLI-DIGE4.5-6.5, ECOLI5-6.
IM   ECOLI, ECOLI-DIGE4.5-6.5, ECOLI5-6.
RN   [1]
```

[1] Some minor details may still need some update to suite the current format changes.

[2] Copyright text removed to save space.

```
RP   MAPPING ON GEL.
RX   MEDLINE=98410772; PubMed=9740056;
RA   Tonella L., Walsh B.J., Sanchez J.-C., Ou K., Wilkins M.R., Tyler M.,
RA   Frutiger S., Gooley A.A., Pescaru I., Appel R.D., Yan J.X., Bairoch A.,
RA   Hoogland C., Morch F.S., Hughes G.J., Williams K.L., Hochstrasser D.F.;
RT   '''98 Escherichia coli SWISS-2DPAGE database update'';
RL   Electrophoresis 19:1960-1971(1998).
RN   [2]
RP   MAPPING ON GEL.
RX   PubMed=11680886;
RA   Tonella L., Hoogland C., Binz P.-A., Appel R.D., Hochstrasser D.F.,
RA   Sanchez J.-C.;
RT   ''New perspectives in the Escherichia coli proteome investigation'';
RL   Proteomics 1:409-423(2001).
RN   [3]
RP   MAPPING ON GEL.
RX   PubMed=12469338;
RA   Yan J.X., Devenish A.T., Wait R., Stone T., Lewis S., Fowler S.;
RT   ''Fluorescence 2-D difference gel electrophoresis and mass spectrometry
RT   based proteomic analysis of E. coli'';
RL   Proteomics 2:1682-1698(2002).
2D   -!- MASTER: ECOLI;
2D   -!-   PI/MW: SPOT 2D-000L0H=5.55/40732;
2D   -!-   PI/MW: SPOT 2D-000L1R=5.43/39855;
2D   -!-   AMINO ACID COMPOSITION: SPOT 2D-000L1R: B=10.9, Z=10.5, S=7.2, H=3,
2D         G=10.5, T=5.3, A=9.4, P=4.3, Y=3.6, R=3.2, V=7.4, M=1.7, I=5.3, L=8.6,
2D         F=4.3, K=4.8;
2D   -!-   MAPPING: AMINO ACID COMPOSITION AND SEQUENCE TAG (SKIF) [1].
2D   -!- MASTER: ECOLI-DIGE4.5-6.5;
2D   -!-   PI/MW: SPOT 2D-001WMY=5.49/39104;
2D   -!-   PEPTIDE MASSES: SPOT 2D-001WMY: 955.51; 1502.78; 1762.98; 1878.01; TRYPSIN.
2D   -!-   MAPPING: Peptide mass fingerprinting [3].
2D   -!- MASTER: ECOLI5-6;
2D   -!-   PI/MW: SPOT 2D-001L5L=5.56/50220;
2D   -!-   PI/MW: SPOT 2D-001L6U=5.56/49421;
2D   -!-   PEPTIDE MASSES: SPOT 2D-001L5L: 1320.801; 1502.988; 1878.257;
2D         2591.649; 2719.736; 2871.916; TRYPSIN.
2D   -!-   PEPTIDE MASSES: SPOT 2D-001L6U: 934.591; 950.583; 953.573;
2D         1320.797; 1502.947; 1878.221; 2591.534; 2719.66; TRYPSIN.
2D   -!-   MAPPING: Peptide mass fingerprinting [2].
DR   UniProtKB/Swiss-Prot; P0AB71; ALF_ECOLI.
//
```

As shown in the example given above, an entry sequentially lists facts about the protein identity and its origin, information about the history record and the literature citations of experimental analysis. In addition, physical facts and identification details on the various spots from where the protein has been identified are specified for each master / reference map. The entry ends with a list of cross-references to other relevant databases.

Identity of the protein and its history

The identity of a protein consists in a common mnemonic but not stable protein name (*ID*) that often originates from UniProtKB/Swiss-Prot; otherwise, *ID* is just a duplicate of the unique identifier: the accession number (*AC*). The latter is the real and sole stable identifier of the protein entry. An *AC* is always conserved to specifically designate its corresponding protein. Accession numbers may be, in theory, any alphanumerical string. By convention, Swiss-Prot patterns are exclusively adopted. It may happen, in some cases, that a protein holds several secondary accession numbers. In fact, at a given time, when biologically appropriate, a protein entry may be split (demerged) into two or more new entries (*e.g.*, a split into distinct species orthologs), or, on the contrary, be merged with some other protein entry (*e.g.*, when it finally occurs that it is the same protein). In this case, a new *AC* is always generated. To ensure the possibility to track back the newly labelled entries, the older identifiers are conserved in the record and are then considered as secondary accession numbers. In our specific example, this means that entry <P0AB71> was once labelled <P11604>. The *ID* line contains also some technical vocabulary controlled information about the current state of the entry, particularly if data shown is complete and verified, or if it is still in a preliminary phase. Comments on the many aspects of the protein are organised in topics, containing each some free text information, and may be as well appended to the entry (the comments *CC* lines are not present in our example).

Lines with the *DT* keyword covers information related to the entry creation date and gives indication of the public releases where it has been lastly modified, as either the general or the experimental annotations are concerned. The *DE* line is an ordinary English scientific description of the protein.

Origin of the protein

Genes coding for the protein, their aliases and ordered locus names (*ORF* number in a sequenced chromosome) if any are listed in the *GN* line. The organism species name, taxonomic classification and an identifier in a taxonomy database (if available) are given by the *OS*, *OC* and *OX* section.

Literature citations

Those lines (all lines beginning with the character "*R*") indicate the source from which the experimental data has been abstracted. They are organised in blocks defining each a distinct reference. The information comprises the type and nature of the publication, its location/citation, title and authors. Optional cross-references to bibliographic databases (*e.g.*, PubMed) may also be given.

Maps and spots, with identification methods and data

The *MT* line lists the master maps in which the protein has been identified and *IM* the images' names associated with the maps. All 2-DE (or SDS) related data, as well as the identification details are covered in the various *2D* (or *1D*) blocks. Those blocks may start with general comment topics similar to the ones in the protein comments lines about the mapping (identification) procedures as a whole (not present in our example, *cf.* <P99015>[1]). Relevant data on the physical properties and the location of the spots on each map is given with subsequent identification methods and related experimental data and analysis (when available). Free topics specific to maps and spot expression, mapping, protein polymorphism, etc. constitute the ending lines of the 2D blocks (neither present in our example, *cf.* <P00734>[2]).

Database cross-references

These are the pointers ensuring one of the simplest manners to link SWISS-2DPAGE entries to related entries found in other resources. The use of the

[1] http://www.expasy.org/swiss-2dpage/ac=P99015&format=text

[2] http://www.expasy.org/swiss-2dpage/ac=P00734&format=text

UniProtKB/Swiss-Prot main index, currently the most common protein index, ensures that the database is able to point to other databases that use the same index.

The raw view of the same entry from a previous version of SWISS-2DPAGE (before the new environment was applied on the database) is given for a light comparison (Table C.IV-3).

Table C.IV-3: SWISS-2DPAGE entry <P11604>, release 17.0 / March 2004.

```
ID   ALF_ECOLI; STANDARD; 2DG.
AC   P11604;
DT   01-SEP-1997 (Rel. 06, Created)
DT   01-OCT-2001 (Rel. 14, Last update)
DE   Fructose-bisphosphate aldolase class II (EC 4.1.2.13) (FBP aldolase).
GN   FBAA OR FBA OR FDA OR B2925.
OS   Escherichia coli.
OC   Bacteria; Proteobacteria; gamma subdivision; Enterobacteriaceae;
OC   Escherichia.
OX   NCBI_TaxID=562;
MT   ECOLI, ECOLI5-6.
IM   ECOLI, ECOLI5-6.
RN   [1]
RP   MAPPING ON GEL.
RX   MEDLINE=98410772; PubMed=9740056;
RA   Tonella L., Walsh B.J., Sanchez J.-C., Ou K., Wilkins M.R., Tyler M.,
RA   Frutiger S., Gooley A.A., Pescaru I., Appel R.D., Yan J.X., Bairoch A.,
RA   Hoogland C., Morch F.S., Hughes G.J., Williams K.L., Hochstrasser D.F.;
RT   "'98 Escherichia coli SWISS-2DPAGE database update.";
RL   Electrophoresis 19:1960-1971(1998).
RN   [2]
RP   MAPPING ON GEL.
RA   Tonella L., Hoogland C., Binz P.-A., Appel R.D., Hochstrasser D.F.,
RA   Sanchez J.-C.;
RT   "New perspectives in the Escherichia coli proteome investigation.";
RL   Proteomics 3:409-423(2001).
2D   -!- MASTER: ECOLI;
2D   -!-   PI/MW: SPOT 2D-000L0H=5.55/40651;
2D   -!-   PI/MW: SPOT 2D-000L1R=5.43/39776;
2D   -!-   AMINO ACID COMPOSITION: SPOT 2D-000L1R: B=10.9, Z=10.5, S=7.2,
2D         H=3, G=10.5, T=5.3, A=9.4, P=4.3, Y=3.6, R=3.2, V=7.4, M=1.7, I=5.3,
2D         L=8.6, F=4.3, K=4.8 ;
2D   -!-   MAPPING: AMINO ACID COMPOSITION AND SEQUENCE TAG (SKIF) [1].
2D   -!- MASTER: ECOLI5-6;
2D   -!-   PI/MW: SPOT 2D-001L5L=5.56/50220;
2D   -!-   PI/MW: SPOT 2D-001L6U=5.56/49421;
2D   -!-   PEPTIDE MASSES: SPOT 2D-001L5L: 1320.801; 1502.988; 1878.257;
2D         2591.649; 2719.736; 2871.916; TRYPSIN.
2D   -!-   PEPTIDE MASSES: SPOT 2D-001L6U: 934.591; 950.583; 953.573;
2D         1320.797; 1502.947; 1878.221; 2591.534; 2719.66; TRYPSIN.
2D   -!-   MAPPING: MASS FINGERPRINTING [2].
DR   Swiss-Prot; P11604; ALF_ECOLI.
//
```

Comment on the flat file entries

Whenever data is initially integrated from direct inclusion of such text records into the database without deep examination, many potential problems and source of errors can occur. Beside the likely syntax or semantics errors and potential redundancy, inconsistencies might arise unintentionally (*e.g.*, *MI* may list some images that do not appear in the *2D* section, or else several distinct spots may share the same exact position on a map). This is explained by the fact that entries are listed independently despite the common data they share together (*e.g.*, two distinct proteins may be identified on the same gel but may be wrongly annotated as belonging to two different organism!). Without an adapted management system, supervising all these issues becomes quickly a tedious task. The text view of a protein entry is certainly practical for human reading, but it can only hold a limited and finite part of the available and crucial data. It is clearly a view centred on the protein and cannot straightforwardly catch and represent by itself experimental details (by opposition to a spot centred or an experiment centred perspective). Quantitative measures of spot expression, experimental conditions, identification results and analysis either are absent or significantly undervalued. We have already mentioned the problem of experimental data representation in such a format, like when we want to associate a specific experiment to its interpretation. For example, in MS/MS analysis, several peptide sequences can be associated with each spectrum. To indicate precisely for each peptide sequence which spectrum and which of its peaks the peptide corresponds is not straightforward. There are apparently several manners of representing this association in a plain text file, none of them being satisfying. One manner is to append, in turn, all subsets of peak values and their corresponding peptide sequences to the entire peak list, at the cost of complicating the annotation syntax rules. A second manner is to create an independent block of data for each spectrum and to strictly follow a defined order to list the peaks and the peptide sequences within each block. A third manner is to assign local identifiers to the different spectra, and then refer to these identifiers when associating peptides with spectra. SWISS-2DPAGE uses similar local identifiers for the bibliographic references (the *RN* lines). Hence, one can point to the bibliographic references in the experimental annotation sections of an entry.

Despite these kind of representation problems, the use of a flat file format as source of data was (and remains somewhat) very popular among many federated 2-DE databases. This is because of its simplicity and its ease of generation, but also because of the promotion it has gotten from the former Make2ddb tool (Hoogland et al. 1997). A tool that was developed in 1997 by the SWISS-2DPAGE team and that helped many laboratories to publish their 2-DE data, either on the Web or for internal access. A transition into a more adapted format is indispensable. However, converting already existent datasets should not be done abruptly. For many practical and social reasons the conversion should be done gradually to ensure a better acceptance of changes. Spending too much time in revising their already published data in order to make it conform would discourage many data producers from adopting a new format. In addition, switching abruptly to a new format may throw end-users into confusion, having suddenly to deal with a new format.

C.IV.4 Similar federated SWISS-2DPAGE-Like databases

A number of 2-DE databases are fully or partially federated. Many of them have adopted SWISS-2DPAGE-like format and have used the initial Make2ddb tool. These databases are supplied with an engine to search identified proteins based on their accession number, identifier, description or related publication authors' names. All of them have map images where information can be displayed by clicking on identified spots. Many of such databases are grouped in the well-known **WORLD-2DPAGE** List. A list maintained by our team and accessible from the ExPASy server at:

- ❖ *http://world-2dpage.expasy.org/list/*

This list provides an up-to-date inventory covering the most important public databases. It currently lists up to 60 databases totalising nearly 400 gel images. Resources are grouped by species categories and tissues. They are also individually ranked from fully federated to non federated. Table C.IV-4 lists some of the most representative databases built with the former Make2ddb tool.

Table C.IV-4: Examples of federated 2-DE databases built with the former Make2ddb tool.

Database	Species	Institution
PHCI-2DPAGE[1]	Parasite Host Cell	Department of Medical Microbiology and Immunology, University of Aarhus, Denmark
SIENA-2DPAGE[2]	Multi-species	Department of Molecular Biology, University of Siena
COMPLUYEAST-2DPAGE[3]	Yeast	Complutense University, Madrid
OGP-WWW[4]	Human	Oxford GlycoProteomics Webpage, UK
PMM-2DPAGE[5]	Multi-species	Purkyne Military Medical Academy, Czech
ANU-2DPAGE[6]	Rice	The Australian National University, Canberra

C.IV.5 Other important federated 2-DE databases

Proteome Database System (2D-PAGE)[7]

The Proteome Database System for Microbial Research (Pleissner et al. 2004) is a major federated pathogenetic bacteria database developed at the Max Plank Institute for Infection Biology in Berlin. Data can be submitted to the centralised database, which contains identification data and offers comparative analysis by means of a graphical interface and statistical analysis. This database currently contains more than 34 highly annotated reference maps.

[1] http://www.gram.au.dk/2d/2d.htm

[2] http://www.bio-mol.unisi.it/

[3] http://babbage.csc.ucm.es/2d/2d.html

[4] http://proteomewww.glycob.ox.ac.uk/2d/2d.html

[5] http://www.pmma.pmfhk.cz/

[6] http://semele.anu.edu.au/2d/2d.html

[7] http://www.mpiib-berlin.mpg.de/2D-PAGE/

Yeast Proteome Map[1]

The Yeast Proteome Map (Perrot et al. 2007) is a 2-DE database of Saccharomyces cerevisiae developed at the “Institut de Biochimie et Génétique Cellulaires” in Bordeaux. It is a highly annotated species-specific database, with good quality identification evidence and a good dynamic gel viewer.

ECO2DBASE

The Escherichia coli gene-protein database has been integrated with its own loader into the Bio-SPICE warehouse system: a system that provides a common representation for diverse bioinformatics databases (Garvey et al. 2003). The most recent version, edition 6, is no longer publicly available. However, if an agreement is reached, there might be a possibility for the database to be hosted by the Proteomics Informatics Group at SIB, and made available from the ExPASy server using our Make2D-DB II tool.

TMIG-2DPAGE[2]

A Human age-related 2-DE database developed by the Tokyo Metropolitan Institute of Gerontology. This database provides identification results and many bibliographic references. It has been integrated into the recent LIPAGE management system, the open source Laboratory Information Management System for 2DPAGE-based proteomics workflow, developed at the same institute (Morisawa et al. 2006).

Rice Proteome Database[3]

The Rice Proteome Database is being developed at the National Institute of Agrobiological Sciences, in Japan. It contains information on proteins identified from many rice tissues and organelles. Identified entries are provided with identification evidence.

HSC[4]

Another federated 2-DE database of Human maps developed at the Heart Science Centre, Harefield Hospital. The database is federated and contains a basic search engine linked to clickable map images. Some few annotations on the identified proteins are provided.

Human 2D-PAGE databases[5]

A Human cancer database, developed at the Danish Centre for Human Genome Research. The database has extensive links to other resources and contains a large gallery of map references.

1 http://www.ibgc.u-bordeaux2.fr/YPM/

2 http://proteomeback.tmig.or.jp/2D/index.html

3 http://gene64.dna.affrc.go.jp/RPD/database_en.html

4 http://www.doc.ic.ac.uk/vip/hsc-2dpage/

5 http://proteomics.cancer.dk/

UPD[1]

The University of Alabama at Birmingham has developed a proteomics federated database (Hill, Kim 2003), which provides a repository for the storage and linkage of 2-DE data (mainly Human). Simple quantitative comparative studies using statistical tools are supplied.

C.IV.6 Not entirely federated databases

2DWG meta-database[2]

One of the earliest 2-DE gel search engines on the Web was the 2DWG meta-database (Lemkin 1997). It is a meta-database organised as a large electronic table containing information about data in other databases. It has become heavily outdated since then. 2DWG used to offer a possibility to visually compare users' gels with others' gels using the Flicker image tool (Lemkin, Thornwall 1999), a simple Web-based tool for image comparison.

PROTICdb[3]

PROTICdb (Ferry-Dumazet et al. 2005) is a project developed at the Bioinformatics Centre, University of Victor Segalen, Bordeaux. It is a Web-based application acting as a repository with high visual capabilities, and in which identified spots are internally interconnected. It is mainly designed to store and analyse projects of plant proteome analysis by 2-DE and MS techniques. Submitters can upload their annotated data into projects. Currently, there are two analysis projects, one for *Arabidopsis* and the other for *Brassica napus*. Users of the database can construct complex queries to search its content, and export of some data to third-party statistical analysis is possible. The management system is freely available and can be installed on any machine.

Other resources

The following resources are partially or not federated 2-DE databases:

DynaProt	DynaProt (Drews, Gorg 2005) is a recent online database of Lactococcus lactis that is developed at the Technical University of Munich. It includes a set of experimental and predicted properties.[4]

[1] http://www.uab.edu/proteinmenu

[2] http://www-lmmb.ncifcrf.gov/2dwgDB/

[3] http://cms.moulon.inra.fr/proticdb/Protic/home/

[4] http://www.wzw.tum.de/proteomik/lactis/

The Key Laboratory	The Key Laboratory of Cancer Proteomics of Chinese Ministry of Health is a repository for nasopharyngeal carcinoma Human cancer 2D/MS data. Images are clickable, but very few annotations are provided. The database has adopted a Xindice XML based storing system (Li et al. 2006).[1]
GelBank	Developed at the Argonne National Lab, University of Chicago (Babnigg, Giometti 2004). This is a multi-species collection of gels. 2-DE gels are deposited by registered users. The image database contains modelled gel patterns representing collections of images. Samples are well annotated, but identification evidence is not systematically provided. For visual comparison, several gels may be displayed one after the other, in what the developers call an "animation". Storage of retrieved objects (sequences, gel patterns and animations) can be carried out by users for later use through the so-called Bio-bag container. We experienced, however, technical problems using the Web interface, as many actions could not be performed.[2]
Human 2D-PAGE	The Human 2D-PAGE Database for Proteome Analysis and Disease is Developed by the Danish Centre for Translational Breast Cancer Research. The database contains a gel gallery on Human tumours, with a dynamic graphical viewer and several external links.[3]
BPP-2DPAGE	A 2D-PAGE database maintained by the U.F.R Leonard de Vinci, University of Paris 13. It lists maps of hematopoietic cell lines with cross-references to UniProtKB, but with no identification evidence.[4]
YMP	The Yeast Mitochondrial Proteome 2D Database, developed at the Department of Biochemistry, Oulu University, Finland. It provides clickable maps linked to a whole summary of the identified spot with little annotations.[5]
VIRTUAL2D	VIRTUAL2D (Medjahed et al. 2003) is not properly a 2-DE data resource, but rather a system to generate and visualise virtual two-dimensional gels from proteins deposited in databases based on their 1-D sequence. This approach may serve as a rough starting point in some proteome investigation, e.g., by narrowing a pH range prior to running 2-DE experiment or by predicting the approximate location of low-expressed unmodified proteins.[6]

[1] http://www.xyproteomics.org/xmldb/

[2] http://gelbank.anl.gov/2dgels/

[3] http://proteomics.cancer.dk/

[4] http://www-smbh.univ-paris13.fr/lbtp/biochemistry/biochimie/bque.htm

[5] http://www.biochem.oulu.fi/proteomics/ymp.html

[6] http://proteom.ncifcrf.gov

Other minor resources are available all over the Web. Minor resources are often single reference maps, reports or gel image captures with few annotations. Many of them cannot be easily exploited by machines, and hence can unfortunately not be of large benefit to the community without appropriate conversion.

A certain number of software tools help building and publishing 2-DE datasets, such as our Make2D-DB II Tool[1], the PROTICdb and the LIPAGE projects. Only 2-DE resources built with one of these tools (and to a lesser extent the Key Laboratory database) have the ability to exchange data with other external systems. Apart from manual interaction and/or full content distribution, the other datasets act like isolated resources lacking flexible capabilities to export data and to dynamically communicate with the outside world in a machine-readable manner.

C.IV.7 Many unavailable datasets

One of our main concerns and our initial motivations was the amount of 2-DE experimental analysis that is generated by the numerous laboratories working in protein identification and characterisation but which are not made available to the researchers' community. A large number of isolated 2-DE data are produced in so many different formats, ranging from highly structured formats to very plain text reports, and with dissimilar levels of annotations. Many of this data is either not available to the community, or is only published in the literature without being electronically accessible. Researchers are aware of how data is of limited use unless intelligent annotation exists in a functional form once presented and published (Hancock et al. 2002). There have always been attempts to share annotations transparently among multiple groups, but the sharing of information requires a general coordination either by keeping all annotations in a centralised open repository, forcing all parties to adhere to a common data representation format, or by requiring a vocabulary and semantic control.

C.IV.8 Semantic control and standards

The microarray community has already reached a major accomplishment in this direction with the MGED society, by defining and promoting MAGE, its own standards (B.I.2). Since the beginning of this century, a number of early attempts with different approaches to share annotations in proteomics have been supported by some bioinformatics solutions, like the rather generic Distributed Annotation System DAS (Dowell et al. 2001) or the Oxford Genome Anatomy Project OGAP, which acts as a biologic reasoning platform. It is not before 2002 that a major actor on proteomics, the Human Proteome Organisation, parented an ambitious project to standardise proteomics data: the **Proteomics Standards Initiative**, PSI, aiming to achieve standards in both data exchange and reporting (Orchard et al. 2003). Collaboration since 2005 with members of the MGED that are developing FuGE, the Functional Genomics Experiment modelling (Jones et al. 2006a), was another major milestone in recent PSI developments (Taylor et al. 2006; Hermjakob 2006; Jones, Gibson 2007). In

[1] No 2-DE database built with Make2D-DB II has been presented in this section.

2007, PSI achieved a significant progress by defining **MIAPE** (Minimum Information About a Proteomics Experiment) documents[1] (Taylor et al. 2007). MIAPE documents outline the minimum information to be reported to unambiguously interpret and reproduce a proteomics experiment.

C.V. Mass spectrometry

C.V.1 Towards a standardised storage and exchange formats

One of the PSI working groups is the PSI-MS group. It defines community data formats and controlled vocabulary terms facilitating data representation, exchange and archiving in the field of proteomics mass spectrometry. PSI-MS was the first PSI working group to achieve concrete and accepted proposals/recommendations. The **PSI mzData** format, intended for raw mass spectrometer output, is now reasonably stable. mzData is a data format capturing peak list information that aims to unite the large number of current formats into one. It is not a substitute for the raw file formats of the instrument vendors. At the same time, there was also another standard format put forward, the **mzXML** format, that has been promoted by the Seattle Proteome Center at the Institute for Systems Biology (ISB) (Pedrioli et al. 2004) and that has gained, since then, a good acceptance. Merging of mzData and mzXML, into one common format, **mzML** (formerly dataXML), is an on-going effort currently undertaken by PSI, with full participation of ISB. This will reinforce even better the acceptance and the adoption of a one unique standard format for mass spectrometry data among the proteomics community.

Analysis of data derived from proteomics experiments is being investigated at the moment by PSI and FuGE members. This collaboration group is expected to provide the functional genomics community with a standard format for representing results of analysing and processing experimental data. The **analysisXML** data exchange standard is the likely format to be yet delivered as a result of this collaboration.

C.V.2 Repositories

"*Much of our rich understanding of global gene expression patterns comes (...) from the fact that experimentalists deposited their raw data into the public domain (...) Proteomics has yet to see the many benefits gained by reanalysis of the (mass spectrometry) data by computational and statistical researchers...*" (Prince et al. 2004).

OPD[2]

As a gesture toward initiating a public repository for MS data, the previously mentioned authors (*cf.* B.II.4) launched the Open Proteomics Database (OPD). This database is a simple storage system for raw (and large size) files in different formats.

[1] http://www.psidev.info/index.php?q=node/91, http://www.nature.com/nbt/consult/index.html

[2] http://bioinformatics.icmb.utexas.edu/OPD/

Files may be downloaded 'as is'. In April 2007, the database contained roughly 3,000,000 spectra representing experiments on five different organisms.

PRIDE[1]

The PRIDE PRoteomics IDEntifications database (Jones et al. 2006b) is a centralised, standard compliant, public data repository for proteomics data centred on protein and peptide identifications as well as on the evidence supporting these identifications. Identifications are related to specific species, tissues and sub-cellular locations, and sometimes made under specific disease conditions. They are generally annotated with supporting mass spectra. Proteomics laboratories can submit their identifications and spectra to PRIDE to support their manuscript submissions to proteomics journals. Data is submitted in PRIDE XML format when identification data is included. Mass spectra without identification data are submitted in mzData format. Being a web application, submission, searching and data retrieval from PRIDE can all be performed using an Internet browser. The repository can be searched by experiment accession number, protein accession number, bibliographic reference and sample parameters. Data can be retrieved as machine-readable PRIDE or mzData (the latter being used for mass spectra without identifications), or as human-readable HTML. The database implementation, while presently centralised, is open-source code and can be mounted on various servers[2].

At the present moment, PRIDE contains more than 2000 separate experiments, comprising over 300'000 protein identifications, 1'400'000 peptide identifications, and approximately 400'000 spectra. PRIDE protein identifications can be queried through DAS, which provides details of the coordination of peptides. PRIDE aims to fully support the yet to emerge PSI analysisXML format for identification data analysis and aspires, in the long term, to provide an automated program for regular re-analysis of deposited mass spectra. Cooperation agreements to exchange data between PRIDE and other emerging repositories are currently in progress.

PeptideAtlas[3]

PeptideAtlas is a first step towards a full annotation and validation of eukaryotic genomes using experimentally observed protein-products. It is a collection of peptides identified in a large set of LC-MS/MS proteomics experiments originating from published and unpublished data (Desiere et al. 2006). All results of sequence searching have subsequently been processed statistically through PeptideProphet[4] to estimate a probability of correct identification for results and to remove false positives (Desiere et al. 2005); Thus, in April 2005, 3.3 million spectra has already been processed. Peptides have been mapped to EnsEMBL (B.I.1) and can be viewed as custom tracks on the EnsEMBL Genome Browser using DAS. A uniform analytical process ensures a meaningful comparison of data across different experiments, generated by different

[1] http://www.ebi.ac.uk/pride/

[2] http://sourceforge.net/projects/pride-proteome/

[3] http://www.peptideatlas.org/

[4] http://tools.proteomecenter.org/PeptideProphet.php

groups using different instruments (149 experimental datasets were available in April 2007). Raw data from these diverse experiments are made available in several formats and all data is loaded into SBEAMS[1]. SBEAMS is an html-based relational database that allows integration of disparate data types, such as ICAT and cDNA experiments, and was implemented to facilitate data management at the ISB.

The Global Proteome Machine[2]

The Global Proteome Machine (GPM) allows users to quickly compare their experimental results with the best results that have been previously observed by other scientists. It is based on a combination of data analysis servers storing publicly submitted identification data, a user interface, and a relational database that retrieve relevant information obtained from the public data analysis servers (Craig et al. 2004). Underlying schemas are designed to validate observed protein coverage and peptide fragmentation data. This resource was initially designed to assist validating MS spectra and protein coverage patterns, exploiting previous peptide assignments. Although somewhat restricted to minimal information, GPM can access a variety of peptide identifications performed in many different experimental conditions. In April 2007, the repository was comprising more than 700'000 distinct peptide identifications. Being open source, the system can be easily installed as an in-house data storage system, or it can be used as an add-on to an existing Laboratory Information Management System (LIMS).

C.VI. So many data resources… what is that good for?

It would have required a whole manuscript to list all the useful existing proteomics resources available for researchers in proteomics. A fairly exhaustive and up-to-date reference list, as previously indicated, is made available each year by the Journal of Nucleic Acid Research[3]. For some rapid access, many other exhaustive lists of links to proteomics resources can be found all over the Internet[4]. But how can we best benefit from all these resources?

Having access to so many resources may be confusing. Navigating in many different locations and processing query results from one resource before accessing another resource is extremely tedious. Gateways or Web portals[5] are partly a solution to simplify this process. There are a number of integrated data retrievals and analysis systems that may be helpful to access many resources all at once, and to appropriately process data using bioinformatics tools. These systems tend to combine original data at different levels and in different manners with some logics related to the nature of the queries. The refinement of such logics, which is a vast domain of investigation in

[1] http://www.sbeams.org/

[2] http://gpmdb.thegpm.org/

[3] http://www.oxfordjournals.org/nar/database/cap/

[4] http://bioinformatics.ubc.ca/resources/links_directory is a handy example

[5] A pathway to many distributed applications and sources made accessible from one single entry point.

Bioinformatics, varies enormously between one system and another. Entrez[1] (B.I.1), the "Global Query Cross-Database Search System" developed at NCBI acts as an integrated search and retrieval system providing access to many databases simultaneously from one single user interface. The databases' coverage ranges from the Entrez gene or protein sequence databases, to biomedical literature citations (PubMed) and taxonomy definitions of species. The EBI provides an equivalent approach, the Integr8 Web portal that accesses more related EBI databases (Kersey et al. 2005). Many other systems will be mentioned later, like Biozon (Birkland, Yona 2006a), which is a unified biological resource on DNA sequences, proteins, complexes, and cellular pathways based on a graph approach and that integrates data from many resources such as UniProtKB, PDB, GenBank, and KEGG.

This underlines the necessity of offering comprehensive and intelligent integration solutions. In Chapter D. many approaches and technologies related to data integration are discussed after examining how data is organised and made accessible.

[1] http://www.ncbi.nlm.nih.gov/gquery/gquery.fcgi

C h a p t e r **D**

CHAPTER D. DATA MANAGEMENT AND INTEGRATION

The transition from in vivo to in silico biology requires not only a good understanding of the nature of the data on hand, but also to deal with a huge amount of information coming from many different and heterogeneous data sources. Those sources are generally called *databases*, which is a term used to designate a large and organised entity of persistent data, associated with the appropriate tools to update and retrieve the data inner items. With the availability of numerous public databases, researchers have to face a large range of related problems and challenges in order to make the best possible use of the data. These problems are ranging from the diversity of semantics, of data formats to technologies and interfaces used to access the database content. However, the lack of technical and semantic homogeneity is not the only one difficulty to surmount. Problems of incomplete information, or worse, of incompatibilities and conflicts between data are often encountered. A part from these problems, social and political concerns also have to be taken into account.

D.I. The nature of biological data resources

Besides the huge amount of biological data, its most notable aspect is its diversity and variability. Biological data sets are complex and are organised in loose hierarchies, which is the consequence of our incomplete (and in worst cases inconsistent) understanding of life science and the relationships between its diverse sub-domains. When inspecting the many data resources available over the Internet, we rapidly notice the degree of autonomy and diversity of those resources when compared to each other. At a semantic level, conflicts may be spotted between the resources. Some of the data is even presented in a legacy system in which no structured information management system exists. As for the adopted structure, the data content is stored or presented in a whole variety of formats, ranging from plain text files, spreadsheets, HTML, XML or RDF (unstructured or semi-structured formats) to relational (structured), object oriented layout and binary files.

D.I.1 Managing data in bioinformatics

Data is at the centre of all researches in life science. It is of crucial importance to accurately capture, organise, annotate, maintain and retrieve data as well as to separate relevant information from irrelevant one. This is precisely what we designate by data management. An introductory book defines bioinformatics to be "the science of creating and managing biological databases to keep track of, and eventually simulate, the complexity of living organisms" (Buehler, Rashidi 2005).

Generally, we use the term *data source* or *data collection* to indicate a source of data that is not managed by a database management system (DBMS) and where no evident organised structure is present. When data is structured and concepts of modelling are apparent, we commonly use the term *database*.

For a simple use case, let us consider a collection of data covering the identified spots on a set of 2-DE gels. For each spot, we may define the following properties:

- A unique spot identifier
- A unique gel identifier
- A species from which the sample originates
- The position and the volume of the spot expressed in some relative measurement unit and its physical properties (pI and Mw)
- One or several identified proteins from a protein sequence database (*e.g.*, UniProtKB), each protein defined by a unique identifier
- Peptide sequences (from MS/MS analysis)
- …

Those properties, along with the relationships among them, constitute the database's *schema*. Properties are called *attributes*. They can be *single-valued* (*e.g.*, the gel identifier or the species) or *multi-valued* (the peptide sequences). The attributes can be either atomic (a simple type, like numeric values for pI and Mw in our example), or nested (a structured type, like "protein", which itself is a combination of an identifier and a knowledgebase name) (Eckman 2003). As for the schema, it must ensure data accuracy. Constraints, both at the logical or the biological levels, must be satisfied and must be reflected within the schema. A logical constraint, for example, would be for the relative volume not to exceed 100%. A biological constraint would also limit the alphabet used in the peptide sequences to the set of symbols representing the basic amino acids. Constraints may also be extended over several entries or *instances* (an instance being a single data record, *i.e.*, a given spot defined by its specific attributes). Two spots sharing the same gel identifier cannot originate from two different species.

This data set is dynamic, in the sense that some records might be added, modified, or deleted. The initial schema itself may be subject to many modifications. New attributes may be added or modified. We may decide, for example, to include a link to a mass spectrometry repository to validate the given peptide sequences, or to use a nested new attribute listing the related genes of the identified proteins and their chromosomal locations.

Another important task in managing data is to correctly handle *concurrency*. This consists, for example, in taking into account how to manage the activities of several users when they are curating the same entries simultaneously. A curator's change must be complete before a second curator's change can be applied. Conflictual situations must be considered with extreme care and inconsistencies should be correctly handled. For example, a curator may split a spot into two new distinct spots, while a second curator is submitting annotations regarding the initial spot. These annotations may be skipped or be applied to one or to both new spots.

A database is useful only if data can be extracted out of it. Queries should be expressed in some querying language that includes *search predicates*, the conditions that must be satisfied by the extracted data. Such languages generally include logical operators (*i.e.*, *OR* and *AND*) and some algebra operators (*e.g.*, *Union*, *Join…*). We may want, for example, to retrieve all the identified proteins with a pI of 7±0.5 that belong to a particular species. This query will be decomposed into distinct sub-queries. An efficient management system should optimise the execution *cost* (evaluated in time and resource usage) by adopting an adequate strategy when portioning this initial query. This is called a *cost-based query optimisation*. The strategy depends on many factors that include the quantity of data to be parsed, the time needed to access the database, the specificity of the search predicates, the existence of indexes, etc. In the earlier example, a strategy would be to first extract all the spots satisfying the pI condition and then, in a second step, check them one by one to only keep those originating from the appropriate species. If the adopted management system has a high cost in comparing numerical values, or if the first sub-query would deliver a huge number of spots in comparison to those belonging to the species, a better strategy would be to start by catching the spots satisfying the species condition before applying the pI condition over the retained spots. Another alternative is to decompose the initial query into two distinct sub-queries, one looking for all spots in the indicated pI range, and the other extracting all those belonging to the specified species. The result of the

Union of the two subsets is then delivered (the spots that are simultaneously present in the two retrieved subsets).

D.I.2 Data Structure

Data structures can be ranged into three levels: base, linear and non-linear data structures. Each level is associated with specific data types. For the base data structures, the data types are the primitive (atomic) types, *i.e.*, strings, integers, float, etc. The linear data structures are associated with one or multi-dimensional lists and associative arrays, while the non-linear data structures are formed by graphs and trees data types.

Unstructured data

The most chaotic data representation in proteomics would be a collection of information with no inherent structure, gathered in an indistinct order. Any particular portion of information may or may not be present and no apparent relations between data items exist. There is no possibility to formulate queries or to extract information. Such a type of data is purely unstructured and is essentially text oriented.

Unstructured data includes any text-based driven documents with meaningful information without following a precisely defined structure. Such documents are to be read by humans, since the lack of a definite structure makes them not adequate for automatic parsing by computers. Expressing dimensionality and complex data types (*e.g.*, nested attributes) in such documents is not an intuitive task. Most of those documents are simple text reports or loose information contained within HTML (Hyper Text Markup Language) pages for Web display. Indexing methods can be applied with unstructured data to retrieve some information.

Structured data

At the other extreme, data follows a formal and well-defined *data model*, which is the abstraction through which data is organised and viewed. The model defines the **relationships** between data items in a strict manner and guarantees completeness, in the sense that nothing defined in the data model is allowed to be missing. Whenever a data item is not provided it is given an *undefined* (absent) value. Nothing not defined in the data model can be present either. This is called **structured** data.

Structured data can be easily organised and remodelled. The **relational** model (based on predicate logic and set theory, *cf.* Appendix V.), the **hierarchical** model (a tree-like structure with nodes having a single parent and multiple children), and the **network** model (a lattice structure of nodes with multiple parents and children / many-to-many relationships) are all models of structured data[1]. Object-relational and object-oriented models also offer methods of structured data.

Spreadsheets

A spreadsheet is a single two-dimensional grid / table consisting of rows and columns of primitive types. Spreadsheets are very popular because of the availability of

[1] http://www.itouter.com/Database/encyclopedia.htm

applications that produce them and because of the simplicity to generate readable text reports using these applications. Although this format benefits from a bi-dimensional organisation, it can only define linear types. Nesting multi-valued attributes cannot therefore be expressed without the use of a well-defined format. No relationships between data in separate tables can be directly expressed. Spreadsheets are exported as text CSV or tab-delimited text files.

Semi-structured data

Many types of data are partially unstructured while following at the same time some of the characteristics of structured data models. Those types are commonly called **semi-structured** data types. They do not follow all the strict rules of a structured model and data content may be incomplete. Due to the fact that this category offers a compromise between the two extreme types and that, in practice, biologists do not wish to spend all their time and efforts using a strict and non-flexible data model approach, many types of data format used to store molecular biology data fall into this category. Besides, making use of semi-structured text formats in data exchange, in data publication, and particularly in data integration, is much more convenient, when compared to the abstraction of a fully structured relational presentation.

In a semi-structured presentation, data is organised in semantic entities where similar entities are grouped together. However, this presentation does not require similar entities to have the same attributes. Besides, the order of attributes is not necessarily important and not all the attributes may be required. The data types of the same attributes in a given group of entities may even differ. The information that is normally associated with a schema is contained within the data, and is commonly called self-describing. Semi-structured data models are built using labelled directed graphs rather than labelled trees that have a unique path for each leaf. Use of object identifiers is possible to refer to any node.

Semi-structured formats have the advantage of being easy to query, without entirely knowing the model or the data types. These formats can be either read by humans or parsed by computers. They offer a flexible solution to integrate heterogeneous data. They are widespread in molecular biology databases, where the need of structural and free text data is desired. However, optimisation and consistency are often hard to achieve.

Flat files

A flat file is a file that contains records or entries (*cf.* C.IV.3). Flat files have syntax-oriented formats. Records are listed sequentially and fields from each record are often delimited by specific characters. They are simple to read but allow computer parsing only if their structure is strictly defined and known. Dealing with flat files to extract data remains however complicated and unsafe, given the lack of constraints, the complexity to express non-linearity, and the difficulty to handle format changes. The simplicity of this format makes many major databases in proteomics continue to adopt it to distribute their contents, *e.g.*, UniProtKB (along with XML and RDF) and SWISS-2DPAGE.

Object Exchange Model

The Object Exchange Model (OEM) is a graph-based, self-describing object instance model, which provides the basis for object representation in a graph structure with objects either being atomic or complex. OEM can be an advantageous alternative to flat files, however, only few data representations in proteomics are exploiting this format, *e.g.*, the ANNODA tool (Prompramote, Chen 2005).

Extensible Markup Language (XML)

The Extensible Markup Language standard was first published by the W3C[1] in February 1998. It is a simple and flexible text format derived from SGML, the Standard Generalised Markup Language, which is a meta-language used to define markup languages for documents. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an important role in the exchange of data on the Web and between applications. XML has two basic mechanisms for declaring text structure: XML Document Type Declarations (DTD) and XML Schemas (XSD). XML DTDs are the original and most widely supported approach, but lack the full ability to define data types and data widths. XSDs, a well-formed schema defined in XML, offer these features, along with namespace and constraint definitions.

XML data representations are widely used in life science (Achard et al. 2001). The microarray community already reached a significant achievement in defining an XML-based data format for data production and exchange: MAGE-ML (Spellman et al. 2002). The international Taxonomic Databases Working Group (TDWG) proposes a data representation for taxonomic information based on XML (Kennedy et al. 2006). In proteomics, many works about the management and the storage of data using XML exist, such as AGML, a centralised approach to describe and store 2D-PAGE data (Stanislaus et al. 2005), Proteios, a repository for the assembly and the analysis of gel-based proteomics workflows (Levander et al. 2007), and PRIDE, a repository to store mass spectrometry data (B.II.4). More importantly, the inclusive Proteomics Standards Initiative developing standards for data exchange and reporting in proteomics delivers XML-based data representations (C.IV.8). The practicality of using XML-based formats in proteomics is examined in many reviews, *e.g.*, to define standards for 2D electrophoresis data representation (Ravichandran et al. 2004).

While XML is appropriate for data exchange, it is not an absolute solution or a substitute for good data representation. Problems of inconsistency and redundancy often arise. Incompatibility is unavoidable since XML representations can be used in different ways to encode the same information. Using directly XML-based formats (also called native XML) for data storage or computation in bioinformatics also engenders performance and scalability problems (Lin et al. 2005). Alternatively, some systems store data in a structure format (*e.g.*, relational databases) and provide an interface that allows users to view and query data in XML. These systems, called XML-enabled systems, overcome the traditional problems related to semi-structured formats. XML-enabled systems are more adapted for data-driven queries, while native XML systems are more efficient for document-driven (nested data) and navigational

[1] http://www.w3.org/XML/

queries (linked data). The type of system that is most appropriate depends therefore on the type of queries expected and the data being integrated.

Resource Description Framework (RDF)

The syntactic and document-driven XML cannot perfectly achieve the level of interoperability required by the highly dynamic and integrated bioinformatics applications (Wang et al. 2005b).

RDF[1], originally designed by W3C as a metadata model, has evolved into a general method of modeling information through a variety of syntax formats. The RDF metadata model is a knowledge-representation that can explicitly describe the data semantics. It is an abstract model that makes statements about resources using subject-predicate-object expressions (called triples). The subject indicates the resource, and the predicate indicates aspects of the resource and expresses a relationship between the subject and the object. RDF statements represent a labelled, directed pseudo-graph[2]. The mechanism is promoted by the W3C's Semantic Web[3] and is particularly convenient with life science data (Mukherjea 2005) in order to make software store, exchange, and use machine-readable information distributed throughout the web with better efficiency and confidence, and using shared ontology (Web Ontology Language / OWL[4]). RDF is mostly serialised in XML format, a representation that is distinguishable from the abstract RDF model.

The main advantage of RDF is its stability regarding structural changes (adding nodes and edges). RDF is an emerging technology in proteomics and in life science in general. Currently, few resources provide data using this technology. UniProtKB is probably the first resource to propose a distribution of its content using this format[5] (an ongoing project) along with the Gene Ontology project GO[6] (C.III). Recent developments include YeastHub, a prototype application to warehouse yeast genome data (Cheung et al. 2005), and AlzPharm[7], an RDF subset of BrainPharm[8] based on an associated ontology (Lam et al. 2007).

D.II. Three main approaches in designing data integration systems

The main characteristic of a data integration system is its ability to provide a uniform query interface from which one can access, in a reliable manner, a multitude of

[1] http://www.w3.org/RDF/ and http://www.w3.org/TR/rdf-concepts/

[2] A pseudo-graph is a graph that contains multiple edges and loops.

[3] http://www.w3.org/2001/sw/

[4] http://www.w3.org/TR/owl-guide/

[5] http://dev.isb-sib.ch/projects/uniprot-rdf/

[6] http://www.geneontology.org/GO.format.shtml#RDF-XML

[7] http://ontoweb.med.yale.edu/AlzPharm/

[8] A database (under development) to support research on drugs for the treatment of neurological disorders. http://senselab.med.yale.edu/BrainPharm/

heterogeneous data sources. The system must be able to perform queries and to react to changes in the underlying sources, whether those changes are updates to the schema or to the data content. Its design therefore implies to consider different sorts of management problems. Those problems are ranging from data warehouse design and modelling, maintenance of physical data independence, significance and evolution of the original data sets, to query optimisation and the generation of physical and logical views. The ultimate goal of a data integration system is to free the user from locating isolated sources relevant to some queries, interacting with each one individually and combining their results manually (Halevy 2001).

There are principally three approaches in designing a data integration system. The data warehouse approach[1], the mediator approach and the federated approach (Tatbul et al. 2001). The first two approaches are clearly distinct, while the third, the federated one, can be coupled with any of the other two.

D.II.1 The warehouse approach

The warehouse approach consists in physically collecting the data to be integrated from multiple sources. This can be done in two different ways: the *procedural* and the *declarative* manners. The simplest one, the procedural manner, consists in storing the sources without using any integrator schema. In an extreme situation where no structure is imposed, we talk of a *data repository*, which means importing and storing data as it is. As far as the declarative manner is concerned, a common format is used to translate and reconstruct the original data, and once it has been cleaned and reconciled, it is then physically stored into a unique system. The warehouse system contains then materialised views (*i.e.*, physically stored rearrangements) of the original data. This is considered a subject-oriented approach that, if well performed, offers the advantage of presenting already combined data to end-users, and to boost efficiency by reducing pre-processing time, but it has the disadvantage of being time-dependent and not spontaneously sensitive to updates on the original sources. Keeping the materialised views up-to-date is then a significant concern (Hull 1997). Planning scheduled incremental updates and properly managing local versions is a way to overcome this issue.

D.II.2 The mediator approach

In the mediator approach, a global schema and a set of sources compose a typical data integration system (Lenzerini et al. 2001). The real data is located at its original sources, while the global schema provides a reconciled and integrated view of the primary sources. This schema represents a set of relations associated with the considered domain and is not physically storing data “as is” in the system (Figure D.II-1). A variant of the schema, called *mediated schema*, differs from the global one by the fact that entities (the physical objects of interest, *e.g.*, a protein family, a 3D structure...) do not all need to be modelled, but only those that are shared within the system. This offers more flexibility in the sense that changes applied to the sources have a minimal impact on the mediated schema. Another advantage is that adding new items (*i.e.*, additional sources) becomes less arduous. Schematically, a mediated

[1] http://en.wikipedia.org/wiki/Data_warehouse

schema can be represented as a graph (or a network) where each node indicates some entity in the domain. Edges connecting the nodes are the relations between those entities (Mork et al. 2001).

D.II.3 The federated systems

A federated integration system is a collection of semi autonomous distributed databases, each of which having significant autonomy though providing access to all the other members of the system in a unified manner. Some authors do not consider federation to be truly a data integration approach (Hull 1997). The system is not centralised and it can be seen as an intermediate situation between no integration at all (where each resource has to be queried individually) and full integration (where all resources are queried by way of the integration system). The schemas of all resources are put together for the query. Mediator systems may be assimilated to *loosely* coupled versions (not sharing a unified schema) of federated systems (Hernandez, Kambhampati 2004). A *tightly* coupled federated system designates constituents that are sharing the same schema. In order to remain autonomous while respecting federation expectations, components of a federated system should share facilities to communicate in three ways:

- Data exchange: all components should be able to share data with each other

- Transaction sharing: for situations where a component does not allow full access to its data, but only to some operations performed on its content.

- Cooperative activities: each component should be able to perform complex queries involving access to data from other components

A federated approach largely provides significant autonomy for resources, allowing users to query them separately while all resources can still collaborate with each other to answer the query. A main inconvenience in such architecture is the need to map each source's schema to the others in a pair wise mapping. In a system of n components, and unless some algorithm is applied to manage the interconnection between the components in an optimal way, the interconnection coverage will require $n * (n - 1)$ schema translations. We may imagine an "intelligent" interconnection coordinator that would only necessitate components to be registered.

A federated system can be allied with the both already mentioned approaches. *Federated mediators*, for example, can be seen as part of a cooperation environment in which queries are addressed to a union of federated mediator databases (Cheng, Boudjlida 2005).

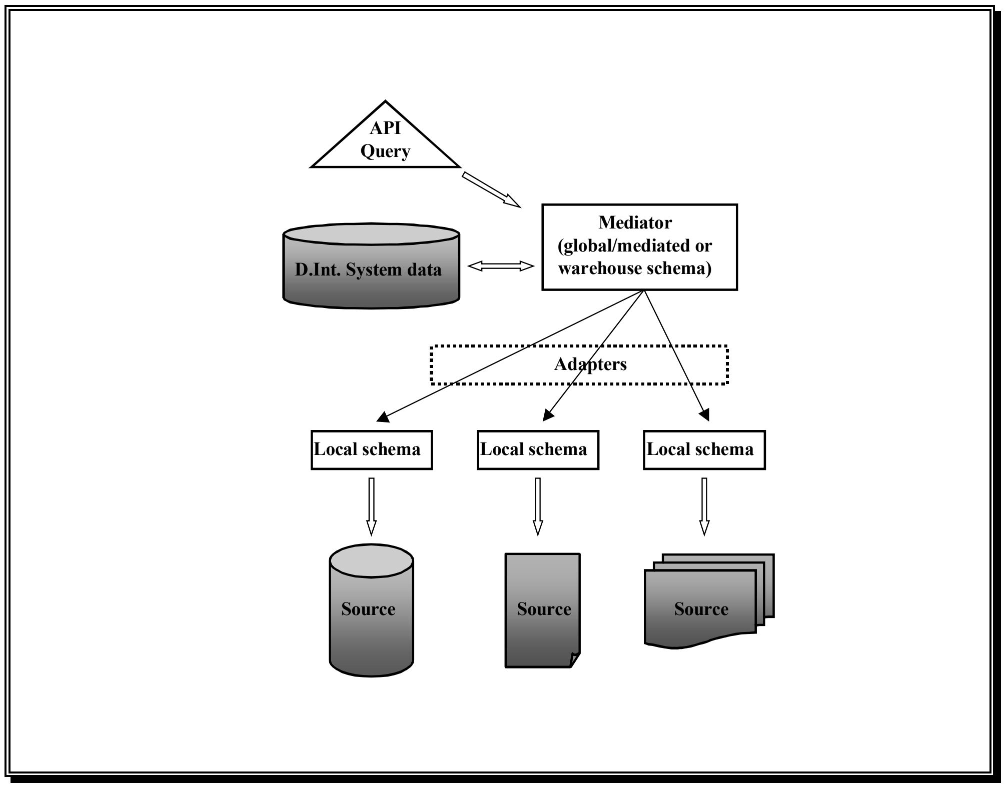


Figure D.II-1: Interaction between the integrator schema and the data sources.

D.II.4 What to consider

One primary aim of the integration task is to propose a suitable model reflecting the relation between the sources and the global (or mediated) schema in the mediator approach, or the materialised views in the warehouse approach. By wrapping data, we mean retrieving data from the sources and translating it into an integrator schema. This data is then, either stored in a warehouse system, or directly presented to the user in a mediator approach. Despite the relative simplicity of the figure (Figure D.II-1), one should keep in mind the many inconveniences suggested by this representation. In particular, the following issues need to be pointed out:

- Heterogeneity of the sources, their associated semantics and/or their time-varying schemas
- Technical limitations to access those sources
- The process of data extraction, of data translation, filtration and the way to bring those data together
- The choice of materialising data vs. a virtual integration (*i.e.*, data warehouse vs. mediators)
- A fair understanding of what users of the system need, balanced with what can be practically offered
- Building of a consistent model to cover the relations between the integrator schema and the sources, but also making this model reasonably flexible to handle potential changes (abstraction)
- Expressing and optimising answers to questions formulated in the global schema reasoning but with data originating from disparate sources, each having its own logic (the query plan)
- Tracking of unambiguously data provenance (Buneman et al. 2000) and the management of local versions
- The maintenance of updates related to the sources at both the content and the structure level
- The Performing of updates on the global schema considering the whole system's components and the evolution of users' demands
- Data exporting / exchange with other systems

The previous list, without pretending to be exhaustive, reflects the main challenges one has to face when conceptualising any data integration system; we will therefore portray the major aspects of each of those points throughout this document. The specificity of the domain covered by the system is by itself a major factor of consideration. Given the high complexity of data on hand in the life science areas, and more particularly in the purpose of our work in the proteomics fields, the problems dramatically expand by their amounts and granularities. As a result, there is obviously no "best" approach to adopt. Many issues can be addressed with different approaches, and one system suitable for a specific situation – with some specific goals - may be inappropriate or extremely difficult to apply to another one.

D.II.5 The query plan

Several works on answering queries using views were inspired by studies on the maintenance of physical data independence in relational and object-oriented databases.

A separation between the logical and the physical views of the data is characteristic of a data integration system (Halevy 2001). An interface is needed between the logical organisation of data and its physical distribution. In a mediated approach, where a conceptual query addressed to the system is expressed in terms of the global schema (source-independent), a mediator is responsible for reformulating the queries into a set of queries to the original sources (source-dependent). A *query plan* defines how to decompose the query into this set of sub-queries, to send the sub-queries to the sources, and to assemble (*i.e.*, filter, aggregate or merge) the results into the final answer.

Typically, the query plan in an integration system will imply a union of several queries over the sources. However, it is hard to always find an equivalent rewriting of the initial query, especially given the fact that views may often not be complete (*i.e.*, they may only contain a subset of their definition attributes). Three typical algorithms are presented in the Halevy review. The *bucket* algorithm, where the query posed on the mediated schema is reformulated into a query referring directly to the available data sources, the *inverse-rules* algorithm, where a set of rules that invert the view definitions is constructed, and the *MiniCon* algorithm, which takes into consideration how each variable of the query can interact with the views. The output of these algorithms is not exactly considered a query execution plan, but rather as a query referring to the view relations. The *completeness* of a query-rewriting algorithm expresses the ability of the algorithm to find a reformulation of the query if this reformulation exists.

Let us take as an example the TAMBIS (Transparent Access to Multiple Bioinformatics Information Sources) system architecture (Stevens et al. 2000). A single user interface lets the user perform his/her queries without a need to choose the relevant sources or the order in which to carry out sub-tasks (Figure D.II-2). The system uses *Concepts* based on basic biomolecular biology knowledge. New concepts are defined thanks to a model of knowledge (ontology-based model[1,2]) expressing the relations between the primary concepts. For example, two concepts *Motif* and *Protein* may be linked by a relation of the type '*is component of*' to form the new concept *Protein motif*. To perform a query involving a specific concept, the TAMBIS interface will ask the knowledge model about the parents, children and siblings of this concept, and about their valid relations to other concepts. A survey of knowledge-based information integration is given by Paton et al. (Paton et al. 2000).

In a warehouse approach, where the data is already combined and locally stored within the integration system, we still need to ensure that we are able to answer all the required queries using the materialised views. Reformulating a query may be necessary in order to optimise the processing time. Let us consider, for example, the case where a user wishes to extract all members of a protein family for some specific species, as well as for all the close descendants of this species. The user may define the protein family criteria and the species taxonomy identifier within his/her query. To optimise the execution process, the order in which the sub-queries are executed has its influence. Should we begin by extracting all the protein families corresponding to the given

[1] http://www.daml.org/ontologies/99

[2] http://www.cs.man.ac.uk/~horrocks/Ontologies/tambis.daml

criteria before filtering only those specific to the species of interest, or should we proceed in reverse? This situation is somewhat similar to what a *query execution plan* and an *optimiser* are responsible for in a RDBM system.

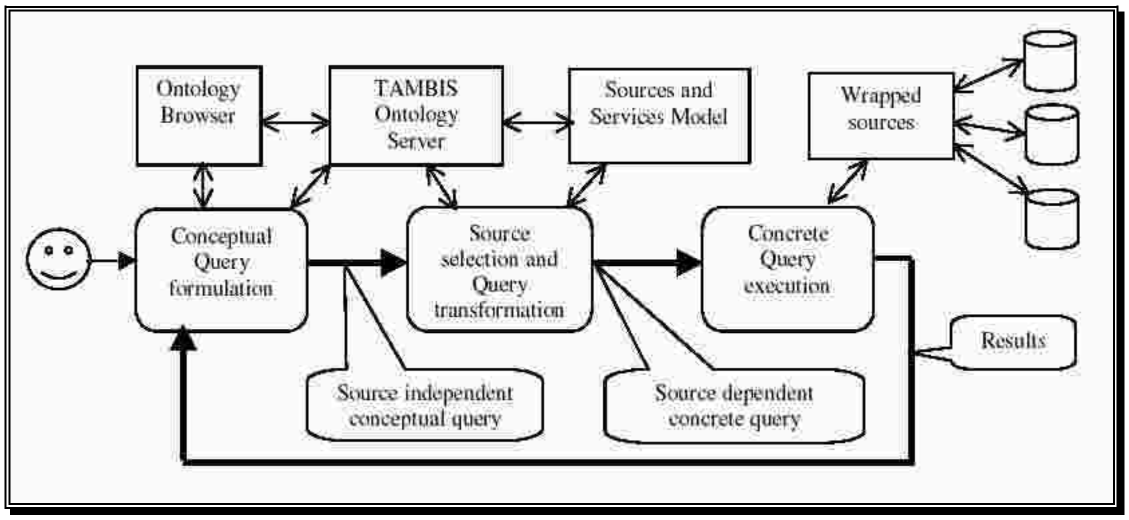

Ontology Browser
TAMBIS Ontology Server
Sources and Services Model
Wrapped sources
Conceptual Query formulation
Source selection and Query transformation
Concrete Query execution
Results
Source independent conceptual query
Source dependent concrete query

Figure D.II-2: The TAMBIS system architecture – processing a query into the query plan (Stevens et al. 2000).

The DiscoveryLink system (Haas, et al. 2001) provides an answer to cases similar to the above mentioned situation. DiscoveryLink is an integration system allowing users to query data, which may be physically located in many disparate sources, as if all the data was collocated in a single virtual database. It is based on a federated database system and built on technologies from the DB2 DataJoiner (Gupta, Lin 1994) and the Garlic research project (Haas et al. 1997). While the integration approach is a federated mediator-based approach, a database middleware engine containing the appropriate wrappers acts as a virtual RDBM system. A Data Definition Language (DDL) - based on the Structured Query Language SQL (Ullman, Widom 2001) – is used to register configuration meta-data in system catalogues. For each data source, selections of data will be exposed to the system as tables. To process a user query, the system will first evaluate the cost based on a *server attribute table* (SAT) that is defined for each data source. An *optimiser* is then invoked to set up the "best" query plan to be adopted, following a traditional dynamic programming approach similar to those used in conventional RDBM systems.

D.III. On the particularities of the data warehouse approach

As already presented, a data warehouse approach is based on a local centralisation of the data sources to be integrated. Most of the implementations are dominated by the relational database management system (RDBMS) and by adopting the high-level standards of SQL. The environment is structured, entirely controlled and benefits from high reliability and stability. On the one hand, network delays and timeouts are avoided at query execution time. On the other hand, the maintenance cost is extremely high given the likely dynamic, and sometime unpredictable, behaviour of biological sources. Moreover, data may not always be up-to-date.

D.IV. On the particularities of the mediator approach

D.IV.1 Modelling and Views in a mediator approach

There are two main approaches for the purpose of modelling the mapping in a mediator data integration system (Lenzerini 2001), the *global-as-view* (**GAV**, also called *global-schema-centric*) and the *local-as-view* (**LAV**, also called *source-centric*) approaches. The first one implies that the schema reflects, and should be expressed, in terms of the data sources. The second one requires a schema independent from the sources. In the latter case, relations between the schema and the sources are defined by considering each source to be a view within the global schema. There are, of course, many other possible types of relations in between those two approaches (**GLAV**). In all cases, and independently of the adopted mapping between the global schema and the sources, the integration system is required to answer queries in terms of the global schema. Users should not be concerned by the manner data is stored or collected from the different sources; instead, they should only be concerned by what we can call the *logic* of the global or the mediated schema. This *logic* represents a set of relations linked to the considered domain. The associated data does not need to be stored in the system.

D.IV.2 Some semantics

As already stated, a mediator system (I) has three main components: the global or the mediated schema (G), the sources (S) and the mapping (M) between G and S. We can formulate the system in terms of the triplet (G, S, M) (Lenzerini 2001). Queries to I are expressed in terms of G in some query language – with some expressive power - that we may call L_Q. This language defines- how data has to be extracted from the virtual database represented by the integration system. Both G and S are themselves respectively expressed in some specific language (L_G and L_S) over a corresponding alphabet (A_G and A_S). The symbols forming those two sets of alphabets are respectively the elements of G and S. A set of assertions constitutes the mapping between the global schema and the sources. Instinctively, $L_{M,G}$ and $L_{M,S}$ symbolise the query languages responsible for the mapping. Those definitions are general enough to describe theoretically any system integration approach.

D.IV.3 LAV, GAV and GLAV

The most important aspect when designing a mediator-based integration system is how correspondences between data from the sources and in the global/mediated schema are modelled. In the GAV approach, integrated data is organised as a view of all data sources. Queries are straightforwardly translated into sources queries, but the wrapping of a data source is a tough work, considering the fact that any change in the structure of the sources implicates the wrappers to be revised. Inversely, as the LAV approach defines each data source to be a view of the entire integrated schema, wrappers maintenance is much easier. However, the cost of query evaluation is higher.

In 2002, Lacroix published a proposal to deal with the heterogeneity and instability of semi-structured data and non structured documents retrieved from the Web, *e.g.*, HTML and flat files (Lacroix 2002). The method is a combination of the local and the global as-view approaches based on an intermediate object view mechanism called *search views*. The presented object Web wrapper (OWW), based on the Object Protocol Model – *OPM* - (Chen, Markowitz 1995) uses a view mechanism where data sources capabilities are listed by mean of *attributes* and tools used to parse the retrieved documents. Retrieved data is then cached in XML format for information extraction. This consists mainly in identifying the access and attributes for each Web data source, designing a search view corresponding to this access, writing a parser and registering it in the search view, and finally designing the user view by collecting information defined in the search view and extending it to all extractable information.

D.V. Comments on the three different integrative approaches

Warehousing emphasises data translation, as opposed to query translation in the mediator approach (Sujansky 2002). As indicated by its name, data translation implies to translate data from native sources into a common shared format and to address them uniformly by the same query executer. Maintenance cost is very high, especially when the underlying databases evolve with time. Translating a query into an equivalent set of local queries is the aim of query translation. This is more difficult to implement, and is not feasible at all, if no adequate interface to receive remote queries is attached to the data sources. In some approaches, query translations are based on procedural mappings. This kind of system is a hybrid between data and query translation. Procedural functions import and translate data into a shared format only when invoked by a query requiring access to this data. Kleisli, a mediator-based system, has adopted this approach (Chung, Wong 1999).

Data integration applications have to handle partial information, no matter what the modelling approach is. In all approaches, answering queries is an inference process that has to cope frequently with incomplete information. Reasoning is needed to answer queries due to the constraints usually applied to the global schema (Lenzerini et al. 2001).

In some conventions, the components called mediators are not “purely true” mediators (Ullman 1997). Given the fact that a mediator generates views, the warehouse approach, which is founded on materialised views, may be somewhat considered as a system using mediators indirectly to collect and combine data from the sources.

The three integration approaches can conceptually coexist and collaborate within the same system, with different levels of implication. In Make2D-DB II, an autonomous database, a mainly warehouse-based resource, is part of a network of federated databases. Nonetheless, those remote database systems are not identical because their versions and their individual schemas can differ (and are, therefore, not tightly coupled). Query translations, which are classically close to a mediator-approach, are then carried out to ensure data exchange between individual systems.

Detecting and characterising changes on data sources in the warehouse approach

Detecting changes on data sources is an important issue in the warehouse approach. A *push* or a *pull* technology can be used. In a push technology, users register queries with the primary data sources and request explicit notification when changes occur that match their queries. In a pull technology, users periodically check the underlying data sources to see if a change of interest has occurred. In order to perform a push action, data sources must be capable of processing and storing user queries, as well as generating and sending back electronic notifications. Currently, only very few data sources / datasets offer such services, *e.g.*, the Swiss-Shop service of UniProtKB/Swiss-Prot, which is a minimal service notifying users by e-mail when new entries satisfying a set of criteria have been created. A more advanced push service is the RSS (Really Simple Syndication) service, which is an automated technology to push data into special programs or filtered displays. UniProtKB is currently in the process of integrating a RSS service.

Another aspect in characterising changes is to track exactly how and when the underlying data source has changed, in order to distinctively propagate these changes to users of the integrative system. This is complex since updates are characterised in many ways: by relying on periodic global data source versions, by time stamping data entries and records, or by maintaining a list of changes and additions in a separate downloadable inventory. Many data sources / datasets offer more than one of these alternatives, *e.g.*, UniProtKB and the Make2D-DB II environment.

D.VI. Examples of data management and integration systems

A large variety of data integration systems in life science has been developed over the last decade (Wong 2002; Lacroix, Critchlow 2003a; Hernandez, Kambhampati 2004; Garcia et al. 2005; Lisacek et al. 2006a). The systems described in these reviews adopt diverse combinations of integration approaches. They vary significantly by their purposes, their scopes, their architectures, and their practicalities.

Table D.VI-1 and Table D.VI-2 display a compilation of different data integration systems. The reason for listing so many systems with their characteristics is not to evaluate them, but rather to illustrate and to compare different approaches in data integration and data management. To adopt any of these systems, a research group must consider many various factors. These factors include, among many others, the nature of data to deal with, the purpose of the research, the flexibility of the integration approach, the type of resources to be integrated, their availability and their steadiness. The factors also include practical considerations, like available computer resources, expertise with the used technologies, the purpose and the lifetime of the project, etc. It is only by having a global overview of the different possibilities on hand that a research group can have a suitable choice adapted to its needs.

Table D.VI-1 displays a non-exhaustive list featuring the elementary aspects of some representative general integration systems:

- SRS (Etzold, Argos 1993; Etzold et al. 2003)
- Entrez (B.I.1)
- EnsEMBL (B.I.1)
- Atlas[1] (Shah et al. 2005)
- DiscoveryLink (Haas, et al. 2001) and OPM (D.IV.3)
- K2/Kleisli and the Genomics Unified Schema / GUS (Davidson, et al. 2006)
- TAMBIS (Baker et al. 1999) (Figure D.II-2)
- Integr8 (Pruess et al. 2005) / BioMart (Durinck et al. 2005)
- Biozon (Birkland, Yona 2006b)

More specific gel-based integration systems are also listed in Table D.VI-2. Some of these systems are mainly management systems with almost no integration characteristics as defined earlier:

- Proteome Database System - 2D-PAGE[2] (C.IV.5)
- PROTICdb[3] (C.IV.6)
- ProteomeWeb[4] (Babnigg, Giometti 2003)
- PARIS[5] (Wang et al. 2005a)
- 2D/MS repository of NPC proteome[6] (Li et al. 2006)
- Make2D-DB II[7]

[1] http://bioinformatics.ubc.ca/atlas/

[2] http://www.mpiib-berlin.mpg.de/2D-PAGE/

[3] http://cms.moulon.inra.fr/proticdb/Protic/home/

[4] http://proteomeweb.anl.gov/

[5] http://w3.jouy.inra.fr/unites/miaj/public/imaste/paris/

[6] http://www.xyproteomics.org/xmldb/

[7] http://world-2dpage.expasy.org/make2ddb/

Table D.VI-1: A non-exhaustive list of general data integration systems.

	System particularity	Overall integration approach	Integration of new data, data update	Data model — Main query language	Model extension cost	Data exchange	Semantic approach	End-user expertise — Interactivity medium	Licensing / distribution
SRS	Retrieval system from indexed flat file, XML and relational databases	Classification of locally imported data	Data must be imported, no automatic updates	Linked text records and hierarchical — Icarus / logical statements	Low	Low	No	Low — API	Commercial[1]
Entrez	Portal for data from a set of pre-defined databases	Classification of data, navigation	Distributed, delayed updates	Linked text records — Web services and ANS.1[2]	Low	High	Medium	Low — API	Not distributed
EnsEMBL	Analysis tools on collected data	Warehouse	Central, automatic	Structured: relational — SQL	High	Low	N/A	Low — API and programmatic interface	Free
Discovery Link (and OPM)	Virtual Integration of data from a set of pre-defined / registered databases	Federated mediator (mediator)	Distributed, on the fly	Structured: relational (object layer) — Wrappers SQL / DDL	High	High	N/A	High — Programmatic interface	Commercial
K2/Kleisli	Distributed query system	Mediator	Distributed, on the fly	Semi-structured: ML (Kleisli) object-oriented (K2) — OQL[3] and CPL[1]	Medium	High	High	High — Text-based and RMI[2] client + GUI	N/A[3]

[1] SRS imposes some constraining conditions for free academic use.

[2] Abstract Syntax Notation One (ASN.1) is a formal language to abstractly describe messages to be exchanged among many applications involving networks and Internet.

[3] The Object Query Language (OQL) is a query language standard for object-oriented databases modelled after SQL.

GUS	A unified schema of several genomics databases and analysis results for data annotation	Warehouse	Central, delayed updates	Structured: relational — SQL	High	N/A	High	Low — API	Free
TAMBIS	Ontology-based query formulation integration system with high transparency	Mediator	Distributed, on the fly	Structured: object-relational — CPL, Conceptual Query Formulation	Medium (mapping of related ontologies)	N/A	High	Medium — GUI	Free
Atlas	Integration of data from a set of pre-defined databases	Warehouse	Data must be imported (dumps), no automatic updates	Structured: relational — SQL	High	Low	High	Low — API	Free
Integr8 / BioMart	Integration of locally installed databases for data mining and analysis	Warehouse	Central, delayed updates	Structured: object-relational — SQL	Low	Low	N/A	Low — API	Free
Biozon	Integration based on a graph-based approach and analysis tools	Warehouse	Central, delayed updates	Structured: hierarchical / relational — SQL / Query graph	High	Low	High	Low — API	N/A

[1] The Collection Programming Language (CPL) is now replaced by the more accessible OQL in K2.

[2] Remote Management Interface.

[3] K2/Kleisil has been imported to form the basis for the TAMBIS system.

Table D.VI-2: A non-exhaustive list of gel-specific data management and integration systems.

	System particularity	Overall integration approach	Integration of new data, data update	Data model — Main query language	Model extension cost	Data exchange	Semantic approach	End-user expertise — Interactivity medium	Licensing / distribution
Proteome Database System	Collection of 2D-PAGE datasets, with data analysis	No integration approach, Locally imported data / Repository	Data must be imported, no automatic updates	Structured: relational — SQL	High	N/A	Low	Low — API	N/A
PROTICdb	Local management system improved by some integration features	Warehouse	Central, delayed updates	Structured: relational — SQL	High	Low	Medium (CV)	Low — API	Free
Proteome Web	Local management system improved by manually integrated annotations	Warehouse	Data must be imported, no automatic updates	Structured: relational — SQL	N/A	Low	Low	Low — API	Not distributed
PARIS	Collaborative grid for 2-DE and MS data with analysis and data mining tools	Federated mediators	Distributed, on the fly	Structured: relational — SQL and Web Services	High	High	High	Medium — GUI / RMI clients	Free
2D/MS of NPC	Local repository based on XML storage, with no integration features	Repository	Data must be imported, no automatic updates	Semi-structured: XML — PHP	Low	High	Low	Low — API	Free
Make2D-DB II	Federated environment with distributed data accessible from any node (via transaction sharing)	Federated mediators + Partial Warehouse (LAV)	Distributed, on the fly updates (some data updates are periodic)	Structured: object-relational — SQL and REST	High	High	Medium (CV)	Low — API	Free

Some additional systems

Many systems were announced but could not be tested due to unavailability or to technical access problems, *e.g.*, ProDB and BRIGEP[1] and the CEBS SysBio-OM[2].

The ProDB system has been initiated in 2003 by the Center for Biotechnology at Bielefield University. Its goal is to store all relevant information about a proteome experiment and to allow simultaneous high-throughput analysis and annotations of mass spectra (Wilke et al. 2003). Detailed experimental parameters and acquired mass spectra are to be captured and stored within a SQL database. ProDB should feature an off-line/batch mode for the simultaneous analysis of mass spectra for an arbitrary number of different pre-configured sets of parameters. It intends to offer an architecture that supports the plugging-in of data-loading and analysis tools. An integration of genome and transcriptome data is also announced within a system called BRIGEP[3] (Goesmann et al. 2005), based on the BRIDGE integrative platform (Goesmann et al. 2003). BRIGEP uses ProDB as one component, along with GenDB and EMMA, the other genomics and transcriptomics plugins. Each component exhibits full-featured analysis software for its area, ranging from raw data processing to diverse functions to analyse the processed data. However, the ProDB and the integrative project have not been made publicly available since their announcements in 2003.

The CEBS SysBio-OM (Xirasagar et al. 2004) provides a comprehensive and flexible framework to store and integrate data generated from transcriptomics, proteomics and metabolomics experiments performed on the same biological samples. The model reflects the on-going standards of each of these underlying fields. SysBio-OM is an object-oriented model, which can be implemented on various computing platforms using an object-relational approach. It supports different ontologies and guarantees a high interoperability with other systems. However, implementing the model requires a high technical expertise. Although an open source distribution of the package was announced in 2004, a download was still not possible by the time of the writing of this work. Only a local data warehouse is currently available.

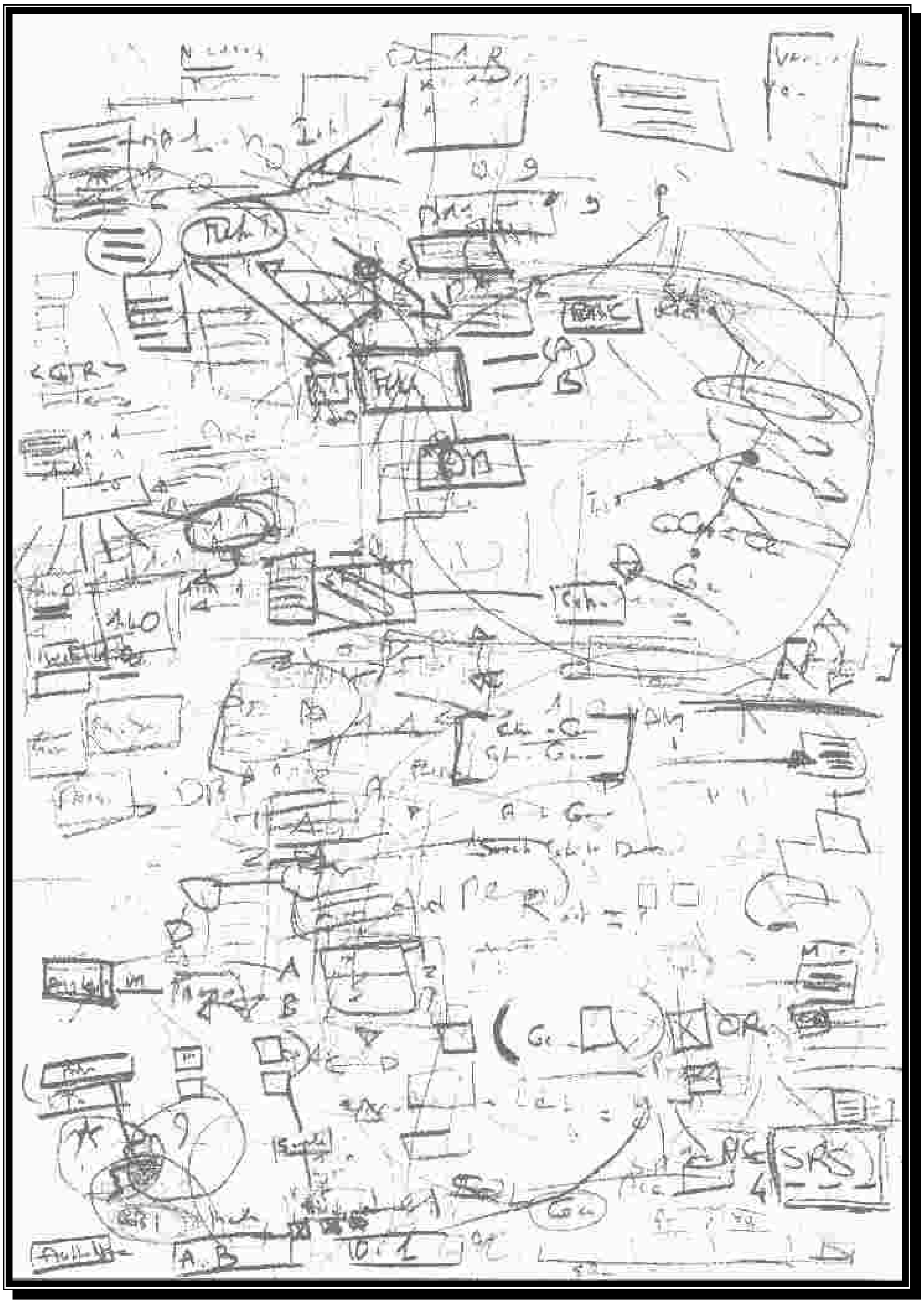
Interoperability of diverse integration systems

Despite the fact that so many integration systems coexist, it is often hard to achieve simple data exchange operations between two different systems. It has therefore become imperative to reinforce interoperability between the diverse systems. Data standardisation efforts and initiatives in the life science fields have already started to play an important role to resolve this issue by defining the appropriate data models, formats and semantics in data exchange. However, collaboration between the different communities is to be reinforced towards a more homogenised systems biology representation, since many challenges are still to overcome in order to reach accurate and long-term ontologies essential to support such common standards (Soldatova, King 2005; Brooksbank, Quackenbush 2006).

[1] http://www.cebitec.uni-bielefeld.de/groups/brf/software/prodb_info/

[2] http://cebs.niehs.nih.gov/cebs-browser/cebsHome.do

[3] https://www.cebitec.uni-bielefeld.de/groups/brf/software/brigep/cgi-bin/bridge.cgi



**Data integration in proteomics: a three hours illustrated discussion with a Swiss-Prot colleague on how to implement a practical distributed data integration system at SIB.*

C h a p t e r E

CHAPTER E. THE MAKE2D-DB II ENVIRONMENT - THE CONCEPTS

Managing and publishing SWISS-2DPAGE exhaustively and efficiently was the challenge we took up when we started to conceive this project. It quickly became evident that there was a great opportunity to provide other researchers, who are producing similar 2-DE data, with the necessary tools to also manage and publish their data. The Make2D-DB II environment was born.

E.I. Introduction

Make2D-DB II emerged from our initial aspiration to convert SWISS-2DPAGE into a new consistent and structured format. The flat file based database (Hoogland et al. 1999) had to be remodelled into a more extensible data representation and to be controlled by a more reliable management system. We have therefore initially focused our attention in building a new data model that was consistent, fairly flexible and extensible enough, but that was also able to faithfully reflect the initial database structure. At the same time, we had to build the required basic components that were needed to physically implement this data model, especially to deeply check and analyse the existing 2-DE data and to correctly convert it into the new representation. We also had to separately develop the requisite interfaces to access the data. Although these new interfaces were adding new functionalities, and were designed differently from the existing SWISS-2DPAGE interfaces, we chose to keep a similar look and feel in order not to confuse SWISS-2DPAGE users.

At the same time, it quickly became obvious that there was a great opportunity to provide other researchers, who are working with similar 2-DE data, with the necessary tools to also manage and publish their data. The fact that the same management system would be shared between remote databases directed the development of the project towards the conception of a large-scale environment. A federated environment in which resources are distributed while still being able to interact and exchange data using a transaction sharing approach (D.II.3).

❖ ***http://world-2dpage.expasy.org/make2ddb/***

E.I.1 Databases and data models

A database is a structured collection of related data with some inherent meaning. It represents some aspect of the real world, and it is designed, built and populated with data for a specific purpose. It can be generated and maintained either manually or with the help of software, depending on its size, complexity and data distribution. The software used to create, maintain and query a database is called a Database Management System (DBMS). The more complex a database is, the more considerable the amount of software to manipulate it.

A data model describes the structure of a specific database. It is represented by a collection of concepts that depend on the underlying database model. Implementing physically a database should always reflect and satisfy the model associated with it.

E.I.2 The EBP project: interconnecting remote databases

In 1999, our group was implicated in a European Union project (European Proteome Database of Pathogenic Bacteria[1]) implying 2-DE proteome analysis of various bacteria by several laboratories. Our role was to propose a means of making the data produced by each laboratory available to the other concerned laboratories. We opted against a centralised warehouse approach, in which data would have been collected and made available by a single group. We wanted data to be under its producers' control throughout the project existence. At the same time, in a federated approach, the provided tool had to be easy to install, as it would be mainly used by scientific groups that had no or little computer science expertise. Finally, the proposed solution had to be based only on public domain components. The data analyser and the data model had also to be reasonably flexible, as we inevitably had to deal with a large variety of data semantics generated by the many laboratories involved.

Having decided not to centralise the data, we also needed an efficient way to interconnect the remote resources using simple interoperability protocols that require no particular skills to operate. Another feature that we considered important was to ensure that each laboratory would be able to publish distinct sub-datasets contained within its own project. This led us to develop an interface capable of managing concurrently several separate local databases (sub-datasets). This same interface was also designed to simultaneously connect to any similar interfaces, in order to send them queries and to receive and consolidate the returned results. This was our vision of a federated database with distributed data under the control of its owners: a global data resource accessible from various entry points.

E.I.3 Further developments

An integrative virtual 2-DE database

When the interconnection procedures were implemented, it became obvious that there was an opportunity to extend the interoperability between the nodes of the system, to strengthen its reliability and to integrate other significant resources. We started to enhance the integration of non 2-DE resources, and we reinforced the mutual awareness between the 2-DE nodes. By means of common indexes, like the Swiss-Prot index, the NCBI taxonomy code, the EC Enzyme number, etc., each node could link to the other nodes, and even update its own inner data relative to the other nodes' content. The common indexes also allowed the system to gather relevant external information and to ensure that this information remained up-to-date.

Consequently, the implementation of the data model and the interfaces was adapted and reinforced by a collection of procedures to ensure an efficient interoperability and an accurate management of the expanded system. At the same time, more stable versions of the tool were made publicly available on the ExPASy server.

[1] The European Union EBPnetwork (QLRT-1999-31536)

Model extension and 2-DE Portals

The data model, initially centred on SWISS-2DPAGE-like 2-DE data, has been progressively enhanced and some of its elements have been made more generic. Users could express their own annotations and their specific identification methods. External documents, such as the various standardised PSI documents and/or user personal documents could be included or pointed at from the database implementation. The core data model was extended to cover proteomics analyses workflows more exhaustively, from project definition and sample description to 2-DE separation and identification methods.

Along with the model extension, the Web interface underwent additional developments that made it also able to act as a fully independent Web portal to access any number of remote 2-DE databases built with the same tool.

E.II. Objectives, constraints and initial choices

The aim of the Make2D-DB II tool is to provide a flexible and easy-to-use distributed environment that creates, converts, publishes, maintains and interconnects 2-DE datasets. It is a tool based on a relational database management system and that focuses on the interoperability between its components. To achieve our objectives we had to make some initial choices and consider some requirements:

- **A federated approach**: The data produced should always remain the producer's property and should stay under his/her control. This corresponds to our vision of a rather federated system and federated mediators with the data being distributed and managed by its producers: a global data resource that should be independently accessible from various entry points.

- **Interconnection and interoperability**: As we have decided to avoid a centralised approach and to virtually group the remote databases into a global one, a particular interface has to be set up. A common shared protocol will be used for the communication between remote systems. This protocol should hold some level of abstraction to ensure that in the long run consecutive versions of the interface will still be able to accurately communicate between each other, even when the database inner structure may undergo significant modifications.

- **Minimal redundancy**: The relational data management system offers a fully structured data organisation. Coupled with an appropriate data conversion, the RDBMS ensures the removal of redundancy, which is highly characteristic of low structured data formats.

- **Data conversion**: A data conversion process is necessary. It should be able to read and analyse data from a variety of text formats, *e.g.*, from flat files, CSV reports or XML exports.

- **Data consistency**: The conversion process should ensure the converted data is correctly imported into the structured format, *i.e.*, the relational database implementation. To achieve this goal, data consistency is essential. The syntax, the nature and the content of the data must be thoroughly checked. Whenever some inconsistency is detected, and if it is possible, suggestions for error correction should be proposed.

- **Semantics flexibility**: As we are dealing with data originating from various and independent groups, 2-DE annotations are inevitably heterogeneous. So far, there is no common consensus adopted by the proteomics community with regard to data syntax and semantics, especially by the time the project has started. We, of course, have to make some specific choices for data input. In our case, these choices are naturally linked to SWISS-2DPAGE, as managing this database is one of the main motivations of our project. Nevertheless, we should ensure minimum work on data adaptation for users that would otherwise dissuade them from using the tool.

- **Model flexibility**: The relational schema should be strongly adaptable for future evolution in protein identification and annotations, and to answer specific needs of different laboratories. Remodelling of the core schema must be easily realisable whenever necessary. This also implies that some parts of the model will be set up for future perspectives and expansion of the model, but will not be necessarily activated in the early implementations. For that reason, we need a database implementation with some object-oriented characteristics and with extended functionalities in creating and abstracting methods. PostsgreSQL, the public domain object-relational database management system, seems to be a suitable choice[1].

- **An entirely free of charge package**: Our tool is predominately destined to academic groups that may not wish, or may not have the needed financial resources, to buy costly software or to hire specialised computer science people to perform technical installations. Thus, some software components options, such as ORACLE RDBMS or CORBA[2], are to be excluded. All the environment components chosen are entirely free of charge and easy-to-install.

- **Ease of use**: Data input formats, the configuration process, and the installation procedure are all simple to handle. Managing several databases and connecting to remote resources must be easily performed. The use of interfaces to query or manage the data should be intuitive.

- **Automatic and transparent data integration**: All operations related to the integration of external data sources should be performed in a transparent manner, with little intervention from the user, except if a critical decision is to be made. The user should however decide when to do it, what kind of data to integrate, and what should be the extent of data replacement to perform.

[1] Details on PostgreSQL are given in Appendix V. (Relational databases)

[2] http://www.omg.org/gettingstarted/corbafaq.htm, *cf.* glossary.

- **Public data and internal data**: We opted for a unique installation in which both public and internal datasets are clearly separated. The database maintainer will then decide when to turn the entire internal data into public. Besides, we will provide an interactive mode to restrict the visibility of some particular data subsets to privileged users.

- **Access to experimental databases, data repositories and external text documents**: Pointers to access experimental databases, such as mass spectrometry and peptide fragments repositories, should be integrated within the environment. Pointers should also direct to local text files or to external Web pages when necessary, for example to access protocol documents. Use of such pointers should closely follow the ongoing efforts in proteomics standardisation for data exchange and publication. Provided that the structure of the accessed file or the retrieved data is known, projections of parts of the data can be physically imported into the main 2-DE database for quick visualisation and first-step search queries. This can be applied, for example, to extract mass spectrometry peak lists, identification confidence values, people to contact, etc.

- **Object designation and extraction**: We want to provide an intuitive way to designate and to extract resources or "objects" using simple Web protocols. An object can be identified over a network by its Uniform Resource Identifier (**URI**) using a specific descriptive syntax called **logical URLs** (as opposed to physical URLs)[1]. Objects in our case are principally proteins, gels, spots and related identification experiments. When pointing to an object, we may also want to specify the format in which to view or extract this object. We also want to extend this concept, using an analogous syntax, to express queries, which would return a list of objects satisfying the input parameters. One of the obvious advantages of this concept is to simplify the interoperability between remote systems and to reinforce the automatic and up-to-date integration of resources.

- **Database distribution**: The database content needs to be exported to guarantee the reconstruction of the database, when necessary, on the same support, as well as for use in alternative locations. Only data that is proper and specific to the 2-DE database are to be exported, as the related external data can be rebuilt by connecting to the same external resources. Exports to rebuild the database are carried out in a SWISS-2DPAGE-like flat file augmented with some supplementary annotations to express additional data structures that cannot be plainly expressed in a flat file. XML exports should be more convenient for generic distributions. An XML distribution, based on a gel/spot perspective, as opposed to a protein perspective in a flat file, is therefore to be considered. However, exports in XML format will be delayed until stable data exchange recommendations from PSI for sample preparation, gel protocols and experimental identification and analysis are validated, so as to avoid the coexistence of potentially diverging formats, and to be fully PSI compliant.

[1] *cf.* http://www.merges.net/theory/20010305.html

- **Backup procedures**: To avoid accidental losses of data, we choose to keep track of any erased or modified record within the database. It will thus be possible to recover the content of the database at any specific date. Even if such operation requires some level of expertise, it guarantees at least that no data will be definitively lost. We will also provide an easy way to dump the 2-DE database. A database dump contains a record of the entire table structure and the data from the database, and has the form of a list of SQL commands.

E.III. Unified Modeling Language and Object-Relational Databases

The Unified Modeling Language

"*A picture is worth a thousand words*"[1].

"*The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artefacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components*" (from the Object Management Group and the UML consortium definition[2]):

Modelling is the designing of software applications before coding. The purpose of introducing the Unified Modelling Language in this document is not to depict the capabilities of this representation language to portray data models, nor to strictly stick to all its specifications (based on UML version 2.0). By using UML notation in our document, we simply try to find a way of visually representing some of the concepts and the relations that govern our implemented model. The reader is invited to read the appendix on UML given at the end of this document (Appendix IV. UML), or to consult it to be familiar with the adopted conventions.

The Object-Relational Database Management System

A relational database, built with a **RDBMS** (Relational Database Management System), is a database with a set of relations that conforms to the relational model. The term relational database refers to both a database's data and schema.

ORDBMS (Object-Relational Database Management System) are fundamentally RDBMS that are extended by an object front end offering the system some object-oriented characteristics. We have chosen the public domain PostregSQL[3] management system to implement our data model. More details on relational databases, on PostgreSQL and on our motivation to use this particular system are given in Appendix V. (Relational databases).

[1] René Descartes (1596-1650)

[2] http://www.uml.org

[3] http://www.postgres.org

E.IV. The central Make2D-DB II data model and its implementation

Modelling should allow us to reflect data structure and relationships between the entities. **Class models** consist of objects (entities) that interact by sending each other messages. **Objects** include things they recognise, the **attributes**, and things they can do, their **behaviours** or **operations**. The values of an object's attributes determine its state. Objects are instances of their classes, which are themselves the blueprints for objects. A class wraps attributes (data) and behaviours (methods or functions) in a distinct group of entities.

A physical model reflects the real implementation of a logical conceptual model, which itself is an abstraction of the material implementation. Both can be described in UML by a class diagram. As far as the relational implementation is concerned, it is technically fully described at the following address:

- ***http://world-2dpage.expasy.org/make2ddb/1.Readme_main.html#schema***

Relations (classes) are listed in alphabetical order and are followed by the different server-side functions or procedures (operations). The entire relational implementation consists of four distinct **schemas**, in which entities are grouped in distinct **namespaces**:

- The ***core*** schema: This is the central schema, which contains all the data. It regulates almost all the main relations and behaviour of the database implementation. The core schema reflects nearly the data model in its totality and is almost self-sufficient (though some of the required operations are defined in the common schema). It contains all private and public data and it can only be accessed by the database administrator and privileged users. All relations, except the materialised views, end with the two attributes *userStamp*, the user who entered the data, and *update*, the time of the insertion or the update.

- The ***public*** schema: The structure of the public schema is a mirror image of that of the core schema (all relations and indexes), but without the operations, the procedures, the associations and the constraints of the core schema. Data from the core schema is filtered from any data marked private before being exported into this schema. The export is regulated by a set of procedures implemented in the core schema. An ordinary user will only access data from the public schema.

- The ***common*** schema: This schema holds some common procedures to both previous schemas. The core and the common schema put together constitute the totality of the 2-DE data model, and are therefore self-sufficient. The common schema has only a single table, with a unique tuple, that is related to the database identity and current state.

- The ***log*** schema: The role of this schema is to register any data modification applied to the core schema. Any tuple that is modified or deleted is reported here. The log schema contains a copy of all the relations of the core schema, except the materialised views. Each table has three additional attributes:

modificationDate, stating when the modification occurred, *updatedOrDeleted*, indicating whether the data was updated or deleted, and *userModifierName*, giving the name of the user who made the modification. Like the public schema, the log schema has no associations or operations.

E.IV.1 URL addresses to access a specific schema implementation

- The *core* schema:
  http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html
- The *public* schema:
  http://world-2dpage.expasy.org/make2ddb/database_schema/public_schema.html
- The *common* schema:
  http://world-2dpage.expasy.org/make2ddb/database_schema/common_schema.html
- The *log* schema:
  http://world-2dpage.expasy.org/make2ddb/database_schema/log_schema.html
- The entire implementation:
  http://world-2dpage.expasy.org/make2ddb/database_schema/all.html

E.IV.2 More of a physical data model than a logical data model

The PEDRo model (Proteome Experimental Data Repository) (Taylor et al. 2003) is a proposal for a standard representation to support the sharing of proteomics data. It aims to capture proteomics data and to make it available for browsing, searching and downloading. PEDRo is principally a conceptual data model that inspired PSI early stage developments in defining standards for data representation in proteomics.

In contrast to the PEDRo model, or the ongoing PSI-OMs (PSI Object Models), the Make2D-DB II data model is not intended to conceptualise a proteomics experiment. It rather aims at effectively implement a functional and evolutionary design built on top of an already existent model (the initial structure deployed in SWISS-2DPAGE flat file entries). In particular, the data model we present here is intended to publish and interrelate 2-DE datasets. It is also meant to reflect a physical relational implementation. Many practical aspects make it distinct from a purely conceptual or logical model:

References to external data documents

The model is not intended to capture all details concerning the many aspects of a proteomics experiment. It is not a Laboratory Information Management System (LIMS); such a system would be outside the scope of our project. Specifically, we do not incorporate all the details that can be covered by external documents, such as full protocols or the various PSI-MIAPE documents (C.IV.8). In many cases, these documents are referenced within a class by their location, *i.e.*, a Web address (the *URI* attribute) or a local file path (the *xxx*D*ocument* attribute). However, we sometimes want some specific data contained in an external document to be directly stored in the relational database for efficient search or comparison purposes, *e.g.*, by extracting all peak list values from a mzData, a mzXML, a "dta" or a "pkl" file. Parsers are implemented in order to extract such data, providing the documents are in a common standard format. As the finalisation and approval of standard PSI formats are still in progress, the use of additional parsers is expected in the future to directly integrate in the database relevant data from an additional variety of external documents.

Denormalisation

Normalisation is a formal approach in data modeling to examine and validate attributes with classes in the logical data model. It is defined by a set of rules that, basically, ensures that each attribute belongs to the class to which it has been defined, that redundant information is minimised, and that logical anomalies are eliminated. Denormalisation is an intentional violation of the rules of normalization done to increase performance in an implemented database and to adapt to some specific situations where data may be incomplete. Denormalisation cannot be done without a thorough understanding of the data and the needs of the database users. The implementation of the 2-DE data model is made more realistic by tolerating a certain level of denormalisation, as listed below.

Flexibility with missing data

In practice, the data needed to define all the theoretically required attributes to instantiate a specific class is often incomplete, because only a subset or none of this data is provided. We may sometimes want to capture the incomplete data and complete it with “Not Defined” values for the missing attributes, even though at the conceptual level, some of the missing attributes are not optional, *i.e.*, they should be defined. Development of the management and querying environment dealing with the database implementation must be fully aware of any such concessions, in their full-context, to correctly handle and present data to end-users. We will come across many such situations in this section while presenting our data model. This aspect leads us to another practical aspect, which is the shortcutting of some associations.

Practical flexibility by shortcutting some associations

We choose to concurrently establish some associations at different levels of our model. In some situations, data is directly captured within a class that, otherwise, would have relied on another class to indirectly “know” about this same data. For the integrity of the implementation, we ensure that some methods guarantee that no conflicting redundancies will arise (for example, by giving an order of preference for concurrent associations). Let us consider the following example to illustrate such a situation (Figure E.IV-1, a simplified situation for illustration purpose):

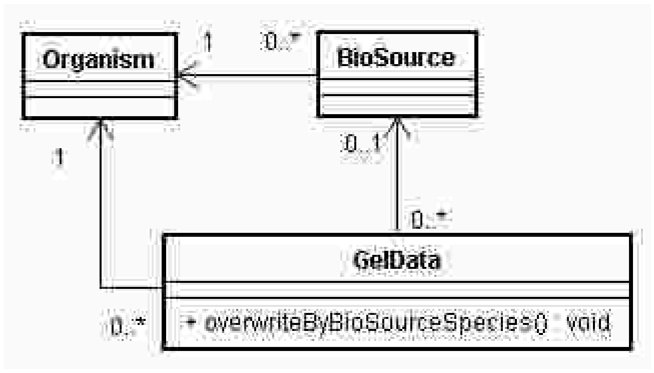


Figure E.IV-1: Example of a concurrent association.

To be quite flexible, the model does not necessarily require biosource[1] related data when instantiating a *GelData* object (multiplicity of 0..1 of *BioSource* objects for a *GelData* object). While it is appropriate to identify the studied organism at the biosource level, the gel classes are also structured to link directly to the *Organism* class. Implemented methods ensure nevertheless that no incompatibility will arise in such a situation. For example, an *Organism* reference at the *GelData* level is overwritten by any *Organism* reference from *BioSource* if the latter is defined.

Analogous classes

Due to the inhomogeneity of some resources concerning the same type of biological data, we sometimes resort to employ diverse classes for the same category of information. These classes may totally differ by their data structure, and thus are not related by any generalisation. The two "analogous" classes *Tissue* and *TissueSP* represent an example of this case:

- *http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html#core.table.tissue*
- *http://world-dpage.expasy.org/ make2ddb/database_schema/core_schema.html#core.table.tissuesp*

The former is a class defining tissue names in a hierarchical manner and is not currently populated (but which could be if a hierarchical classification of tissues was to be agreed on in proteomics). The latter is a class that contains a plain list of tissue names as defined by the Swiss-Prot group.

Whenever a mapping is potentially possible between two analogous classes, we create an association class to relate them. An example is the class that maps between the general tissue class and the Swiss-Prot tissue class.

- *http://world-2dpage.expasy.org/ make2ddb/database_schema/core_schema.html#core.table.tissuesptissuemapping*

[1] Biological material: origin, preparation and description.

Materialised views

In the relational world, a virtual view is a derived relation, which means that the information it contains is derived from other sources. On the contrary, a materialised view is a concrete table. Materialised views are extensively used and presented as physical classes in our data model. They offer an efficient access for pre-formatted data views, but at the cost of being potentially out-of-date. Due to their physical materialisation, they can be considered in modelling as full-fledged classes with extreme dependency on several other classes. They are accompanied in our data model by a full collection of management operations and interfaces to facilitate their construction and update. The protein entry views that are physically constructed from many atomic attributes distributed all over the various classes are an example of these materialised views.

Internal and External operations

Operations have different roles, ranging from performing batch commands, constructing materialised views, rewriting attribute values, etc. In addition, they can verify and apply some constraints that cannot be expressed in a relational model. Such is the case when an attribute data domain must be restrained to satisfy a condition depending on other relations' content (for example, to verify that all the gels on which a specific protein has been identified belong to the same organism). Many operations are implemented as internal operations, or server-side functions, meaning that they are part of the PostgreSQL implementation and that they operate from within the relational environment. However, a number of operations are applied from outside the relational environment. They are invoked during the data analysis process that is in charge of checking the data before populating the database. We will refer to such operations by external operations or external methods, and we will precede their name by the package-visibility sign "~".

Management classes

The model includes many classes and operations that are directly related to the technical aspects of the implementation and to the specific management of some particular biological data.

As already stated, due to practical considerations, the data model has not been constructed with a *top-down*[1] but rather a *bottom-up* approach. An internal model has been initially built to catch all the existing SWISS-2DPAGE data representation. An early operational implementation of the model was rapidly made available for the EBP project. This implementation made it then possible for the project partners to share 2-DE data. It is only with the first public version, version 1.00, that the model was extended to include some more general concepts covering, for example, project and sample description, as well as more details on the identifications.

[1] http://en.wikipedia.org/wiki/Top-down_and_bottom-up_design

Ontology and use of controlled vocabulary

Ontology can be seen as "*a formal, explicit specification of a shared conceptualisation*" (Gruber 1994), a description of the concepts and relationships between defined objects or notions. Object representations in ontologies are frequently used as basic concepts for object models used in database design. We may therefore consider the definitions and the relationships of the different classes and attributes in our model to reflect an ontology that is suitable for the purpose of our work.

Not all the ontology we are concerned with is directly expressed in the data model implementation. An agreement to use a vocabulary respecting the assumptions specified by an ontology in a consistent way is defined as a **controlled vocabulary**. Many classes within our model make use of such a controlled vocabulary.

E.V. The main constituents of the data model

E.V.1 Schematic classification

The main constituents of the model can be schematically arranged into interrelated packages (Figure E.V-1). This schematic classification does not reflect distinct entities of the model, as some of these packages are rather overlapping. Some classes and implemented interfaces may belong indeed to several of these entities. The purpose of this separation is to give the reader a simplified picture of the aspects covered by the model. The interrelated packages in version 2.50.2 of the tool are the following:

- *Projects, Biosources and Samples*: Project description, organism, study groups and individuals, biosource (biological source), tissue, sample preparation, …

- *Analytes and Separation Techniques (2-DE)*: Analytes, 2-DE technique, gel protocols, dimension, images, related documents, set of gels for specific protein entries, …

- *Spots*: Spots as distinct separated entities, identifiers, coordinates, physical properties, …

- *Identifications*: Analyses, identification techniques, experimental results, identified proteins, protein related annotations and comments, related documents, …

- *Protein Annotations*: Identifiers, description, localisation, ontology classification, static cross-references, …

- *Cross-References*: Dynamic cross-references and external databases' metadata.

- *External General and 2-DE Data*: External data integration, UniProtKB data, protein functions and classification, related remote 2-DE maps for same species and tissue, …

- o *Bibliographic References*: Type of publication, location, authors, … Contacts and people can also be categorised as references.

- o *Materialised Views and batch operations*: Consolidation of data to form non-virtual views.

- o *Additional Ontology and CVs*: Ontology and Controlled Vocabularies are of different nature and intervene in many classes.

- o *Metadata and Technical Records*: Database metadata, package version, operations' dates, history of data modifications, …

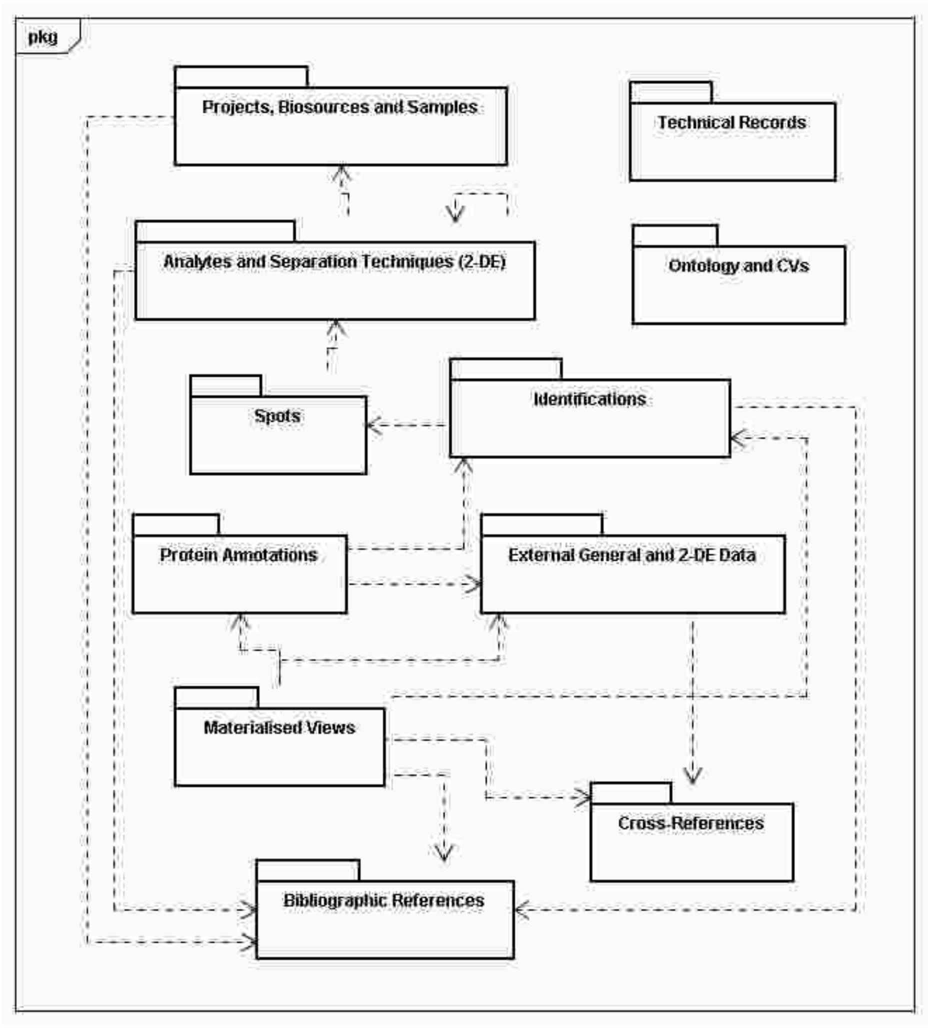


Figure E.V-1: Main constituents of the Data Model.

In Figure E.V-1, not only is the package classification schematic, but so are the dependencies, in the sense that we only want for the moment to represent interdependency at a very high level. The next sections will focus on the details of the different constituents. This will cover the inner classes, their attributes and operations, the associations between the classes and the related interfaces. As already stated, some entities will appear more than once, either because they may be considered part of several packages, or because they will be needed to explain the relationship between one package and another.

Some conventions:

- We will explicitly list hereafter the relational Primary Key attributes for the majority of the classes / relations. They will have a "{PK}" constraint attached to them. Sometimes the Primary Key of a class is a combination of several attributes. For example, two attributes (*spotID* and *GelID*) are needed to instantiate a *Spot* primary key. Whenever the Primary Key is a combination, all the attributes involved in the combination will each have a "{PK}" constraint. However, no Primary Keys will be listed in association classes because, as far as these classes are concerned, Primary Keys are typically a combination of those of their associated classes.

- The relational Foreign Keys are not listed, as they are implicitly defined in the associations drawn between the classes. Foreign keys in classes referencing themselves (parent/child relationships) and those that are invoked in a constraint within the class (*e.g.*, uniqueness over several attributes or {Check *constraint*}) are exceptions to this convention.

- When there should be a uniqueness constraint on a combination of several attributes, a constraint "{Unique#n}" will be attached to each attribute of this combination. The number 'n' is an identifier for this particular constraint. Many distinct combination constraints may coexist within the same class.

- Attributes will not systematically be displayed within a class, especially if they have already been displayed in a previous figure. Conversely, some classes will not show their attributes in detail if the classes are described in more details afterwards.

- Some attributes have a name of the form *condidion1orCondition2* (*e.g.*, *IsAorPartOf*). They are boolean attributes that, when defined to be TRUE, validate *condition1*, and when defined to be FALSE, validate *condition2*. If the attribute is not defined (NULL), then neither condition is verified.

- In an inheritance relationship, we will commonly attach the suffix "*Parent*" to the top most parent class name, *e.g.*, *SpotIdentification**Parent***. A child class in such a relation may sometimes be referred to using the "*Child*" suffix, *e.g.*, *SpotIdentification**Child***.

- Many triggers, which are activation processes that are associated with some function or procedure and that are "fired" before or after a specific operation is attempted on a tuple, are directly associated with the relation that contains the target tuple, and whose manipulation causes the automatic activation of the trigger.

- A *user* refers to the person who uses the tool to build a database and publish his/her data. A *database user* or *end-user* is a person who accesses this data.

The reader is also encouraged to get familiar with the SWISS-2DPAGE and its annotations, as described in the database user manual at: http://www.expasy.org/ch2d/manch2d.html.

E.V.2 Projects, biosources and samples

This part of the module lies on top of the other main parts that cover proteomics analyses. Mainly centred on the biological material generation, this part acts like a container for the analytes that will undergo the process of protein separation and identification. For the time being, there is still a need for standard definitions in biological material and sample reporting in proteomics. The MGED community have already achieved significant advances in its definition of sample reporting in microarrays experiments (Ball, Brazma 2006), and we have been partly inspired by their representation of samples[1]. We have also some similarities with the eVOC ontology[2] (Kelso et al. 2003) in the definition of some of the characteristics of biosources (that they call "samples"). An adjustment to PSI propositions[3] has been also partially performed, but without fully adopting all the propositions, as they are not yet refined. Our model differs in particular from a conceptual point of view from the unfinished PSI model in what is defined to be a biosource, a sample and an analyte.

We believe this part is temporarily incomplete as far as attribute definition and semantics are concerned. The presented classes are the backbone for a more comprehensive representation to cover projects, biosources and samples, and they can be easily extended in the future. Many of these classes currently include pointers to external documents (*i.e.*, Web links or local files). These documents are assumed to cover project and study descriptions, biosource details and sample preparation protocols. The lack of a standard representation in such documents makes them not viable to parse, which prevents their content or parts of it from being extracted and imported into the database. Nevertheless, by linking to these documents, the database ensures the display of a minimum supply of materials that cover such aspects of a proteomics experiment[4].

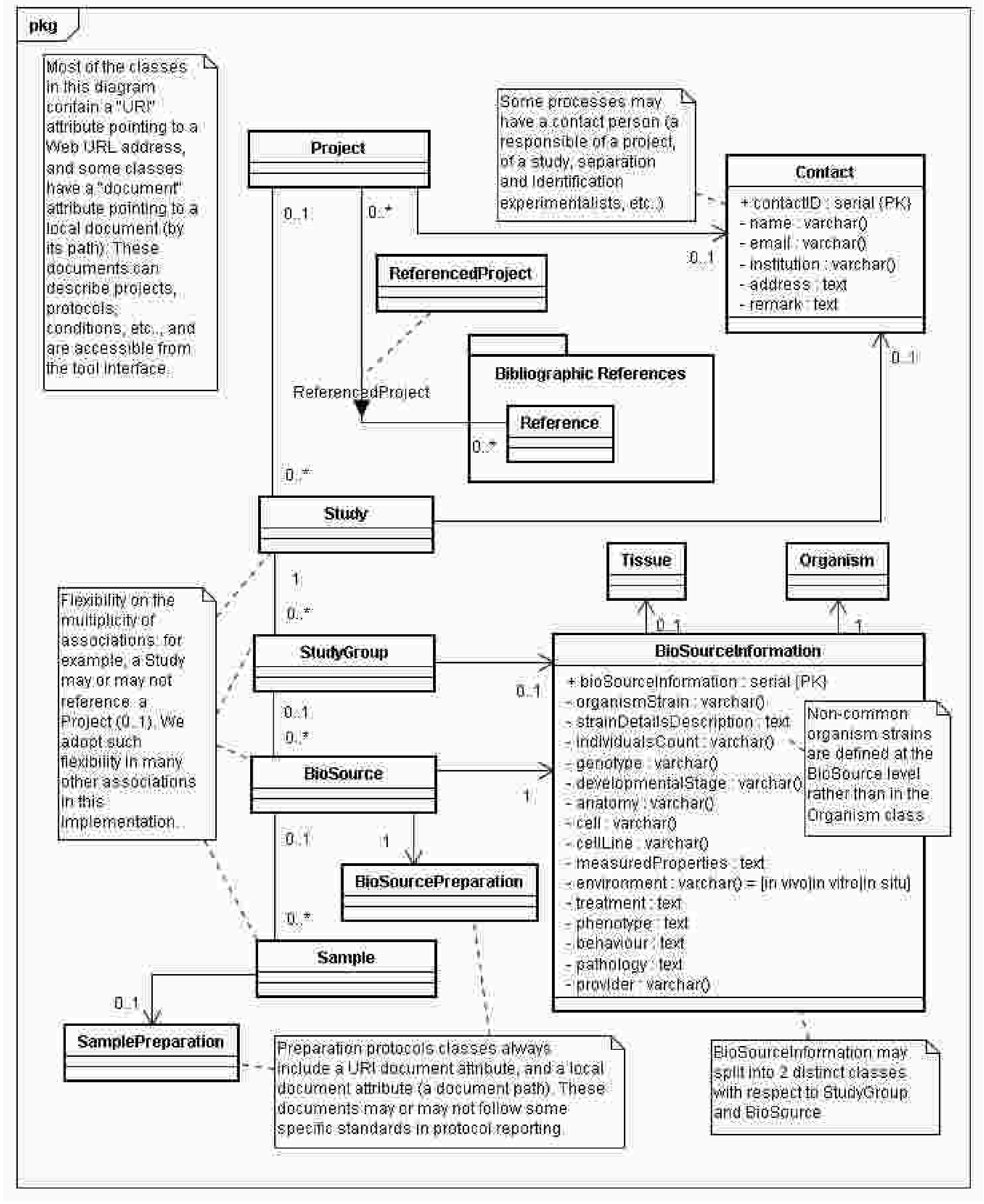
Figure E.V-2 shows the classes that are implemented at the project and biosource levels, as well as their relationship. Most of the displayed attributes are self-explanatory. This part of the schema widely includes what we have already defined as "*flexibility with missing data*" (E.IV.2), meaning that many supposedly required attributes (in a conceptual approach) are optional.

1 http://www.mged.org/Workgroups/MIAME/miame.html

2 http://www.evocontology.org/site/Main/OntologyDescriptions

3 http://www.psidev.info/index.php?q=node/90

4 The database Web interface also provides some other ways of linking and displaying related documents.



Most of the classes in this diagram contain a "URI" attribute pointing to a Web URL address, and some classes have a "document" attribute pointing to a local document (by its path). These documents can describe projects, protocols, conditions, etc., and are accessible from the tool interface.

Some processes may have a contact person (a responsible of a project, of a study, separation and identification experimentalists, etc.)

Flexibility on the multiplicity of associations: for example, a Study may or may not reference a Project (0..1). We adopt such flexibility in many other associations in this implementation.

Non-common organism strains are defined at the BioSource level rather than in the Organism class

Preparation protocols classes always include a URI document attribute, and a local document attribute (a document path). These documents may or may not follow some specific standards in protocol reporting.

BioSourceInformation may split into 2 distinct classes with respect to StudyGroup and BioSource

Figure E.V-2: Data Model - Projects, Biosources and Samples.

We portray a proteomics investigation as part of a main project. The project is refined into distinct studies, each describing generalities about a particular exploration. A studied group implies a uniform grouping of subjects (organisms, individuals, cell cultures, etc.) based on some characteristics. Biosource is the designation of the source of the biological material specifically selected for an analysis. A full biosource description includes both biosource information and preparation. From the biosource, we generate the samples that will be subsequently analysed (a sample preparation describes the generation of a sample, *e.g.*, by an extraction process). At this stage, we do not yet include the material "entity" that will be conditioned for a specific analysis technique, and that will be designated by "Analyte" in the next section. In all cases, the model makes possible the introduction of a biosource with no reference to a project or to a study group. It is also possible to create standalone samples, with no reference to a biosource. Projects, along with studies, may refer to a contact person. Projects may also refer to any number of bibliographic references as well.

An alternative way to create distinct projects

The interfaces that access the database implementations are also designed to access and query simultaneously several databases all at once. This offers the possibility to build distinct databases, each specific to a particular project, and to present all of them to end-users as one unique virtual database. The interfaces also include simple ways to annotate, describe and link to external sources each database (or project) from outside the RDBMS implementation.

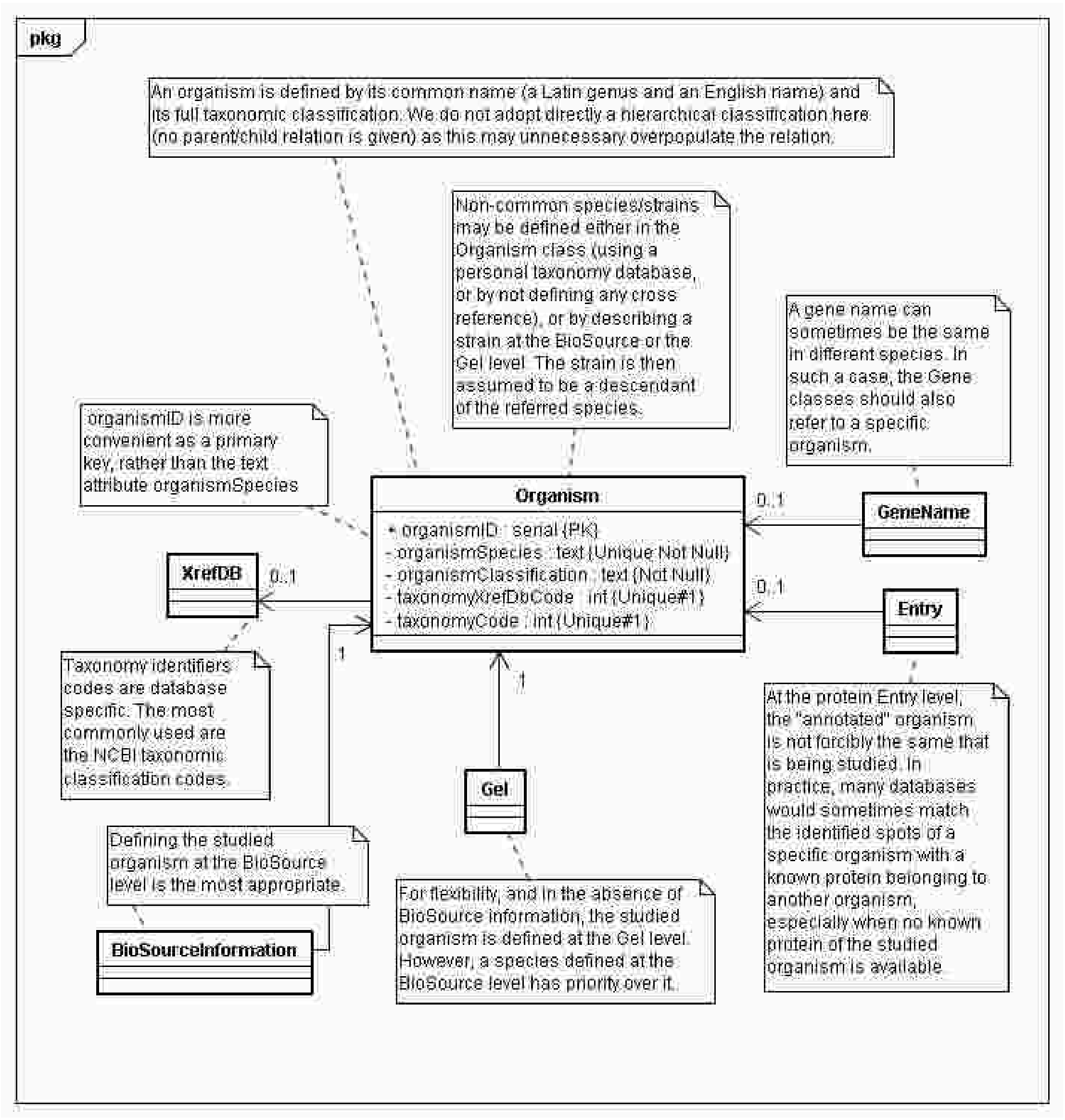
The BioSourceInformation class

This class defines all the details related to both the group of subjects and the selected biosource material. It mainly covers the organism taxonomy (by referencing a known organism), its strain (a less common organism strain defined by the user), a tissue of interest and the various properties related to the development and the characteristics of the selected subjects, such as the cell line, the environment and the treatment. There may be currently an overlap of information due to the concurrency of associations between this class on the one side, and the *StudyGroup* and *BioSource* classes on the other side. To overcome this inconvenience, the *BioSourceInformation* class may split into two distinct classes and undergo heavy structuring of its attributes in future refinement of the model. This will require all *BioSource* instances to obligatorily reference a *StudyGroup* object to cover all the details of the biosource material, which is currently only optional.

The Organism class

An organism is defined in the *Organism* class (Figure E.V-3) by a common name and a taxonomic lineage, which is generally a rooted tree graph classification. The *Organism* class offers also optional references to taxonomic databases, in particular to the NCBI taxonomy database[1]. The NCBI taxonomy database is not a primary source for taxonomic information, but rather an incorporation of taxonomic knowledge from various sources that is used by the nucleotide sequence databases (EMBL/GenBank/DDBJ) and UniProtKB. We would rather reference the studied organism from the *BioSourceInformation* class, but for more flexibility (when no biosource data is provided), the organism can be referenced from the *Gel* class, a central class in our model. The *BioSourceInformation* and the *Gel* classes offer both the possibility to define and to describe any particular strain originating from the referenced species.

[1] http://www.ncbi.nlm.nih.gov/Taxonomy/


pkg
Organism
+ organismID : serial {PK}
- organismSpecies : text {Unique Not Null}
- organismClassification : text {Not Null}
- taxonomyXrefDbCode : int {Unique#1}
- taxonomyCode : int {Unique#1}
XrefDB
0..1
GeneName
0..1
Entry
0..1
Gel
1
BioSourceInformation
1

An organism is defined by its common name (a Latin genus and an English name) and its full taxonomic classification. We do not adopt directly a hierarchical classification here (no parent/child relation is given) as this may unnecessary overpopulate the relation.

Non-common species/strains may be defined either in the Organism class (using a personal taxonomy database, or by not defining any cross reference), or by describing a strain at the BioSource or the Gel level. The strain is then assumed to be a descendant of the referred species.

A gene name can sometimes be the same in different species. In such a case, the Gene classes should also refer to a specific organism.

organismID is more convenient as a primary key, rather than the text attribute organismSpecies

Taxonomy identifiers codes are database specific. The most commonly used are the NCBI taxonomic classification codes.

At the protein Entry level, the "annotated" organism is not forcibly the same that is being studied. In practice, many databases would sometimes match the identified spots of a specific organism with a known protein belonging to another organism, especially when no known protein of the studied organism is available.

Defining the studied organism at the BioSource level is the most appropriate.

For flexibility, and in the absence of BioSource information, the studied organism is defined at the Gel level. However, a species defined at the BioSource level has priority over it.

Figure E.V-3: Data Model – The Organism class.

Many biologists commonly annotate their identified proteins using information that is collected from protein knowledge databases (*e.g.*, from the highly annotated UniProtKB). When no protein specific to the studied organism is available, biologists may use annotations originating from a close ortholog to the protein. In such case, the ortholog's organism is referenced within the model from the *Entry* class (the protein entry class). The gene classes may also directly reference the organism. For many reasons, ambiguity in gene names is frequent in genomics. One source of ambiguity is due to the fact that a gene name, associated with an identified protein in the database, might be shared between several species (homonymy). A direct association between the gene and the *Organism* is therefore required to distinguish homonyms. In most cases, the gene reference to an organism is inherited from the protein that is related to this gene.

The Tissue classes

The tissue classes (Figure E.V-4) have already been cited as an example of analogous classes. The main *Tissue* class integrates any hierarchical tissue classification based on a parent/child relationship (a tree graph). A child may be marked as '*is a*' or '*part of*' relative to his direct parent. Currently, there is no unique list of tissues shared between the different communities. However, many specialised tissue classifications, such as the Brenda tissue classification[1], the eVOC ontology for human gene expression, or the Ontology of Human Development Anatomy[2], can be imported into this representation, depending on the domain of interest. The only negative aspect is that the use of different and non-mapped classifications prevents linking remote databases based on the study of specific tissues. We propose an alternative option by employing a common simple and limited tissue list that can be used in parallel with the main *Tissue* class. This list is the tissue list maintained by the Swiss-Prot group. A plain text list extended by tissue aliases, regularly updated and available from the ExPASy server[3]. The Swiss-Prot tissue list contains all the tissues present in the "TISSUE" topic of the UniProtKB/Swiss-Prot entries RC lines. It is organised in a flat file listing sequentially tissue names and their common aliases. During the database installation, the list is automatically integrated into the *TissueSP* and the *TissueSPAliase* relations. Subsequent updates of the database update also the relations with the most recent version of the list.

[1] http://www.brenda.uni-koeln.de/ontology/tissue/tree/update/update_files/BrendaTissueOBO

[2] http://www.ana.ed.ac.uk/anatomy/database/humat/standard.html

[3] http://www.expasy.org/txt/tisslist.txt

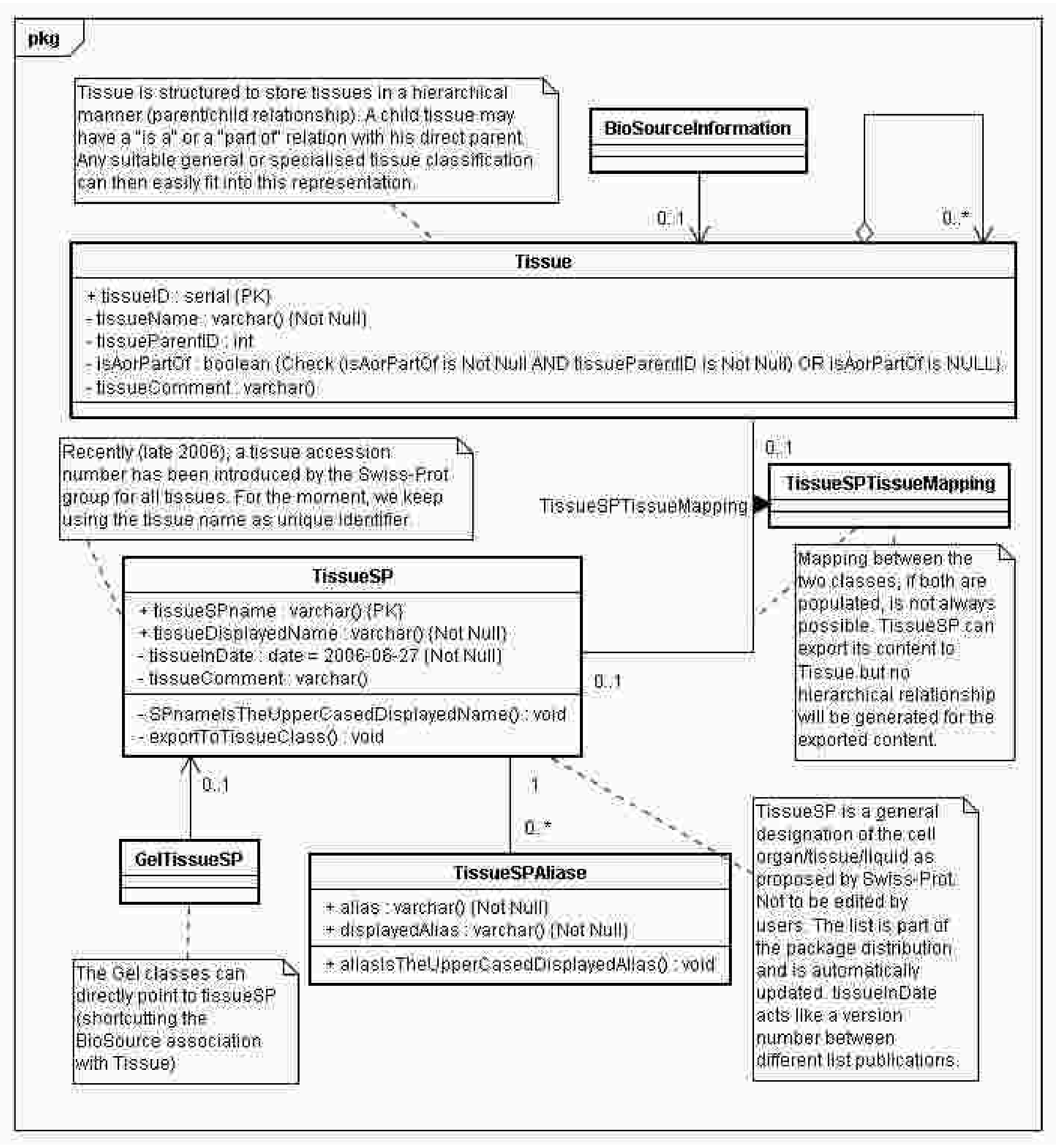

pkg
Tissue is structured to store tissues in a hierarchical manner (parent/child relationship). A child tissue may have a "is a" or a "part of" relation with his direct parent. Any suitable general or specialised tissue classification can then easily fit into this representation.
BioSourceInformation
0..1
0..*
Tissue
+ tissueID : serial (PK)
- tissueName : varchar() (Not Null)
- tissueParentID : int
- isAorPartOf : boolean (Check (isAorPartOf is Not Null AND tissueParentID is Not Null) OR isAorPartOf is NULL)
- tissueComment : varchar()
Recently (late 2006), a tissue accession number has been introduced by the Swiss-Prot group for all tissues. For the moment, we keep using the tissue name as unique identifier.
0..1
TissueSPTissueMapping
TissueSPTissueMapping
Mapping between the two classes, if both are populated, is not always possible. TissueSP can export its content to Tissue but no hierarchical relationship will be generated for the exported content.
TissueSP
+ tissueSPname : varchar() (PK)
+ tissueDisplayedName : varchar() (Not Null)
- tissueInDate : date = 2006-06-27 (Not Null)
- tissueComment : varchar()
- SPnameIsTheUpperCasedDisplayedName() : void
- exportToTissueClass() : void
0..1
0..1
1
0..*
GelTissueSP
TissueSPAliase
+ alias : varchar() (Not Null)
+ displayedAlias : varchar() (Not Null)
+ aliasIsTheUpperCasedDisplayedAlias() : void
TissueSP is a general designation of the cell organ/tissue/liquid as proposed by Swiss-Prot. Not to be edited by users. The list is part of the package distribution and is automatically updated. tissueInDate acts like a version number between different list publications.
The Gel classes can directly point to tissueSP (shortcutting the BioSource association with Tissue)

Figure E.V-4: Data Model - Tissue and Swiss-Prot tissue classes.

Recently, the Swiss-Prot group introduced a new format for the tissue entries, which includes a distinct accession number for each tissue and optional DR lines that cross-reference the entries with the eVOC database. In the next Make2D-DB II release, the newly introduced accession numbers will almost certainly replace the *tissueSPname* primary key used in the *TissueSP* relation. As for the cross-references to the eVOC ontology, they will help mapping between *TissueSP* and *Tissue* whenever an eVOC classification is being used. An extra attribute defining the classification source (*e.g.*, *tissueClassificationSource*) will be required in the *Tissue* relation to help performing the automatic mapping.

All the tissue names and aliases in *TissueSP*, *TissueSPAliase*, *TissueSPTissueMapping* and *GelTissueSP* are uppercased when stored.

An example of a Swiss-Prot tissue entry:

```
ID   Adipose tissue.
AC   TS-0013
SY   Adipose; Fat cell; Fat.
DR   eVOC; EV:0100381; anatomical-system: adipose tissue.
//
```

E.V.3 Analytes, separation techniques (2-DE) and gel related data

Analytes

The *Analyte* class is used to represent analyte items derived from samples. They are the objects over which separation, fractioning and analysis techniques are performed. An analyte can also be created by combining other analytes into a new analyte. This process is known as pooling. It is also possible to create standalone analytes, with no reference to a sample or to other parent analytes (Figure E.V-5).

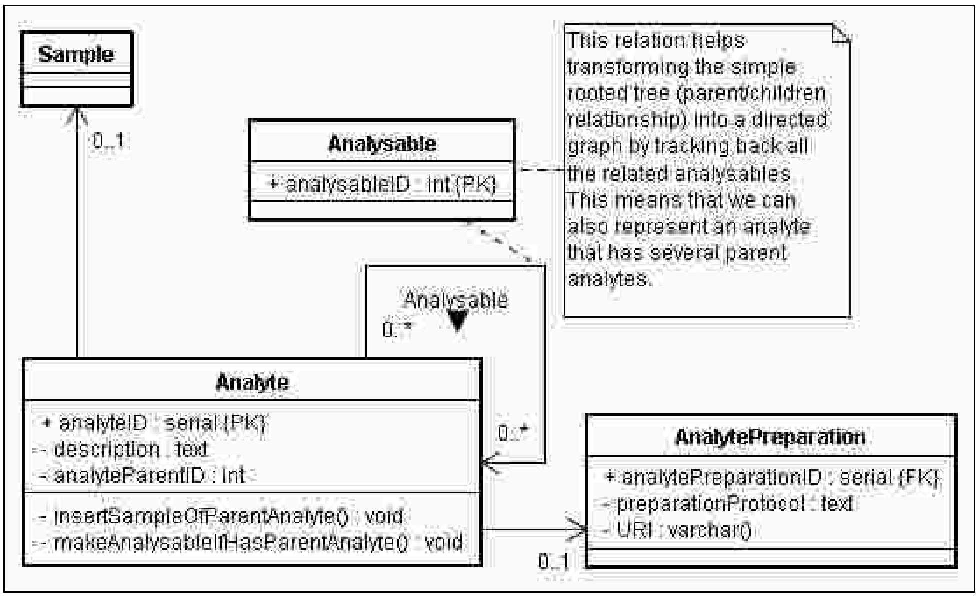

Sample
0..1
Analysable
+ analysableID : int {PK}
This relation helps transforming the simple rooted tree (parent/children relationship) into a directed graph by tracking back all the related analysables. This means that we can also represent an analyte that has several parent analytes.
Analysable
0..*
0..*
Analyte
+ analyteID : serial {PK}
- description : text
- analyteParentID : int
- insertSampleOfParentAnalyte() : void
- makeAnalysableIfHasParentAnalyte() : void
0..1
AnalytePreparation
+ analytePreparationID : serial {FK}
- preparationProtocol : text
- URI : varchar()

Figure E.V-5: Data Model – The analytes mechanism.

The parent/children relation of the *Analyte* class makes it possible to generate several analytes from a parent analyte, to perform, for example, a different separation technique on each of them. A method ensures that all children refer to the same sample of their root parent. Combining several analytes into a new analyte is indirectly implemented using an intermediate class that acts as some sort of analyte registry (the *Analysable* class). All children get an analysable identifier, and those that are combined together share the same identifier. It is then possible to track back all the original analytes forming the combination. In order to be assigned several analysable identifiers, an analyte should then generate an equal number of copies of itself as its own children (fractioning, Figure E.V-6).

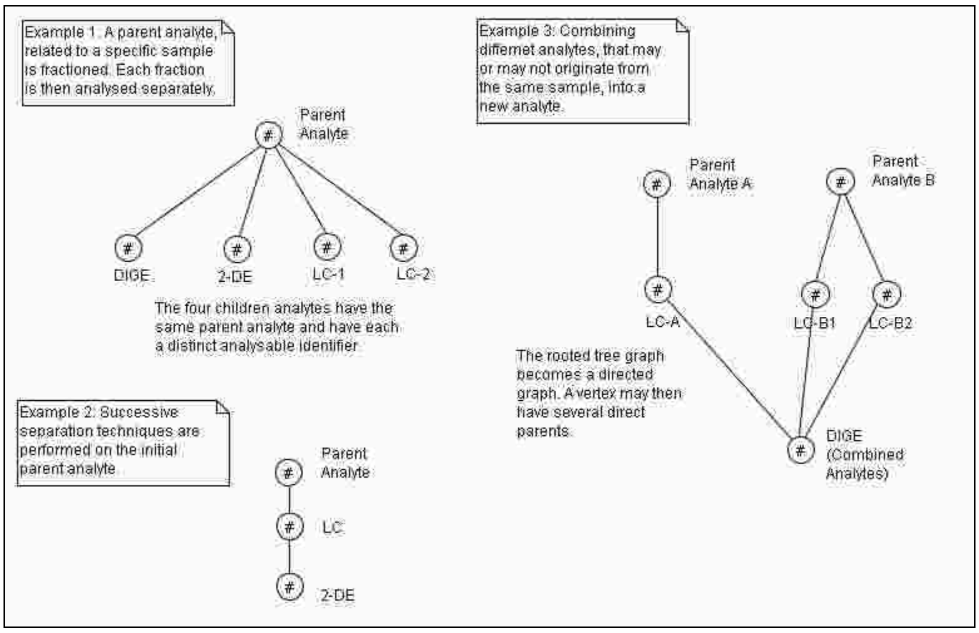
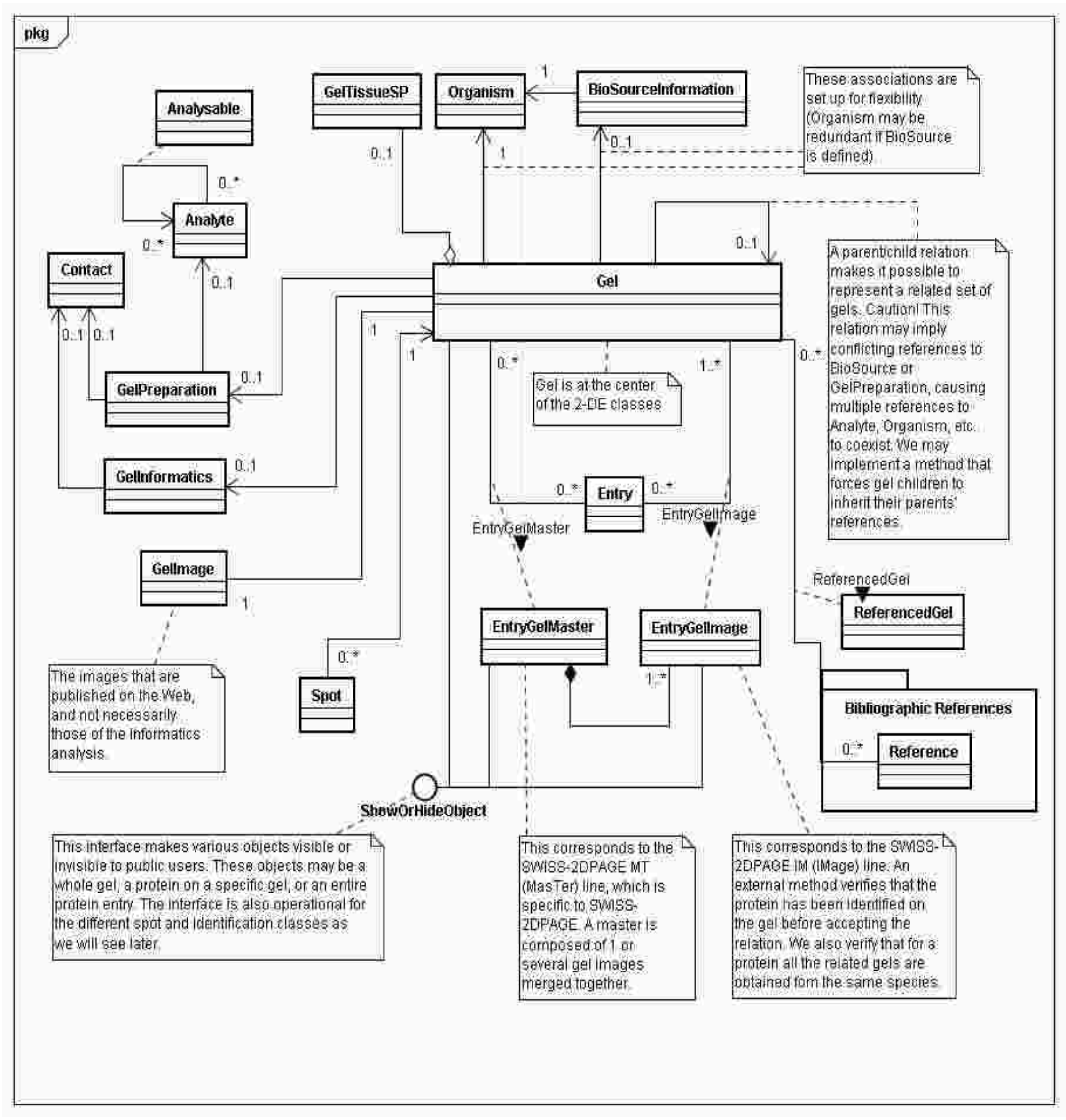

Example 1: A parent analyte, related to a specific sample is fractioned. Each fraction is then analysed separately.
Parent Analyte
DIGE
2-DE
LC-1
LC-2
The four children analytes have the same parent analyte and have each a distinct analysable identifier
Example 2: Successive separation techniques are performed on the initial parent analyte
Parent Analyte
LC
2-DE
Example 3: Combining differnet analytes, that may or may not originate from the same sample, into a new analyte.
Parent Analyte A
Parent Analyte B
LC-A
LC-B1
LC-B2
The rooted tree graph becomes a directed graph. A vertex may then have several direct parents.
DIGE (Combined Analytes)

Figure E.V-6: Some examples of fractioning or combining analytes.

2-DE classes

The *Gel* class is at the centre of the 2-DE package classes (Figure E.V-7), and the objects it instantiates - the gels - are more than essential to the database purpose. The gels have direct associations with *Organism, Tissue* and *BioSourceInformation* classes, as well as with the bibliographic references' classes. These associations are shortcutting the logical path leading from *GelPreparation* up to *Analyte*, *Sample* and *BioSource*, in case no data is provided to follow this regular path. *GelPreparation* and *GelInformatics* define where to find the protocols that are used to prepare the gels and to perform informatics analyses over their scanned images. *GelImage* is therefore just the image file that is displayed to the database users. The gels are also directly related to the entries of the identified proteins they contain. This reflects the protein perspective through which the gel images and masters are listed. The *Spot* class is also related to *Gel*, and we will examine its properties in the appropriate spot package section.



These associations are set up for flexibility (Organism may be redundant if BioSource is defined)

A parent/child relation makes it possible to represent a related set of gels. Caution! This relation may imply conflicting references to BioSource or GelPreparation, causing multiple references to Analyte, Organism, etc. to coexist. We may implement a method that forces gel children to inherit their parents' references.

Gel is at the center of the 2-DE classes

The images that are published on the Web, and not necessarily those of the informatics analysis.

This interface makes various objects visible or invisible to public users. These objects may be a whole gel, a protein on a specific gel, or an entire protein entry. The interface is also operational for the different spot and identification classes as we will see later.

This corresponds to the SWISS-2DPAGE MT (MasTer) line, which is specific to SWISS-2DPAGE. A master is composed of 1 or several gel images merged together.

This corresponds to the SWISS-2DPAGE IM (IMage) line. An external method verifies that the protein has been identified on the gel before accepting the relation. We also verify that for a protein all the related gels are obtained fom the same species.

Figure E.V-7: Data Model – The 2-DE classes.

The Gel Class

In addition to the direct association with the *Organism* class, the Gel class may store details about the organism strain that is being studied. This information is exploited if no biosource information is given. The *Gel* class is also linked to the bibliographic references' package in a many-to-many association through the *ReferencedGel* class. A linkage that, in theory, may be inherited from the root *Project* class. A tissue, as defined by the *TissueSP* class, can also be attached to a gel by means of the *GelTissueSP* class. We have not yet implemented a direct association between *Gel* and the more general *Tissue* class as, for the moment, only *TissueSP* is populated during the database installation process.

We do not separate 1-DE and 2-DE gels in distinct classes. In our conception, the relevant element is the identity of a container - the gel – an object with a defined area over which the detected spots are located. Other classes are in charge of describing the corresponding gel characteristics, preparation protocol and informatics analysis. Only the electrophoresis dimension is reported in this class. Information about the gel measurements is optional (the start and end pI and Mw).

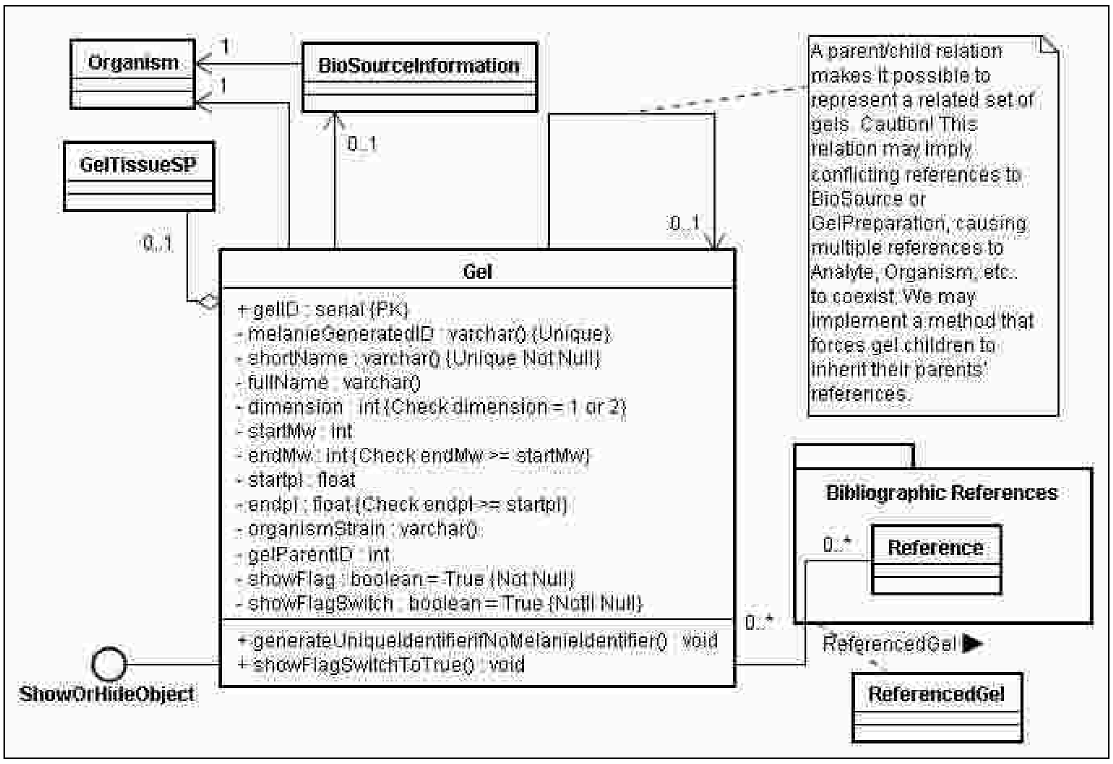

Organism
1
1
BioSourceInformation
0..1
GelTissueSP
0..1
0..1
A parent/child relation makes it possible to represent a related set of gels. Caution! This relation may imply conflicting references to BioSource or GelPreparation, causing multiple references to Analyte, Organism, etc. to coexist. We may implement a method that forces gel children to inherit their parents' references.
Gel
+ gelID : serial {PK}
- melanieGeneratedID : varchar() {Unique}
- shortName : varchar() {Unique Not Null}
- fullName : varchar()
- dimension : int {Check dimension = 1 or 2}
- startMw : int
- endMw : int {Check endMw >= startMw}
- startpI : float
- endpI : float {Check endpI >= startpI}
- organismStrain : varchar()
- gelParentID : int
- showFlag : boolean = True {Not Null}
- showFlagSwitch : boolean = True {Notll Null}
+ generateUniqueIdentifierIfNoMelanieIdentifier() : void
+ showFlagSwitchToTrue() : void
Bibliographic References
0..*
Reference
0..*
ReferencedGel ▶
ReferencedGel
ShowOrHideObject

Figure E.V-8: Data Model – The gel class.

The *Gel* class has a numerical internal identifier, but uses an additional alphanumerical identifier: the *shortName* attribute. This identifier serves to designate the gel in the database using an informative mnemonic name or joined words that may include information about the species, the tissue, the pH range, etc. (*e.g.*, HUMAN-PLASMA, MOUSE-LIVER-4-7). *shortName* is systematically lowercased when stored in the Gel class. A larger description of the gel is given by the optional *fullName* attribute. An additional characteristic attribute, the *melanieGeneratedID*, holds a third special identifier serving to unequivocally identify the gel within the virtual global 2-DE database. The attribute name is inherited from the Melanie / ImageMaster^{TM} 2D Platinum (*cf.* Figure B.III-4) 2-DE gel analysis software. This software generates a unique key for each gel and this key can be imported into the database (as data exports generated by Melanie can be directly read by Make2D-DB II). In the absence of this value, the database implementation generates an equivalent unique identifier based on the database and the gel names. When the database is updated, this generated identifier is maintained and transmitted, even if the gel full name and the database name have changed.

To represent a set of related gels we take advantage of the parent/child relationship between a reference gel, the parent, and a group of gels, its children. A master is a reference gel (or reference map). In practice, it is common to merge several related gels with an initial "arbitrary" reference gel, *e.g.*, by gel matching; thus improving the quality or the inclusiveness of the master gel. In its current state, the tool does not physically implement any procedure forcing all the gels that belong to one set to unconditionally share identical references, like having a common preparation protocol and the same sample and biosource. We have not tested yet this concept (of many related gels) with real datasets, and the Web query interface does not explicitly put forward this relation. If the need to make use of such a representation arises, the supporting management procedures should be simple to implement.

Two attributes control the privacy of a gel and its visibility to public end-users. The *showFlag* attribute tells if the current state of the gel is visible or not. The other attribute, the *showFlagSwitch*, is used internally by the procedures that update the materialised views (in order to avoid unnecessary efforts when only the public schema needs to be updated, but not the core schema[1]). The collection of procedures symbolised by the *ShowOrHideObject* interface controls the access to the visibility attributes. For example, once the visibility inversion has been applied on public data, the *showFlagSwitch* is ordered to change its state. The mechanism managing the privacy of data is also applied to many other objects within the database, as we will find out when describing other classes, like protein entries, spots and identification results.

The Gel protocols

The *GelPreparation* and *GelInformatics* define where to find the protocols used to prepare the gels and to perform informatics analyses over their scanned images. Both classes have pointers to a Web URL address and to a path to a local document where the corresponding protocols are located. *preparatoinDescription* and *informaticsDescription* are symbolic free text attributes. They should be extended in the future into a more detailed set of attributes, and should be populated by meaningful data extracted from the related protocol documents. The *soft* attribute in *GelInformatics* is also free text (for the moment). It is associated with the detection software, its version and the used detection algorithm[2].

[1] The full algorithm is given at http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_final_views.pgsql, under "Global Updates"

[2] Links to the gel protocols can also be directly given from the interface configuration files.

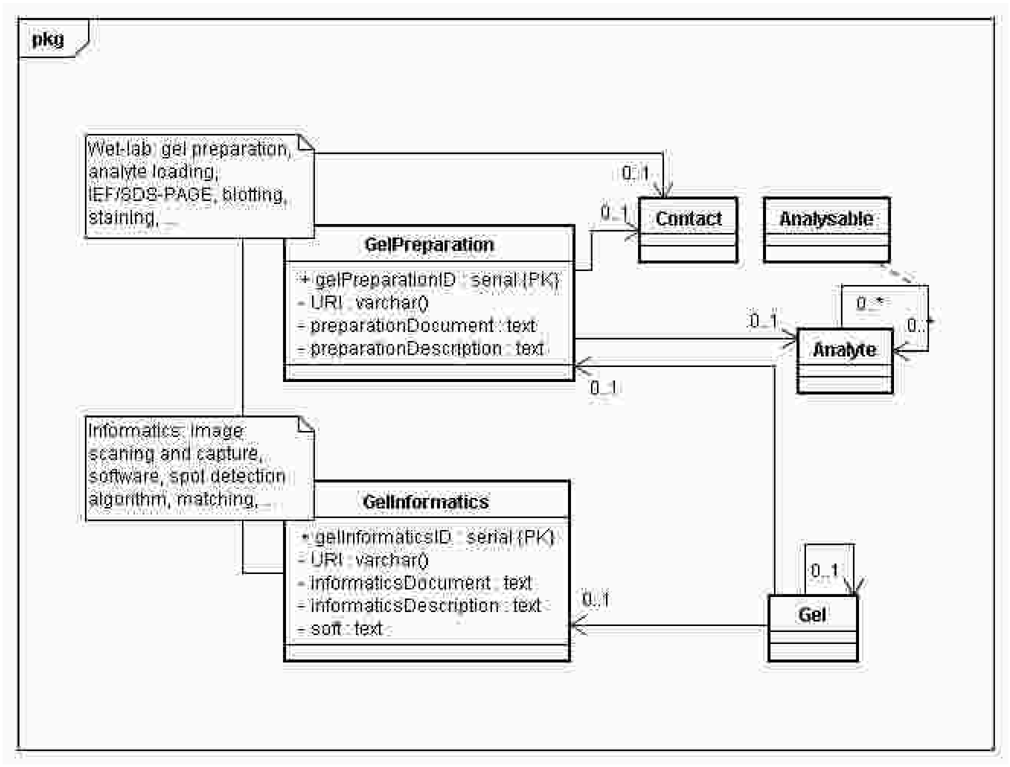

pkg
Wet-lab: gel preparation, analyte loading, IEF/SDS-PAGE, blotting, staining, ...
GelPreparation
+ gelPreparationID : serial {PK}
- URI : varchar()
- preparationDocument : text
- preparationDescription : text
Contact
Analysable
Analyte
0..1
0..*
Informatics: image scaning and capture, software, spot detection algorithm, matching, ...
GelInformatics
+ gelInformaticsID : serial {PK}
- URI : varchar()
- informaticsDocument : text
- informaticsDescription : text
- soft : text
Gel

Figure E.V-9: Data Model – The gel protocols.

Gel Image

The characteristics of the graphical image that is presented to the database users are defined in the *GelImage* class. The image can be stored in the database as a BLOB (binary large object), but due to the non-portability of such objects in PostgreSQL, we have chosen not to rely on this attribute to physically store and load the image file. The image is therefore better given by its URL (the location will be automatically evaluated based on the interface configuration parameters), as well as its local system path and filename on the server machine. The image type can be any standard graphical type recognised by Web browsers. Images compressed in JPG format are a good compromise between quality and file size, which is essential for a rapid transfer through the Web. A small image, a thumbnail of the main image that is also displayed by the Web interface, needs the same definitions.

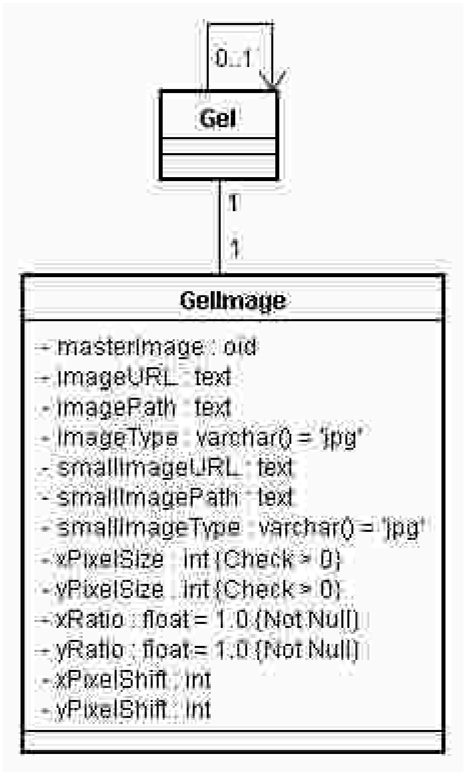

0..1
Gel
1
1
GelImage
- masterImage : oid
- imageURL : text
- imagePath : text
- imageType : varchar() = 'jpg'
- smallImageURL : text
- smallImagePath : text
- smallImageType : varchar() = 'jpg'
- xPixelSize : int {Check > 0}
- yPixelSize : int {Check > 0}
- xRatio : float = 1.0 {Not Null}
- yRatio : float = 1.0 {Not Null}
- xPixelShift : int
- yPixelShift : int

Figure E.V-10: Data Model – The gel image.

The dimension of the original image (the image used to detect the spots) is given by two attributes: *xPixelSize* for the 'x' axis (left to right, corresponding to pI) and *yPixelSize* for the 'y' axis (top to bottom, corresponding to Mw). The top left corner is consequently the origin of the coordinate system, and pixels (picture elements) are the units of measure[1]. In some cases, the original images are modified to include, for example, a legend or a pI and a Mw axis. This results in the origin of the coordinates system being shifted. The shift must therefore be reported in the *xPixelShift* and *yPixelShift* attributes. It is also frequent that the image published on the Web is only a reduced image of the original one (which may be a very large image). In such cases, we have to provide the ratio of the dimension decrease (or increase) for both axes independently. For example, defining *xRatio* and *yRatio* = 0.5 means that the displayed image is half-large and half-length of the original one. For practical reasons, almost all the *GelImage* attributes may be overwritten by the Web interface configuration file, which is a simple text file that can be modified at any moment by the database administrator.

Entry related classes

From the protein perspective, an inclusive list of the gels in which the protein has been identified or matched can be supplied. The list is a record of the images

[1] The dimension of some image types, in particular GIF format, is automatically extracted from the image file during the installation process.

(equivalent to the SWISS-2DPAGE *IM* list), and optionally the masters (equivalent to the SWISS-2DPAGE *MT* list), of the corresponding gels.

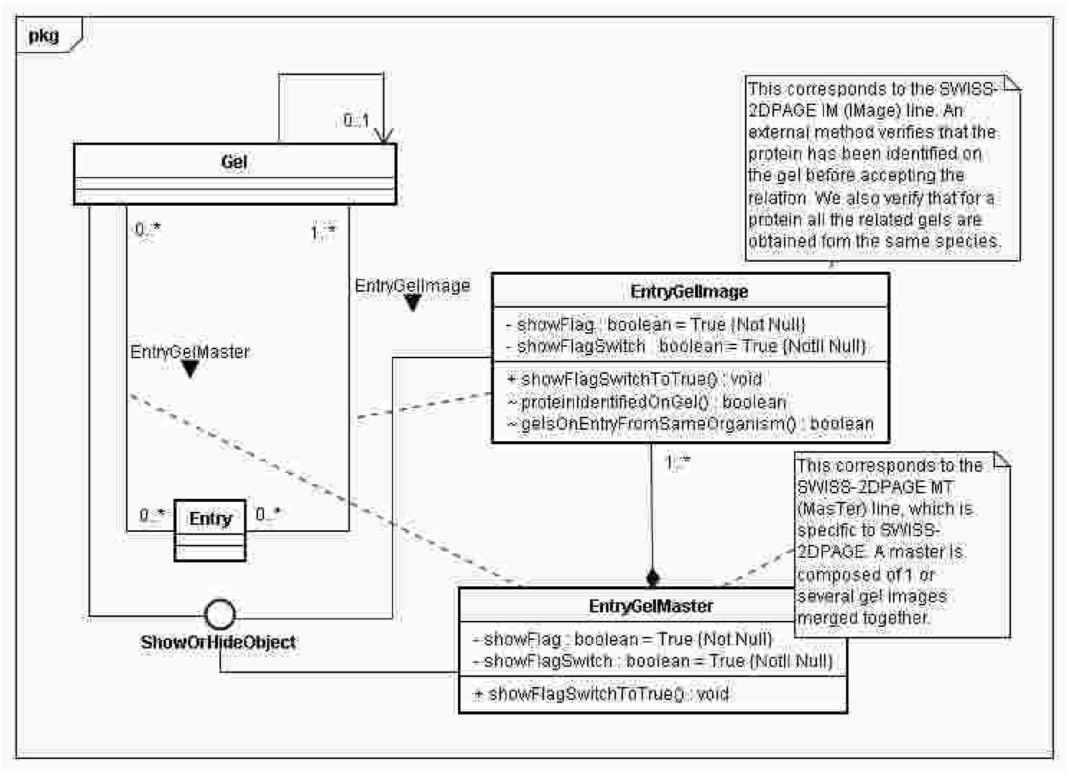

pkg
0..1
Gel
0..*
1..*
EntryGelImage
EntryGelMaster
0..*
Entry
0..*
ShowOrHideObject
This corresponds to the SWISS-2DPAGE IM (IMage) line. An external method verifies that the protein has been identified on the gel before accepting the relation. We also verify that for a protein all the related gels are obtained fom the same species.
EntryGelImage
- showFlag : boolean = True {Not Null}
- showFlagSwitch : boolean = True {Notll Null}
+ showFlagSwitchToTrue() : void
~ proteinIdentifiedOnGel() : boolean
~ gelsOnEntryFromSameOrganism() : boolean
1..*
This corresponds to the SWISS-2DPAGE MT (MasTer) line, which is specific to SWISS-2DPAGE. A master is composed of 1 or several gel images merged together.
EntryGelMaster
- showFlag : boolean = True {Not Null}
- showFlagSwitch : boolean = True {Notll Null}
+ showFlagSwitchToTrue() : void

Figure E.V-11: Data Model – Relating the Gel class to the Entry class.

The relation between the two association classes connecting *Gel* and *Entry*, namely *EntryGelImage* and *EntryGelMaster*, can be represented by a composition relationship (an aggregation type). This relationship may sometimes be implicit, because the hierarchical parent/child relationship of the *Gel* class itself can already represent a set of children gels related to the same parent. The parent would consequently be the reference gel and the children the additional merged gels. Many published datasets provide data only for master gels, without further details on the individual gels that have been merged with the masters. Hence, for practical or for simplicity purposes, the gel lists given by *EntryGelImage* and *EntryGelMaster* is often identical (which implies a one-to-one, and not a one-to-many, relation between the two classes). In such a case, the tool just relies on the *EntryGelImage* class when listing the gels that are related to a specific protein. The gels are therefore considered independent from each other.

It is important to indicate that listing related gels for a protein is not, in theory, limited to the gels in which the protein has been physically identified (or has been explicitly matched by gel comparison). The list should also include all the gels merged with the reference gels that are associated with the protein. As we will see hereafter, the spots in which a protein has been physically identified (or matched) are managed by another class, the *SpotEntry* class. We must ensure at least that, for a protein, the *EntryGelImage* relation includes all the gels that contain the spots listed in *SpotEntry* and in which the protein has been either identified or matched. This is ensured by using an external method (implemented outside the RDBMS), and that we represent here by the *proteinIdentifiedOnGel()* method responsible for checking that all such gels are included in the *EntryGelImage* gel list. Another external method, the *gelsOnEntryFromSameOrganism()*, verifies an additional condition that requires all gels listed for a specific protein to belong to the same organism (a condition that cannot be directly expressed in an efficient relational model).

E.V.4 Spots: identity and physical properties

The spots identity is defined using a spot identifier (*spotID*) that is unique over a specific gel. Therefore, the formal spot identifier is a combination of both this *spotID* attribute and the gel identifier (*gelID*). This is the relation's primary key. The spot is given a defined Mw value (the molecular weight, given in dalton) and a pI value (the isoelectric point, in the appropriate range). For non-identified spots, the tool does not require a defined value for Mw and pI. By convention, and only for non-identified spots, Mw and pI are set to minus one (-1) when no values are provided by the user. The 'x' and 'y' coordinates of the spot, expressed in pixels, are reported relative to the origin situated at the upper left corner of the original analysed image of the gel. They are not concerned by any resizing or shifting of the original image (we have already seen that this is handled by the *GelImage* class). Theses coordinates typically define the gravity centre of the spot shape or its density distribution. Optionally, the Spot class also stores the values of the relative optical density and the relative volume of the spot. These estimated values are expressed as a percentage of the total value of all the spots detected over the gel. Relative optical density and volume are typically evaluated using 2-DE analysis software.

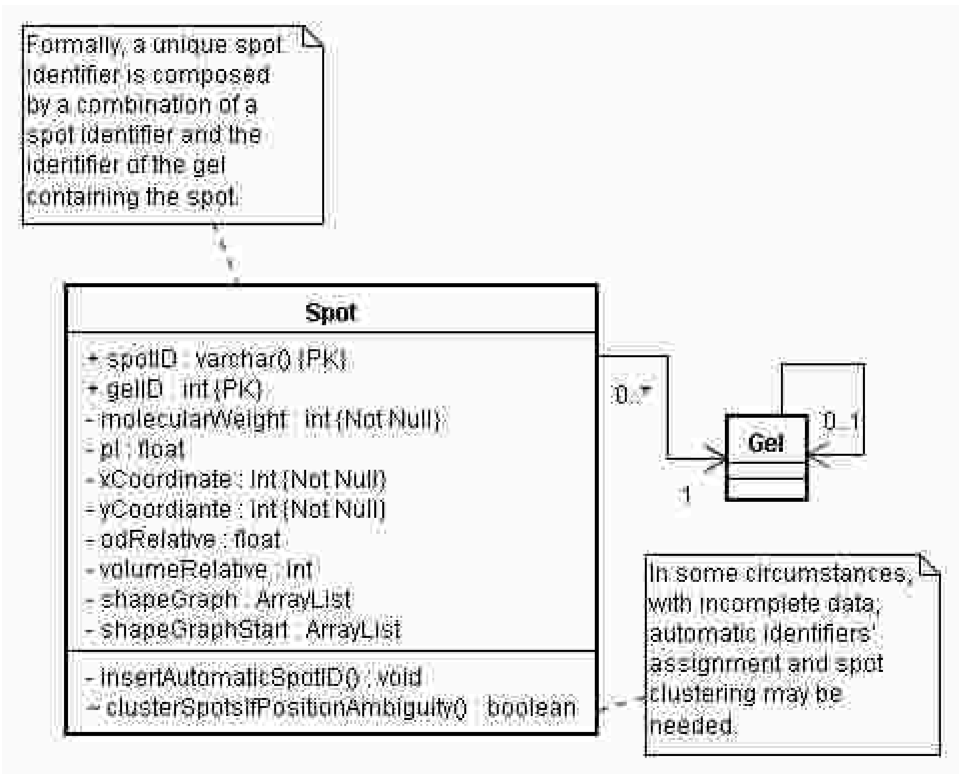

Formally, a unique spot identifier is composed by a combination of a spot identifier and the identifier of the gel containing the spot.
Spot
+ spotID : varchar() {PK}
+ gelID : int {PK}
- molecularWeight : int {Not Null}
- pI : float
- xCoordinate : int {Not Null}
- yCoordiante : int {Not Null}
- odRelative : float
- volumeRelative : int
- shapeGraph : ArrayList
- shapeGraphStart : ArrayList
- insertAutomaticSpotID() : void
- clusterSpotsIfPositionAmbiguity() : boolean
0..*
1
Gel
0..1
In some circumstances, with incomplete data; automatic identifiers' assignment and spot clustering may be needed.

Figure E.V-12: Data Model – The spot identity.

Spots should not overlap, which means that two spots belonging to the same gel should never share the same coordinates; they should neither share the same molecular weight and pI combination. *shapeGraph* is an optional list of integers that portrays the shape of a spot using a simple direction-driven vectorial representation[1] (relatively similar to the principle used in the Melanie / ImageMaster^{TM} 2D Platinum software). The initial point's coordinates are given by the *shpaeGraphStart* attribute. Storing precisely the computer-detected shape of a spot is not mandatory for the purpose of the majority of 2-DE databases. This may inflate the database storage size and slow down the process of visualisation, which explains why we disable by default shape's storage. As a result, spots are simply highlighted in the gel viewer using a tiny cross over their gravity centre[2].

Methods

During the EBP project development phase of the tool, we have been confronted with some datasets that lacked precise spot identifiers, *e.g.*, PHCI-2DPAGE. This database does not explicitly define specific spot identifiers in its distributable flat file format (in this case, the combination of the given pI/Mw values may act as a simulated spot identifier). Besides, some of these datasets could only provide, in addition to the flat file entries, a separated list of spot coordinates by protein entry without establishing any correspondence between these spots and those listed with their pI/Mw values in the protein entry. The external method *clusterSpotsIfPositionAmbiguity()* has therefore been developed to establish a correspondance between spot positions and pI/Mw values using a simple clustering algorithm[3]. This method was also used in some early versions of the tool when we were allowing direct data extraction from the binary annotated TIFF images produced by the Melanie software version 2 and 3, as the extraction process we were using was causing the same problem for spot correspondence. We have finally decided to deactivate this clustering method by default, thus encouraging users to systematically provide precise data. We also abandoned direct data extraction from the binary annotated images to avoid any potential inaccuracy.

The reason for tolerating such data incompleteness was essentially to maximise the chances for legacy data to be integrated into our new representation. This is especially important for datasets that are incomplete and that are no longer maintained by their producers, but still are of interest.

As for the second method, it is used to keep up with some SWISS-2DPAGE specifity. Historically, SWISS-2DPAGE had opted to randomly generate a spot identifier of the form '1/2D-*nnnnnnn*' (*e.g.*, 2D-0017PD) for each detected spot. This identifier was unique all across the database rather than being unique for an individual

[1] We propose to use values ranging from 0 to 7 and pointing to one of the 8 adjacent pixels surrounding the current position. Zero points to the North and rotation is clockwise.

[2] In order to highlight the whole shape of a spot, a C module had been previously developed by our group for the initial make2ddb tool to work along with old versions of the Melanie software. This module can be adapted to work with the new tool by reading shape values stored directly within the database. As we are concerned with our tool's portability and easiness to be installed by non-expert users, it would be preferable to first transcript the module into an interpreted language (Perl), thus avoiding system-dependant code compilation.

[3] *cf.* http://mordor.isb-sib.ch/make2ddb/lib2d/make2db_ASCII_TABLES.pm, under the "CLUSTER:" section

gel. We therefore included the optional *insertAutomaticSpotID()* method that automatically generate unique identifiers of the same previous SWISS-2DPAGE form for all detected spots.

E.V.5 Identifications

The spot identification sub-model (Figure E.V-13) links the spot entities to the identified proteins by means of the *SpotEntry* association class. As already mentioned, a protein entry is an instance of the *Entry* class, thus a *SpotEntry* instance associates a specific spot with a distinct protein. We also employ the term "mapping", a synonym for "identification" that is traditionally used in SWISS-2DPAGE.

The same protein may be found in several spots, and several proteins may be identified in the same spot. Two related subsystems[1] include the experiments performed on the spots, the experimental data, and the analysis and interpretation of this data for protein assignment. We will call them the "*Experimental Data*" and the "*Spot Identification*" subsystems. These two subsystems cover some predefined identification techniques and follow a definite structure.

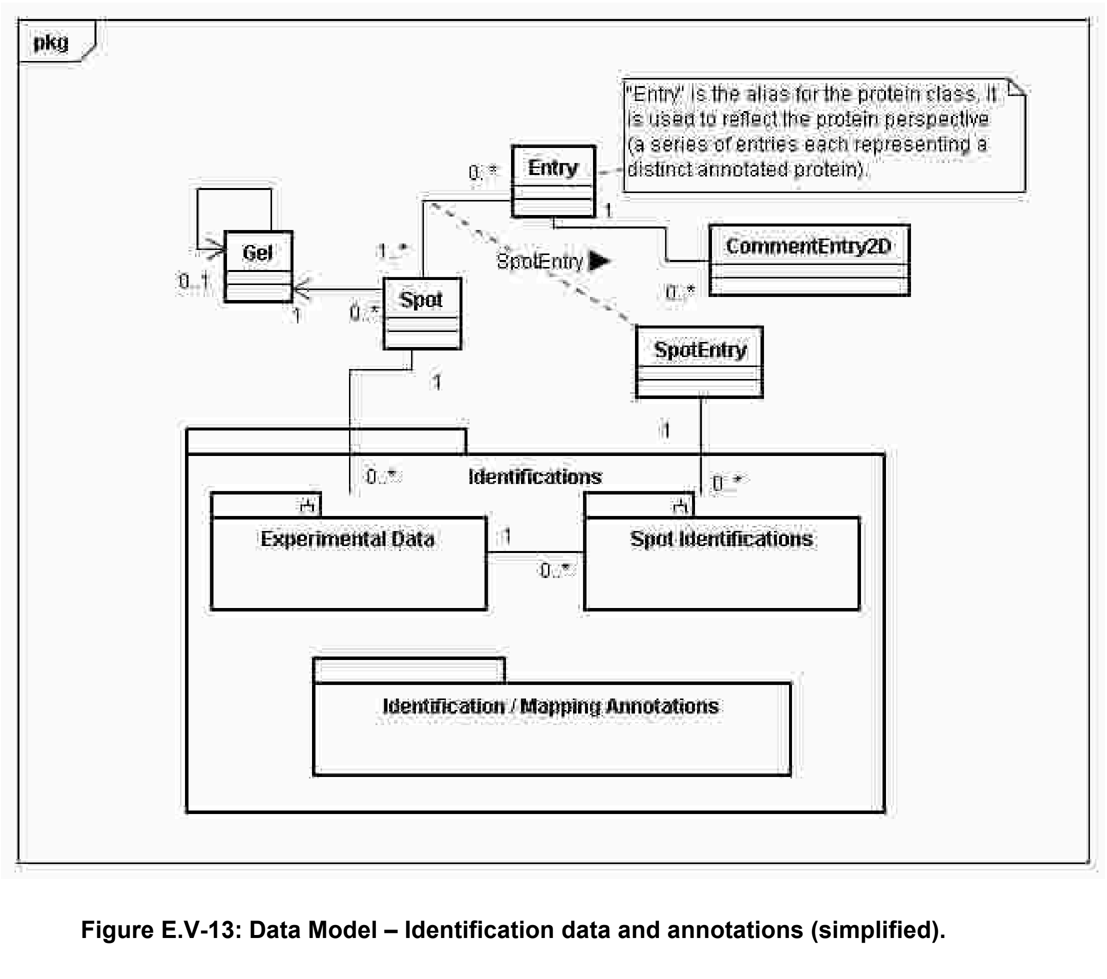

pkg
"Entry" is the alias for the protein class. It is used to reflect the protein perspective (a series of entries each representing a distinct annotated protein).
0..*
Entry
1
Gel
0..1
1
1..*
SpotEntry ▶
CommentEntry2D
0..*
0..*
Spot
1
SpotEntry
1
0..*
Identifications
0..*
Experimental Data
1
0..*
Spot Identifications
Identification / Mapping Annotations

Figure E.V-13: Data Model – Identification data and annotations (simplified).

[1] A subsystem is a category grouping several elements that have some common features, also known as a "classifier".

In addition to the predefined identification techniques, extra identification methods, as well as any related free text annotation, description or observation, are all grouped schematically in the "*Identification / Mapping Annotations*" sub-package.

Expanding the previous diagram reveals the following details (Figure E.V-14):

- The two subsystems mentioned above obey an inheritance structure. The classes at the very top are the two superclasses *SpotDataParent* and *SpotIdentificationParent*. Their children classes, or subclasses, are respectively *SpotDataChild* and *SpotIdentificationChild*. Each *SpotIdentificationChild* instance refers to a unique *SpotDataChild* instance, and results in the assignment of a specific protein to the analysed spot. A *SpotDataChild* instance may lead to different identifications, resulting in one or several distinct protein assignments for the same spot.

- Annotations are free text and are classified into user-defined topics. They are either general annotations and observations, or very specific identification / mapping annotations. The latter cover all identification techniques included in the predefined identification subsystems, as well as all additional user-defined identification methods not belonging to the identification subsystems. A *SpotEntry* instance may have several independent annotations.

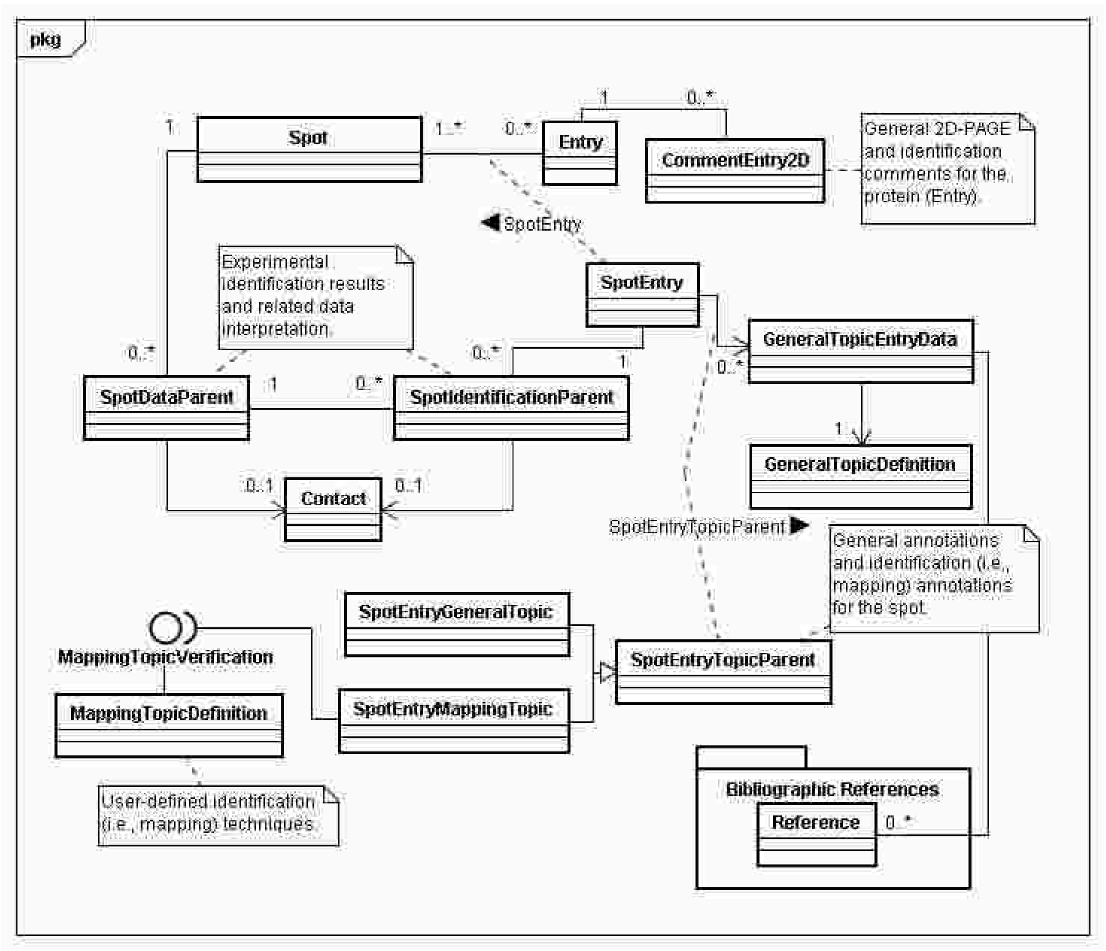

pkg
Spot
Entry
CommentEntry2D
General 2D-PAGE and identification comments for the protein (Entry)
SpotEntry
Experimental identification results and related data interpretation
SpotDataParent
SpotIdentificationParent
GeneralTopicEntryData
GeneralTopicDefinition
Contact
SpotEntryTopicParent
General annotations and identification (i.e., mapping) annotations for the spot
SpotEntryGeneralTopic
MappingTopicVerification
MappingTopicDefinition
SpotEntryMappingTopic
Bibliographic References
Reference
User-defined identification (i.e., mapping) techniques

Figure E.V-14: Data Model – Identification data and annotations (details)

The SpotEntry association class and the 2D comments

The *SpotEntry* class associates spots with their corresponding entries (proteins). As such, it includes a mechanism to show or hide any object from public end-users. In this case, the object to hide is the association between the spot and the protein. The mechanism is similar to the one implemented with the *Gel* class as described earlier. The *fragment* attribute tells whether the amino acid sequence identified on the spot is a protein fragment or an entire protein (Figure E.V-15).

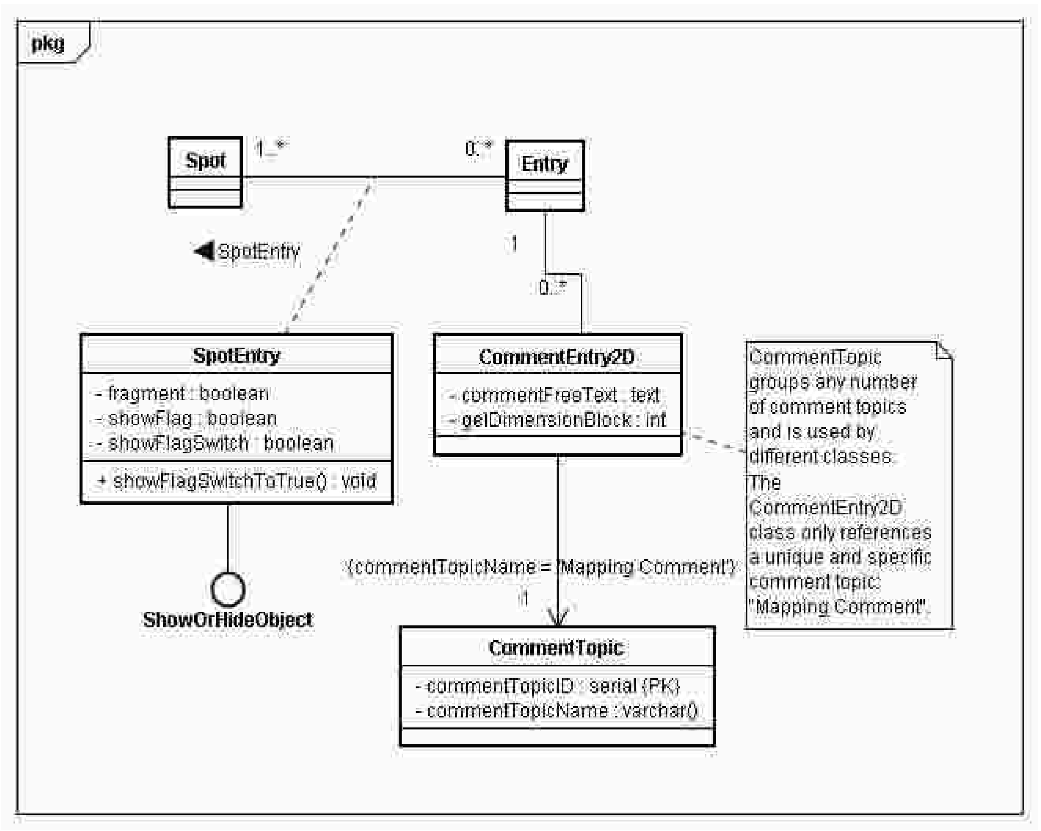

pkg
Spot
1..*
0..*
Entry
SpotEntry
1
0..*
SpotEntry
- fragment : boolean
- showFlag : boolean
- showFlagSwitch : boolean
+ showFlagSwitchToTrue() : void
CommentEntry2D
- commentFreeText : text
- gelDimensionBlock : int
CommentTopic groups any number of comment topics and is used by different classes. The CommentEntry2D class only references a unique and specific comment topic: "Mapping Comment".
{commentTopicName = 'Mapping Comment'}
1
ShowOrHideObject
CommentTopic
- commentTopicID : serial {PK}
- commentTopicName : varchar()

Figure E.V-15: Data Model – The *SpotEntry* association class and the 2D comments.

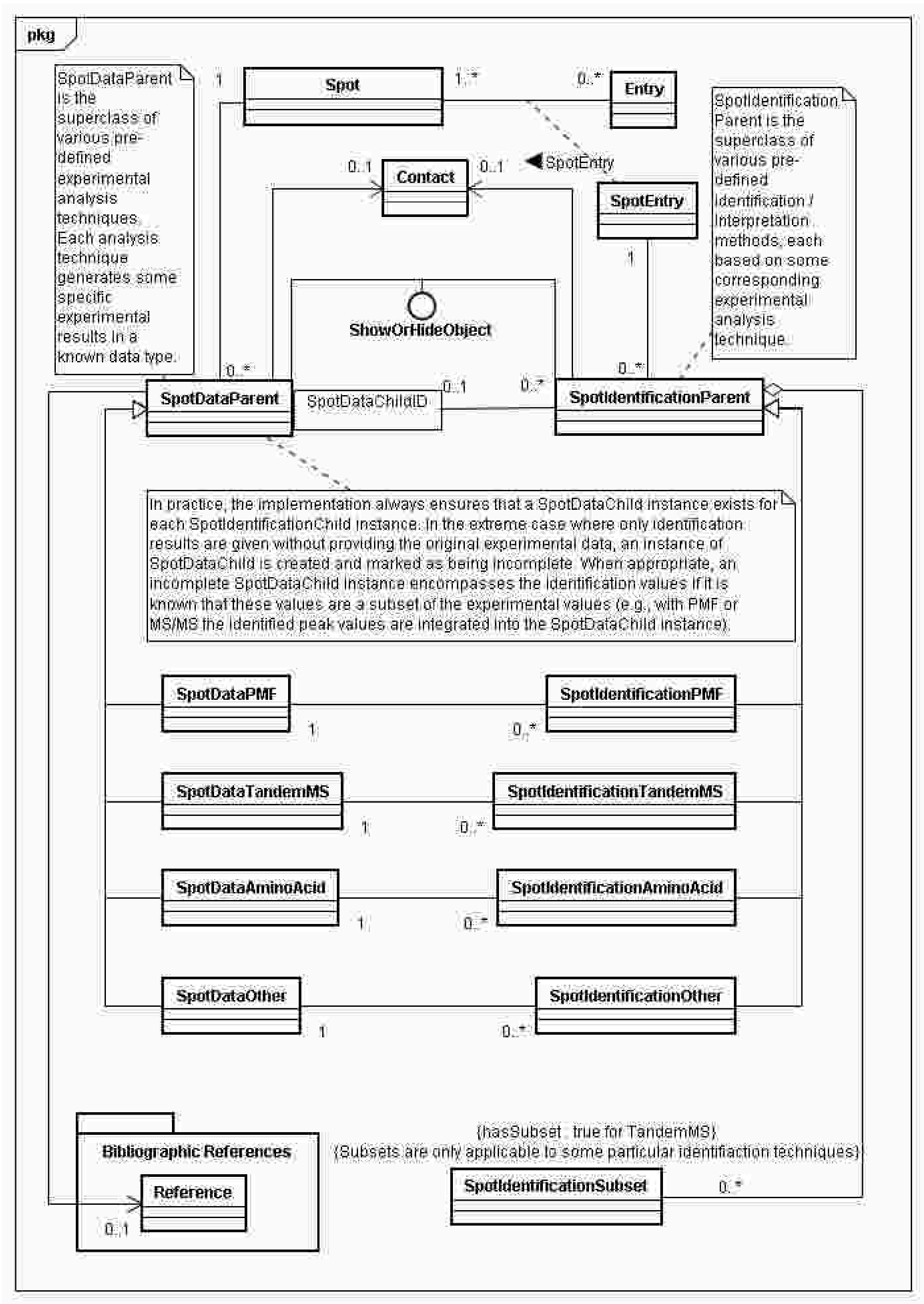
An entry may have some free text comments regarding the entire set of reference maps available for the protein (*e.g.*, cross-species identification comments). By convention, and like with SWISS-2DPAGE, these comments are grouped under the same topic keyword: "*Mapping Comment*"[1]. Consequently, *CommentEntry2D* only points to a unique *CommentTopic* instance[2].

The predefined identification subsystems

In Figure E.V-16 we present the various identification techniques that have been chosen as subclasses of *SpotDataParent* and *SpotIdentificationParent*. For each *SpotDataChild* sublass, a corresponding *SpotIdentificationChild* subclass exists in a one-to-many relationship. This means that any experiment / analysis data that is contained in a *SpotDataChild* instance may produce one or several different identifications (or interpretations), represented each by a *SpotIdentificationChild* instance, each in its turn leading to a distinct protein assignment for the spot. Consequently, each experiment indirectly assigns one or several proteins to a specific spot.

[1] *cf.* http://www.expasy.org/ch2d/manch2d.html#Heading28

[2] In addition to "Mapping Comment", the *CommentTopic* class includes a list of all user-defined protein-related comment topics, like the comment topics that are used in UniProtKB (http://www.expasy.org/sprot/userman.html#CC_line).



SpotDataParent is the superclass of various pre-defined experimental analysis techniques. Each analysis technique generates some specific experimental results in a known data type.

SpotIdentification Parent is the superclass of various pre-defined identification / interpretation methods, each based on some corresponding experimental analysis technique

In practice, the implementation always ensures that a SpotDataChild instance exists for each SpotIdentificationChild instance. In the extreme case where only identification results are given without providing the original experimental data, an instance of SpotDataChild is created and marked as being incomplete. When appropriate, an incomplete SpotDataChild instance encompasses the identification values if it is known that these values are a subset of the experimental values (e.g., with PMF or MS/MS the identified peak values are integrated into the SpotDataChild instance).

Figure E.V-16: Data Model –Subsystems of the predefined identification methods.

There are currently three predefined identification techniques selected in the model:

- **Peptide Mass Fingerprinting** (PMF)
- **Tandem mass spectrometry** (MS/MS)

- **Amino acid composition** (AAcid)

A generic unspecified technique (*Other*) is also part of the subsystems. The corresponding subclasses are intended to serve as a prototype for any additional identification technique that we may want to include in the future within our model.

To identify a protein we typically use experimental data. The experimental data is managed in our model by the *SpotDataChild* subclasses. Either the data is used entirely, or only a subset of this data is retained to predict the protein. An amino acid composition experiment provides a set of data that is generally entirely used in the protein assignment, while a mass spectrometry analysis provides a set of peak list values that may be only partially retained for the identification. In both cases, the data used or retained for the prediction is reported in the *SpotIdentificationChild* subclasses.

A predefined identification class may or may not have an annexed class that extends the interpretation of the data retained for the identification. In some techniques, we need to describe the identification with a presentation that is not based on the entire retained data (as a whole), but based on separate subsets of this data. Each subset is therefore an element that contributes to the final interpretation and to the assignment of the protein. This is typically the case in tandem MS De novo sequencing, where sequence tags are deduced from distinctive subsets of the retained peak values. *SpotIdentificationSubset* is the class aggregated to *SpotIdentificationChild*, with the condition that *Child* is *TandemMS*.

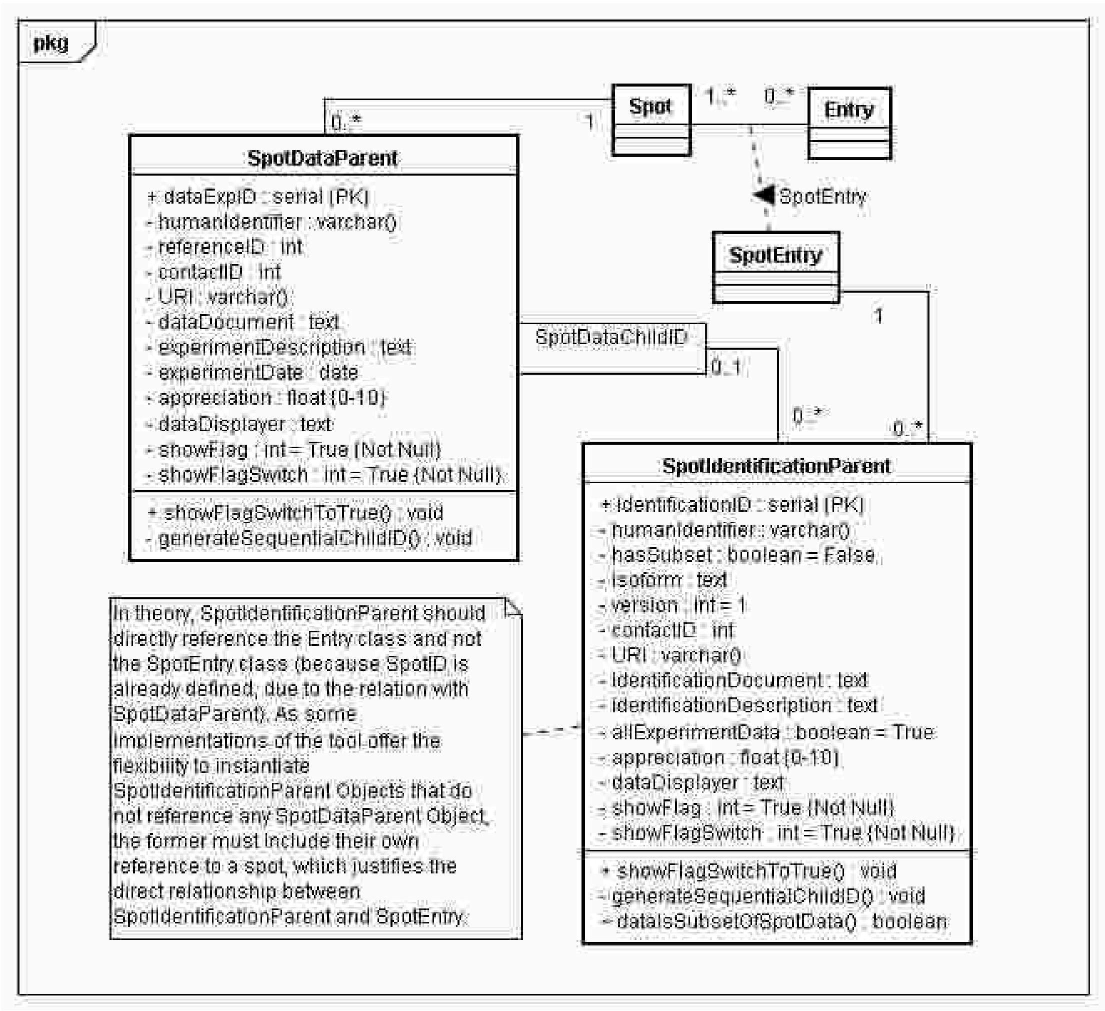
As already mentioned, the data retained for the identification is either the entire experimental data or only a subset of this experimental data. A set of implemented methods must therefore ensure that this inclusion condition is verified. We must also deal with incomplete datasets that do not provide all experimental data, but only the data retained for the identification (*e.g.*, when all the masses of a PMF spectrum is not made available, but only those assigned to a peptide). In our model, any *SpotIdentificationChild* instance ideally refers to an experimental *SpotDataChild* instance. In the absence of the latter, we may create it, mark it incomplete, and populate it with the data associated with the identification.

Like many other objects in our model, any experimental and identification data can be hidden from public access (the *ShowOrHideObject* interface). This also implies hiding an association of a spot with a protein. Any experiment or identification is also optionally linked with a bibliographic reference, as well as with a contact person (*e.g.*, the experiment performer).

The parent superclasses

Some implementations of the tool, especially the most recent ones, offer additional flexibility by allowing the instantiation of *SpotIdentificationChild* objects that do not necessarily reference any *SpotDataChild* object. This concerns datasets that do not contain all the experimental data. We may then avoid generating incomplete instances of *SpotDataChild*. To express this flexibility, we have shown a multiplicity of ‘0..1’ between *SpotIdentificationParent* and *SpotDataParent*. (Figure E.V-17).

When an identification does not have an associated *SpotDataChild* object, there is a need to explicitly define the spot identifier within the identification class itself. This also explains why the *SpotIdentificationParent* has a direct relationship with *SpotEntry* (spot association with proteins) rather than with the *Entry* class.


pkg
Spot
Entry
1..*
0..*
0..*
1
SpotEntry
SpotEntry
1
SpotDataParent
+ dataExpID : serial {PK}
- humanIdentifier : varchar()
- referenceID : int
- contactID : int
- URI : varchar()
- dataDocument : text
- experimentDescription : text
- experimentDate : date
- appreciation : float {0-10}
- dataDisplayer : text
- showFlag : int = True {Not Null}
- showFlagSwitch : int = True {Not Null}
+ showFlagSwitchToTrue() : void
- generateSequentialChildID() : void
SpotDataChildID
0..1
0..*
0..*
SpotIdentificationParent
+ identificationID : serial {PK}
- humanIdentifier : varchar()
- hasSubset : boolean = False
- isoform : text
- version : int = 1
- contactID : int
- URI : varchar()
- identificationDocument : text
- identificationDescription : text
- allExperimentData : boolean = True
- appreciation : float {0-10}
- dataDisplayer : text
- showFlag : int = True {Not Null}
- showFlagSwitch : int = True {Not Null}
+ showFlagSwitchToTrue() : void
- generateSequentialChildID() : void
- dataIsSubsetOfSpotData() : boolean

In theory, SpotIdentificationParent should directly reference the Entry class and not the SpotEntry class (because SpotID is already defined, due to the relation with SpotDataParent). As some implementations of the tool offer the flexibility to instantiate SpotIdentificationParent Objects that do not reference any SpotDataParent Object, the former must include their own reference to a spot, which justifies the direct relationship between SpotIdentificationParent and SpotEntry.

Figure E.V-17: Data Model – Parent superclasses of the identification subsystems.

Both superclasses have an inner method – *generateSequentialChildID()* - that produces sequential identifiers for their respective subclasses. This generates distinct identifiers across all the subclasses[1], which is more convenient than having similar identifiers shared between the different experiments and identifications. Having a unique identifier across all subclasses allows the interrogation of the superclass without explicit specification of which identification technique is involved.

The *humanIdentifier* attribute is an identifier given by experimentalists and data analysers to respectively identify (or regroup) their experiments or their identifications.

[1] In postgreSQL, some proprieties are not inherited, including uniqueness of attributes. For an attribute to be distinct across all the subclasses of a superclass, we must implement a method (trigger) that ensures the attribute is unique.

It can be a mnemonic word, a file name, or a folder name holding the data. It is different from the database-generated identifier and is not unique. For example, an MS/MS output file may contain a set of spectra, each having its own *SpotDataTandemMS* instance, but sharing all the same experiment *humanIdentifier*. Similarly, one identification analysis may reveal several identified proteins, thus several instances of *SpotIdentificationChild* become necessary. All the instances can then share the same *humanIdentifier* while having each a distinct *identificationID*. The *humanIdentifier* attribute is optional, as such identifier is not systematically given by users.

Both superclasses have attributes for a short description of how their procedure was performed, and a Web location where the experimental data or the identification details are accessible. Alternatively to a Web location, a local document is given by its file system path (*e.g.*, a local PSI::mzData or a PSI::AnalysisXML file). An appreciation of the work is suggested. The appreciation is a very simple way to evaluate one's work, and is arbitrarily defined as a numerical value ranging from zero to ten. Although this is only a personal and a relative appreciation, it may help to evaluate and compare the quality of different experiments and identifications within a small group of people. The *appreciation* attribute could lead to defining a more sophisticated manner of evaluating the quality of an identification process in subsequent versions of the model.

A data displayer is an interface, often given by its Web location, which can display the experiment data or the identification report. The displayer depends on the nature of the identification technique. It may be sent as data input a SQL command, a file location, or a set of GET/POST attributes/values. We may even add an additional *dataDisplayerParameter()* method to formulate the displayer input parameters independently from any external code.

In *SpotIdentificationParent*, the *allExperiementData* attribute tells whether the identification process uses all the relevant data provided by the experiment or only a part of it. The external method *dataIsSubsetOfSpotData()* verifies that any data retained for the identification is part of the experimental data[1]. The class also holds two additional attributes, *isoform* and *version*. We adopted the same definition of a protein entry like in UniProtKB. This means that a protein entry in our model represents the protein and all its isoforms, including its variants and varsplices. An identification may match a specific isoform, which is indicated by the *isoform* attribute. Furthermore, we adopted a version notation for entries. A protein entry that has been modified between two consecutive releases of a database sees its version number raised by one. The *version* attribute in the identification classes gives the entry version at the time the identification was integrated into the database. This attribute may also serve to assign different versions to a specific identification (for example, by reviewing a previous interpretation).

Due to the kind of data we have to deal with (SWISS-2DPAGE and similar databases), it was sufficient to define experiments and link them directly with spots in a unique class. Nevertheless, it is generally more convenient to define experiments

[1] We may ideally transform this external method into an internal method.

independently from spots (in an *Experiment* class), and use a *SpotDataParentExperiement* association class between *Spot* and *Experiement*, rather than the somehow potentially redundant *SpotDataParent* class (Figure E.V-18).

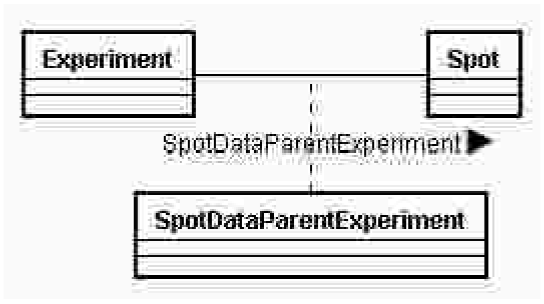


Figure E.V-18: Data Model – SpotData as an association class between Spot and Experiement.

Amino acid composition

Amino acid composition is stored in a list of (ordered) one-letter amino acid codes. Each amino acid gives its composition percentage in the analysed protein:

A=12.03, B=4.12, F=1.03, G=5.84, H=0.69, I=0.69, K=4.12, L=12.71, M=2.41, P=2.75, R=11.68, S=4.81, T=3.78, V=7.56, Y=1.37, Z=24.40

We have chosen to store such data in a row-text string (Figure E.V-19), with no special verification of its content. Another option would have been to store this data in a two-dimensional matrix, but we estimated that no significant overall profit would be gained this way. *relatedData* may be any additional information directly related to the experiment data or the identification process, and *dataIsEntireSpotData*() is an external method, analogous to *dataIsSubsetOfSpotData()* and supplanting it. It verifies that the amino acid list in *SpotIdentificationAminoAcid* is identical to the one in *SpotDataAminoAcid*, and not only a subset of it

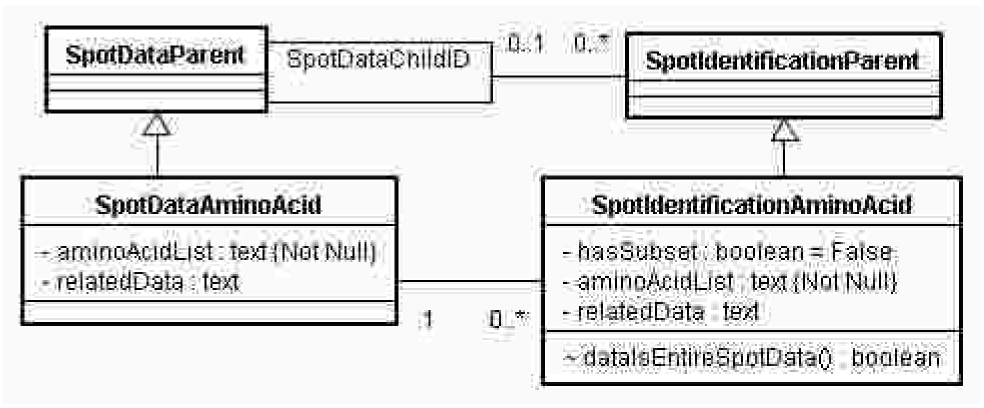

SpotDataParent
SpotDataChildID
0..1
0..*
SpotIdentificationParent
SpotDataAminoAcid
- aminoAcidList : text {Not Null}
- relatedData : text
1
0..*
SpotIdentificationAminoAcid
- hasSubset : boolean = False
- aminoAcidList : text {Not Null}
- relatedData : text
~ dataIsEntireSpotData() : boolean

Figure E.V-19: Data Model – Amino acid composition subclasses.

Peptide Mass Fingerprinting

The peptide peak list values of a spectrum are stored in *peptideMasses*. This attribute is a two-dimensional matrix holding the masses and their intensities. In *SpotDataPMF*[1], the original *relatedData* attribute is labelled *enzyme* and identifies the digestion enzyme. In *SpotIdentificationPMF*[2], only the peptide masses that have been retained for the identification are stored in the *peptideMasses* attribute[3].

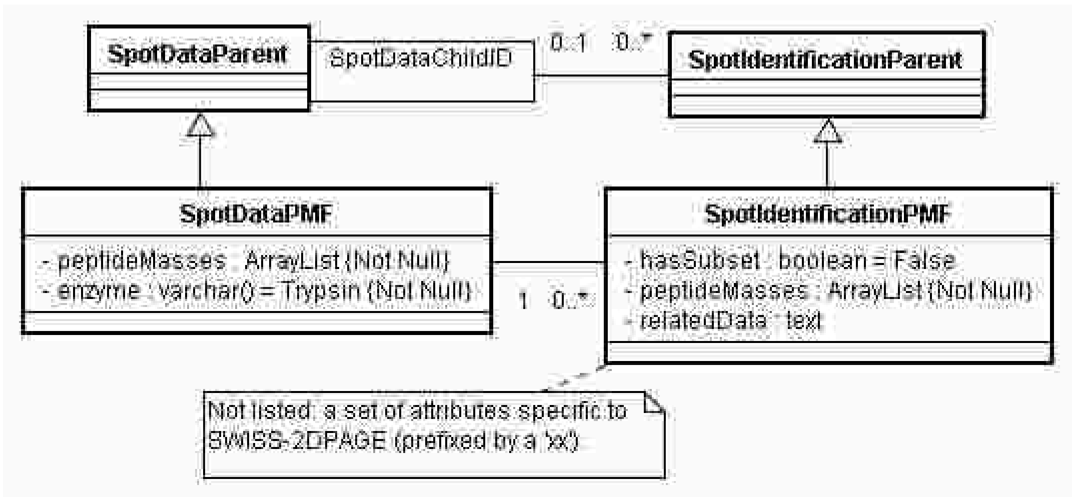

SpotDataParent
SpotDataChildID
0..1
0..*
SpotIdentificationParent
SpotDataPMF
- peptideMasses : ArrayList {Not Null}
- enzyme : varchar() = Trypsin {Not Null}
1
0..*
SpotIdentificationPMF
- hasSubset : boolean = False
- peptideMasses : ArrayList {Not Null}
- relatedData : text
Not listed: a set of attributes specific to SWISS-2DPAGE (prefixed by a 'xx')

Figure E.V-20: Data Model – Peptide Finger Printing subclasses.

Tandem MS

Like in PMF, the ion masses are stored in a two-dimensional matrix. The parent mass and its charge are stored in *SpotDataTandemMS*. Tandem MS is currently the only identification technique in the model that has an identification "subset" class. *SpotIdentificationTandemMSSubset* is a class that covers the peptide sequences' identifications. An instance is created for each peptide sequence that has been deduced from the spectrum[4].

By analogy with *dataIsSubsetOfSpotData()*, the *ionMassesAreSubsetOfSpotIdentificationTandemMS()* method verifies that the ion masses stored in *SpotIdentificationTandemMSSubset* are a subset of those stored in *SpotIdentificationTandemMS*. Whenever some ion masses relevant to the prediction of the peptide sequences are not listed in *SpotIdentificationTandemMS*, the same method duplicates them within this class.

[1] The implemented relation is labelled *SpotDataPeptMassF* rather than *SpotDataPMF*.

[2] The implemented relation is labelled *SpotIdentificationPeptMassF* rather than *SpotIdentificationPMF*.

[3] *SpotIdentificationPMF* also includes some extra attributes, specific to SWISS-2DPAGE, and prefixed with a double 'x' (xxAc, xxDirectory, xxFile and xxProgVersion).

[4] Peptide fragment fingerprinting does not need to use the peptide sequences subset class.

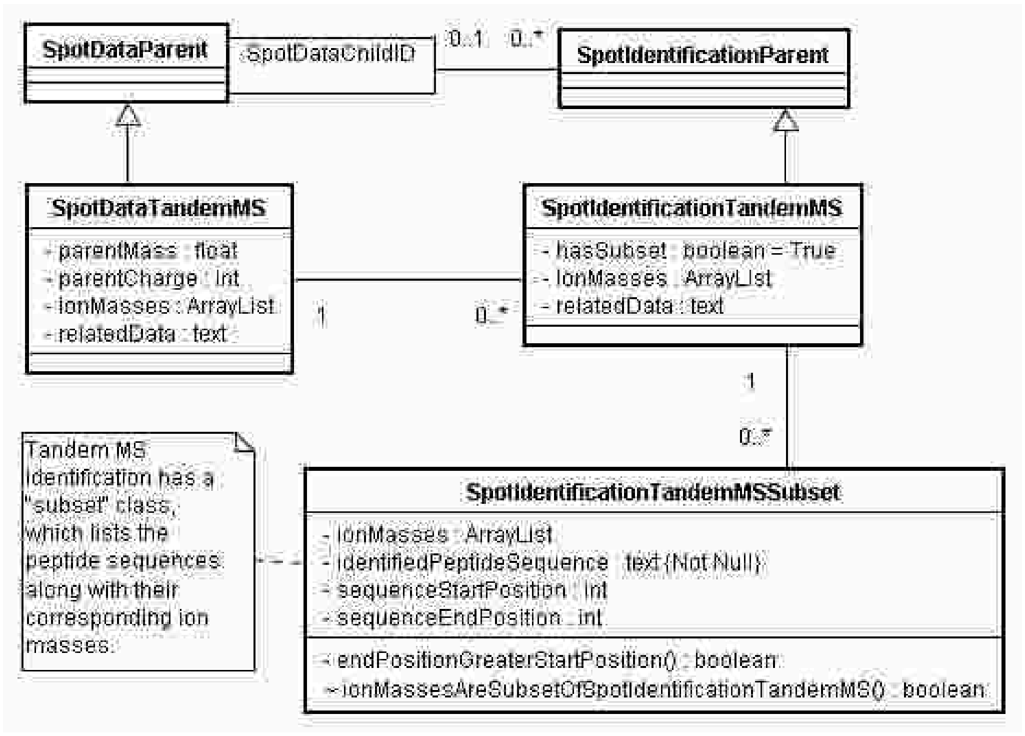

SpotDataParent
SpotDataChildID
0..1
0..*
SpotIdentificationParent
SpotDataTandemMS
- parentMass : float
- parentCharge : int
- ionMasses : ArrayList
- relatedData : text
1
0..*
SpotIdentificationTandemMS
- hasSubset : boolean = True
- ionMasses : ArrayList
- relatedData : text
1
0..*
Tandem MS identification has a "subset" class, which lists the peptide sequences along with their corresponding ion masses.
SpotIdentificationTandemMSSubset
- ionMasses : ArrayList
- identifiedPeptideSequence : text {Not Null}
- sequenceStartPosition : int
- sequenceEndPosition : int
- endPositionGreaterStartPosition() : boolean
~ ionMassesAreSubsetOfSpotIdentificationTandemMS() : boolean

Figure E.V-21: Data Model – Tandem MS subclasses.

Comments on the mass spectrometry subclasses

As previously mentioned, users may provide their mass spectrometry data by giving an external link to a Web address or to any repository where their files have been deposited (*e.g.*, PRIDE). In this case, the PMF subclasses will still require a short list of the identified peptide masses, while the tandem MS subclasses will not. If the user points to a local file containing the peak lists in some common format (mzData, mzXML, pkl, mgf, etc.), or if he/she directly provides the peptide or the ion masses and their intensities in the input files (text or CSV), the values will be automatically extracted and populate the mass spectrometry subclasses. This is typically convenient if the user wishes to display and visualise his/her peak lists within the 2-DE database. Users may even use both options, *i.e.*, provide external links as well as local files that include their peak lists.

As opposed to PMF, the tandem MS *ionMasses* attribute is optional. This is to give users the opportunity to indicate their identified peptide sequences even if they do not strictly provide a list of ion masses. It is then highly recommended to give at least the location of the mass spectrometry files, in order to verify the authenticity of the identification by end-users or reviewers.

The generic 'Other' subclasses

These subclasses are not specific to any particular identification technique. As already stated, they serve as a prototype that can be refined in order to include any additional identification technique that might need to be included in the future.

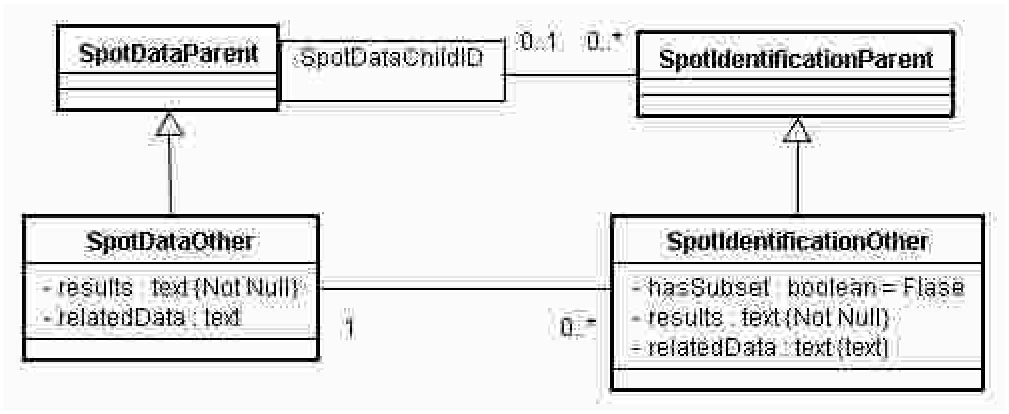


Figure E.V-22: Data Model – Generic identification technique subclasses (Other).

User-defined identification methods and free text annotations

We have already illustrated (Figure E.V-14) how user-defined identification methods and related free text annotations are implemented in our model. This section of the model is independent from the general comments regarding the overall identification process of all the maps available for an entry (the "Mapping Comment", as illustrated in Figure E.V-15).

Datasets that follow the recommendations of Make2D-DB II, *e.g.*, the recommendations for the spreadsheet mode[1], offer a precise data structure that helps users easily defining the topics and the spot assignments for their free text annotations. Nevertheless, to also integrate legacy and less structured datasets, the data model had to be adapted.

We have introduced the concept of 2-DE topics to be any category of annotation or observation related to a specific spot identification process. 2-DE Topics are defined by the user and are free text. They are stored in the *GeneralTopicDefinition* class. There is however one particular topic that is fixed among all tool implementations: the **MAPPING** topic. This topic groups all the various techniques used to identify the spots. It therefore covers all the already predefined techniques listed in the identification subsystems (PMF, MS/MS and AAcid), as well as any additional user-defined identification method. The MAPPING topic is automatically instantiated by the tool as the first object within the *GeneralTopicDefinition* class. Some other examples of user-defined topics are listed in Table E.V-1[2].

[1] http://world-2dpage.expasy.org/make2ddb/2.Readme_preparation.html#spreadsheetsMode

[2] Examples of user-defined free text topics from SWISS-2DPAGE.

Table E.V-1: Examples of user-defined 2-DE topics.

TOPIC	Description	Annotation example
MAPPING	Description of the technique that has allowed the identification of the spot.	Matching with a plasma gel.
NORMAL LEVEL	Description of the physiological protein expression.	30-60 MG/L.
PATHOLOGICAL LEVEL	Description of pathological protein expressions (an increase or decrease).	Increased during the acute-phase reaction; decreased during emphysema.
(NORMAL) POSITIONAL VARIANTS	Description of physiological polymorphisms.	30 genetics variants known as PI alleles.
(PATHOLOGICAL / DISEASE) POSITIONAL VARIANTS	Description of pathological polymorphisms.	Alpha-1-antitrypsin Pittsburgh.
EXPRESSION	Description of the protein expression modifications including level and/or post-translational modifications.	Decrease after benzoic acid treatment.

Any 2-DE annotation falls under one of the 2-DE topics. Annotations are free text data that are attached to some spots. They are all grouped in the *GeneralTopicEntryData* class with no direct reference to a particular spot, which is convenient as the same annotation may often be used to annotate several spots (Figure E.V-23).

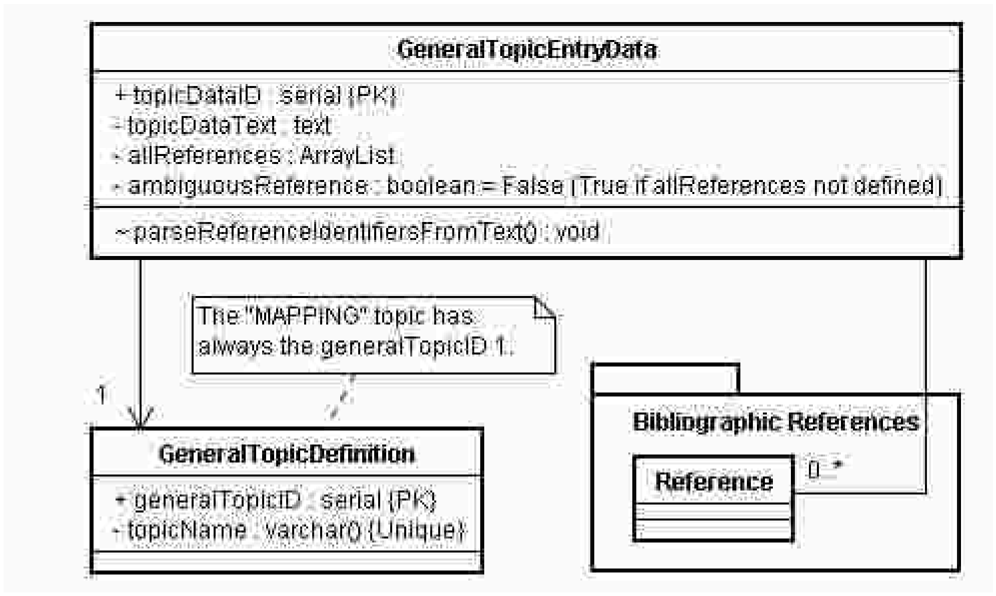

GeneralTopicEntryData
+topicDataID : serial {PK}
-topicDataText : text
-allReferences : ArrayList
-ambiguousReference : boolean = False (True if allReferences not defined)
~parseReferenceIdentifiersFromText() : void
The "MAPPING" topic has always the generalTopicID 1.
1
GeneralTopicDefinition
+generalTopicID : serial {PK}
-topicName : varchar() {Unique}
Bibliographic References
Reference
0..*

Figure E.V-23: Data Model – The spot annotation general topics.

Topics, which are listed in *GeneralTopicDefinition*, are referenced by the *GeneralTopicEntryData* class that includes all the annotations. In addition to the *topicDataText* attribute that stores individually the annotation texts, a mechanism that relates each specific annotation to the bibliographic references is managed by the *parseReferenceIdentifiersFromText()* external method. At this stage, we must give some insights into how bibliographic references are expressed within the protein entries. Like in SWISS-2DPAGE[1], bibliographic references related to an entry are listed sequentially within the entry, each having a specific local numerical identifier (the '*RN*' line). One or more reference identifiers can be integrated within the annotation text to inform about the source of this annotation. By convention, we integrate the bibliographic reference local identifiers within the annotation text using brackets, *e.g.*, "*some text* [1][3]"[2]. *parseReferenceIdentifiersFromText()* parses consequently the annotation text to extract these identifiers and establishes the relation with the "Bibliographic References" package (more details will be given hereafter on how this package operates). The *allReferences* attribute is the list that contains the extracted identifiers. When only one unique bibliographic reference is attached to the entry, it is then assumed that the annotation is necessarily related to this reference, even if the annotation does not explicitly list any bibliographic reference. In case no reference is explicitly given in the annotation text, and that several references are included within the entry, then the annotation is marked as ambiguous using the *ambiguousReference* attribute. The user is given a detailed report by the tool listing which of his/her annotations are ambiguous, so he/she may decide to rewrite them, or to keep them marked as ambiguous.

Assigning annotations to spots

To establish a relationship between the list of annotations and the spots within the different entries we make use of the *SpotEntryTopicParent* association class between *SpotEntry* and *GeneralTopicEntryData* (as already shown in Figure E.V-14). *SpotEntryTopicParent* is the super class of two distinct subclasses, *SpotEntryMappingTopic* and *SpotEntryGeneralTopic*. The former references all the annotations recognised by the tool as being directly related to a mapping (identification) method, while the latter references all the other kinds of annotations and observations. All annotations referenced by *SpotEntryMappingTopic* are necessarily classified under the MAPPING topic (defined in the *GeneralTopicDefinition* class), while those referenced by *SpotEntryGeneralTopic* are classified under any other topic (*cf.* Table E.V-1). A variety of annotation examples, randomly chosen from SWISS-2DPAGE, is listed in Table E.V-2. Some of these annotations contain explicit bibliographic references as described in the previous section. Others contain indications about the spots to which the annotations apply.

Table E.V-2: Some annotations randomly chosen from SWISS-2DPAGE.

Mapping annotations	**General annotations**

[1] http://www.expasy.org/ch2d/manch2d.html#Heading14

[2] This convention is applied to any data input by users in both flat file and CSV mode, except that for the latter the user gives a list of all his bibliographic references grouped together (not related to any entry), gives an identifier to each reference, and uses this identifier in his annotation text. The tool applies then a transformation to the annotation text to locally renumber these identifiers within each entry.

Gel matching [1] and identified by Garrels [2]	Decrease after benzoic acid treatment [3]
Amino acid composition and sequence tag (AVVA)	Increased during pregnancy and iron deficiency
Microsequence analysis (XNSQXEXPVA)	More than 20 variants
Spot 2D-001B0W: Matching with the Mouse Liver master gel	900-4500 mg/L [1]
Spots 2D-00179*, 2D-0017BM: Peptide mass fingerprinting [1]	Has been observed as a very early event in the progression of colon carcinoma [2]

Legacy and unstructured annotations

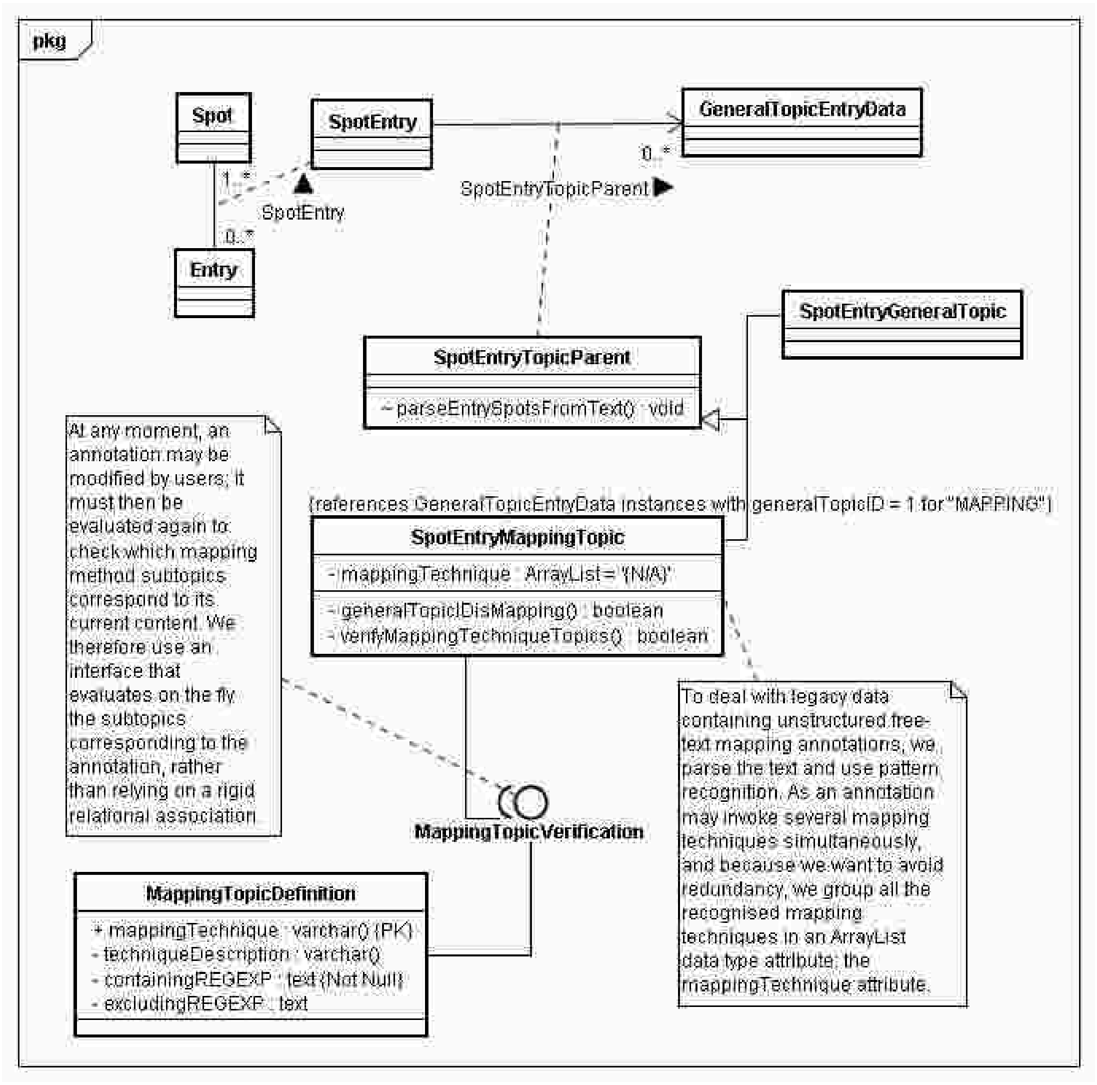
We have already expressed our concern for maximising the chances of integrating legacy datasets that are not fully structured into our new representation without needing to rewrite the data. To achieve this goal, we had to build a model that integrates these datasets as efficiently as it would do with ideally structured datasets. This aspect is particularly apparent in the previous section of the model that was built to accurately incorporate free text annotations from SWISS-2DPAGE and similar databases provided in flat file format. Besides building an accurate model, we also had the task of reading and interpreting free text annotations correctly (*cf.* annotations listed in Table E.V-2). By parsing such texts, we can establish which spots are concerned by the annotation, and the exact general topic or mapping topic category in question. We also had to consider the fact that many of such annotations are not systematically written separately from each other. We must therefore know exactly when a spot-specific annotation starts and when it ends.

By SWISS-2DPAGE convention, a free text annotation within a protein entry is gel or spot-specific, and is mostly written in the following way:

TOPIC: SPOT *spotID#1*: *some annotation*; SPOT *spotID#2*: *some annotation*;..

When no spots (or bands) are designated, the annotation is assumed to involve all the spots (or bands) of the protein over the annotated gel. Several spots separated by commas may share the same annotation. Moreover, abbreviations are frequently used, *e.g.*, from SWISS-2DPAGE: 2D-00179* (all spots starting with 2D-00179), 1WC* (all spots including 1WC, *e.g.*, 2D-001WCU). To deal with this kind of annotations, we have built an adapted parser[1] that is able to isolate the different annotations, recognise their topics, and extract the list of involved spots. The *parseEntrySpotsFromText()* method in Figure E.V-24 symbolises this external parser. The annotations themselves are never rewritten.

[1] *cf.* http://mordor.isb-sib.ch/make2ddb/lib2d/make2db_ASCII_TABLES.pm, the WRITE_2D subroutine of the make2db_ASCII_TABLES Perl package.


pkg
Spot
SpotEntry
GeneralTopicEntryData
1..*
0..*
SpotEntry
SpotEntryTopicParent
0..*
Entry
SpotEntryGeneralTopic
SpotEntryTopicParent
~ parseEntrySpotsFromText() : void
{references GeneralTopicEntryData instances with generalTopicID = 1 for "MAPPING"}
SpotEntryMappingTopic
- mappingTechnique : ArrayList = '{N/A}'
- generalTopicIDisMapping() : boolean
- verifyMappingTechniqueTopics() : boolean
MappingTopicVerification
MappingTopicDefinition
+ mappingTechnique : varchar() {PK}
- techniqueDescription : varchar()
- containingREGEXP : text {Not Null}
- excludingREGEXP : text

At any moment, an annotation may be modified by users; it must then be evaluated again to check which mapping method subtopics correspond to its current content. We therefore use an interface that evaluates on the fly the subtopics corresponding to the annotation, rather than relying on a rigid relational association.

To deal with legacy data containing unstructured free-text mapping annotations, we parse the text and use pattern recognition. As an annotation may invoke several mapping techniques simultaneously, and because we want to avoid redundancy, we group all the recognised mapping techniques in an ArrayList data type attribute: the mappingTechnique attribute.

Figure E.V-24: Data Model – spot general and mapping annotations.

The mapping annotations

The mapping annotations, covering the various methods used to identify the spots, and falling under the MAPPING topic, are managed differently from the general annotations. Besides having their dedicated *SpotEntryMappingTopic* subclass, which links the annotations stored in *GeneralTopicEntryData* with the spots, they integrate a mechanism to classify the mapping methods into subtopics.

The *generalTopicIDisMapping()* method in Figure E.V-24 ensures that the mapping annotations point to the MAPPING topic reference that is defined in *GeneralTopicDefinition*.

An autonomous class, *MappingTopicDefinition*, contains all the mapping subtopics. These subtopics include the predefined techniques listed in the identification subsystems and their subsets[1], in addition to the additional user-defined identification methods. The *mappingTechnique* attribute acts as an identifier and as an abbreviation for the identification method (*e.g.*, "PMF" for peptide fingerprinting).

The class includes three additional attributes: a short description of the identification method and, more importantly, two attributes, *containingREGEXP* and e*xcludingREGEXP*, which are in charge of assigning mapping subtopics to the different mapping annotations. *containingREGEXP* is a user-defined regular expression[2] that, when found in an annotation, associates the corresponding mapping subtopic to this annotation. e*xcludingREGEXP* removes this association if the regular expression given by this attribute is found within the annotation. Regular expressions are case-insensitive.

The tool proposes by default a pre-configured set of mapping topic definitions adjusted for SWISS-2DPAGE (Table E.V-3). These definitions can be easily altered or extended by users to comply with the syntax of their annotations. Besides, the procedure may also serve to map between personal annotations and any agreed-on controlled vocabulary that defines the names of the identification techniques.

Table E.V-3: Mapping topic definitions currently in use by SWISS-2DPAGE.

mapping Technique	techniqueDescription	containingREGEXP	excludingREGEXP
N/A	_Not_Defined_Method_	_Not_Defined_Method_	
Aa	Amino acid composition	Amino acid composition	
Co	Comigration	Comigration	
Gm	Gel matching	Matching|Identified on [1-2]-D	
Im	Immunobloting	Immuno	
MS/MS	Tandem mass spectrometry	Tandem mass spectrometry	
Mi	Microsequencing	Microseq|Internal sequence|Sequence tag|Tagging	
PMF	Peptide mass fingerprinting	Mass fingerprinting|Mass spectrometry|PMF	Tandem
PeptSeq	Peptide sequencing	Peptide sequencing	

An annotation may refer to several mapping techniques simultaneously. To avoid redundancy, we group all the recognised mapping methods subtopics in an ArrayList data type attribute: the array attribute in *SpotEntryMappingTopic*. For example, Table E.V-2 contains a mixed annotation: "Amino acid composition and sequence tag (AVVA)". Based on the mapping topic definitions given in Table E.V-3, the *mappingTechnique* array attribute will be "{Aa, Mi}".

[1] Peptide sequencing has therefore its own mapping subtopic that is independent from tandem MS.

[2] http://en.wikipedia.org/wiki/Regular_expression

Note that we did not purposefully set up a relational association between the *SpotEntryMappingTopic* and the *MappinTopicDefinition* classes via the *mappingTechnique* attribute for two reasons. The first one is purely technical. A RDBMS does not allow a direct referencing between an ArrayList data type and an atomic data type (dimensional data type incompatibility). The second is a practical reason. An annotation may be modified by users at any moment. The modified content must be evaluated to define the corresponding mapping method subtopics. For these reasons, we use the *MappingTopicVerification* interface that evaluates on the fly which mapping subtopics correspond to the current content of the annotation. Such an operation is impossible with a rigid relational association.

E.V.6 The protein annotations and related classes

We have so far extended the representation of three distinct entities - or objects – central to our data model: the gels, the spots and the identifications. The fourth central entity in our model is the protein. Identified proteins are defined by the *Entry* class. In a 2-DE model where data is commonly presented to end-users from a protein perspective, the choice of using the term “entry” for the name of the main protein class seems justified. The Entry class is the most referenced class within our data model.

Among the several classes directly related to the *Entry* class, some are directly associated with the protein identity. Figure E.V-25 illustrates these classes, along with the central *Entry* class.

Protein identity

A protein has a stable unique identifier: the *AC*. It is commonly referred to as the entry primary accession number. Accession numbers are defined by the user and should be unique within a database. They can be up to 16 characters long. Typically, users would use the UniProtKB identifiers, as well as the less stable NCBI GI identifiers. The tool recognises UniProtKB AC patterns. If a protein with an identical accession number (primary or secondary) exists in UniProtKB, then many of the protein annotations found in UniProtKB will be automatically integrated into the local database[1]. Entries may have several accession numbers. When two entries merge, only one of their primary AC numbers remains the primary AC number of the merged entry. The other AC becomes a secondary accession number. On the contrary, if an entry is split into two or more new entries, each entry acquires a new primary AC. The entry’s original AC becomes a secondary accession number in all the newly created entries. Secondary accession numbers may therefore refer to more than one entry. They are stored in the *SecondaryAC* class where the *secondaryAC* attribute has no uniqueness constraint.

Some other attributes in *Entry* have similar definitions to the ones given in SWISS-2DPAGE and UniProtKB user manuals[2]. *ID* (IDentification) is generally a mnemonic name to identify a protein, although it is not necessarily a stable identifier like AC. If not defined, *ID* will be assigned the current accession number. *entryClass* is a means of

[1] Cross-referencing an entry to UniProtKB gives the same result, independently of the user-defined local accession number.

[2] http://www.expasy.org/ch2d/manch2d.html and http://www.expasy.org/sprot/userman.html

informing about the status of the entry, for example SWISS-2DPAGE uses the status "STANDARD" for data that are complete, and the status "PRELIMINARY" for data that are not. *method* is an optional attribute indicating the primary separation/analysis technique used to identify the protein, and it defaults to "2DG" in 2-DE datasets. The *extractEnzymeCodeFromDE()* method extracts the EC enzyme commission codes from the *description* attribute and stores them in the *EnzymeNomenclature* class (an operation that is performed each time the protein description is modified or automatically updated); the *description* attribute gives general descriptive information about the protein.

The *Entry* class also includes a reference to the organism corresponding to the protein identified against a search database. In our representation, we consider this organism independent from the original studied organism, although by default, and in the majority of cases, they are the same. The Difference may originate from the fact that the protein from the studied organism is not present in the search database, and that the assignment of the identified spots has been made on a close ortholog.

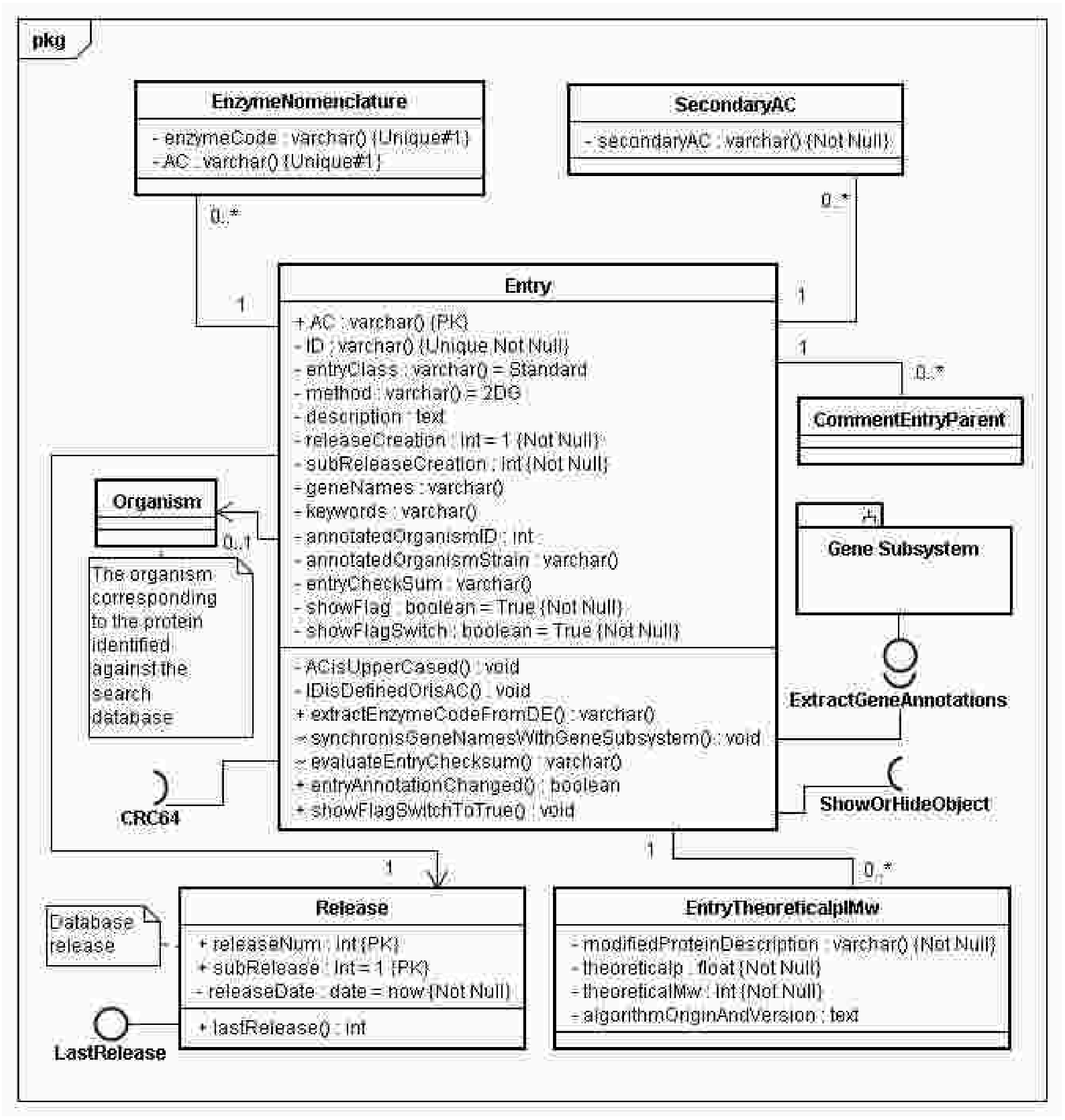

pkg
EnzymeNomenclature
- enzymeCode : varchar() {Unique#1}
- AC : varchar() {Unique#1}
SecondaryAC
- secondaryAC : varchar() {Not Null}
0..*
0..*
1
1
Entry
+ AC : varchar() {PK}
- ID : varchar() {Unique Not Null}
- entryClass : varchar() = Standard
- method : varchar() = 2DG
- description : text
- releaseCreation : int = 1 {Not Null}
- subReleaseCreation : int {Not Null}
- geneNames : varchar()
- keywords : varchar()
- annotatedOrganismID : int
- annotatedOrganismStrain : varchar()
- entryCheckSum : varchar()
- showFlag : boolean = True {Not Null}
- showFlagSwitch : boolean = True {Not Null}
- ACisUpperCased() : void
- IDisDefinedOrIsAC() : void
+ extractEnzymeCodeFromDE() : varchar()
~ synchronisGeneNamesWithGeneSubsystem() : void
~ evaluateEntryChecksum() : varchar()
+ entryAnnotationChanged() : boolean
+ showFlagSwitchToTrue() : void
1
0..*
CommentEntryParent
Organism
0..1
The organism corresponding to the protein identified against the search database
Gene Subsystem
ExtractGeneAnnotations
ShowOrHideObject
CRC64
1
1
0..*
Database release
Release
+ releaseNum : int {PK}
+ subRelease : int = 1 {PK}
- releaseDate : date = now {Not Null}
+ lastRelease() : int
LastRelease
EntryTheoreticalpIMw
- modifiedProteinDescription : varchar() {Not Null}
- theoreticalpI : float {Not Null}
- theoreticalMw : int {Not Null}
- algorithmOriginAndVersion : text

Figure E.V-25: Data Model – The protein Entry class.

A protein entry is conceptually a physical object comparable to gels or spots. It may be visible or hidden from end-users. The *ShowOrHideObject* interface is therefore also applicable to proteins and acts on the central *Entry* class.

Keywords

The *keyword* attribute is a list of functional, structural or location categories separated by semicolons. Users may define these keywords themselves, or let the tool automatically integrate the keywords corresponding to their protein from both manually curated and electronically assigned UniProtKB keywords[1]. The Gene Ontology

[1] http://www.expasy.org/cgi-bin/keywlist.pl

Annotation project GOA[1] aims to map gene ontology (GO) terms with UniProtKB keywords. We will illustrate hereafter the gene ontology classes related to the *Entry* class, and we will see how this mapping could be of interest to extend data integration within the 2-DE databases.

Creation release and entry annotation changes

The *Release* class is an independent class containing the release numbers of the successive publications of the database, along with their corresponding dates. A database release is formally made of a release number (*releaseNum*) and a sub-release (*subRelease*), *e.g.*, Rel. 1.1 or Rel. 15.12. The *releaseCreation* and the *subReleaseCreation* attributes within the *Entry* class refer to the release number when the entry was first added to the database. A newly added entry is assigned to the next database release before being published. SWISS-2DPAGE, as well as UniProtKB, formerly included indications about database releases for annotation changes in entries. We have replaced this representation by entry version numbers similar to the ones currently used in UniProtKB[2]. The *entryAnnotationChanged()* method is meant to verify whether a relevant modification of the protein annotations has occurred, in which case the entry version will be automatically increased by one in the next database release. The entry version mechanism will be detailed further in a following subsection.

Entry checksum

A Make2D-DB II user may need to apply a major update to his/her database installation, either to upgrade to a new version of the tool, or to add a substantial amount of data. In both cases, the tool erases the old database installation to rebuild a new database. Protein entries may be modified by adding some new data, for example some additional gels or spots. Even without adding any new data, the protein entries may still undergo modifications during the update process, since any database update also implies a significant amount of data integration from external resources. Thus, it is important to be able to compare the content of each protein entry before and after the update process, so that the tool can decide whether an entry version increase is necessary. One manner to guarantee a simple comparison is to use cyclic redundancy check (CRC)[3] over the entry content. For optimisation purpose, we chose to include the computed CRC values within the database implementation, so that they are not repeatedly computed each time a comparison is required outside from the RDMBS implementation. The *entryCheckSum* attribute holds the values computed by the *evaluateEntryChecksum()* external method[4].

[1] http://www.ebi.ac.uk/GOA/

[2] http://www.expasy.org/sprot/userman.html#DT_line

[3] http://en.wikipedia.org/wiki/Cyclic_redundancy_check. The algorithm to compute the Cyclic Redundancy Check CRC64 is described in the ISO 3309 standard. The generator polynomial is $x^{64} + x^4 + x^3 + x + 1$.

[4] http://mordor.isb-sib.ch/make2ddb/lib2d/CRC64.pm, courtesy the Swissknife team of the Swiss-Prot group in Geneva.

Gene names

The *geneNames* attribute within an entry is the string of protein-related genes and their details as presented to the end-user. It is mostly based on the *Gene* subsystem that structures independently, and in depth, all the genes contained in the database. In order to evaluate the *geneNames* attribute, the *Entry* class can access the *Gene* subsystem (using the symbolic *synchroniseGeneNamesWithGeneSubSystem()* external method), but is not forced to do so, as *geneNames* can be also defined independently from the *Gene* subsystem. More details will be given hereafter.

Theoretical pI and Mw

We have included a class that stores theoretical pI and Mw values for the proteins. *Entry* is related to *EntryTheoreticalpIMw* with a multiplicity of one-to-zero or many. We may therefore list many isoforms, fragments or modified proteins regarding a specific protein entry. The Mw is directly evaluated according to a protein sequence, while the pI value can be estimated by means of some algorithm that should be described in *algorithmOriginAndVersion*. Using *EntryTheoreticalpIMw* is optional; the tool does not have a procedure to automatically populate the corresponding relation.

Entry comments

We have already shown a relationship between *Entry* and *CommentEntry2D*. In reality, the relationship is a more general relation between *Entry* and *CommentEntryParent*. The latter is a parent class for two children subclasses, *CommentEntryFreeText* and *CommentEntry2D*, like shown in Figure E.V-26. The *CommentTopic* class classifies a list of user-defined protein-related comment topics, like the comment topics that are used in UniProtKB[1].

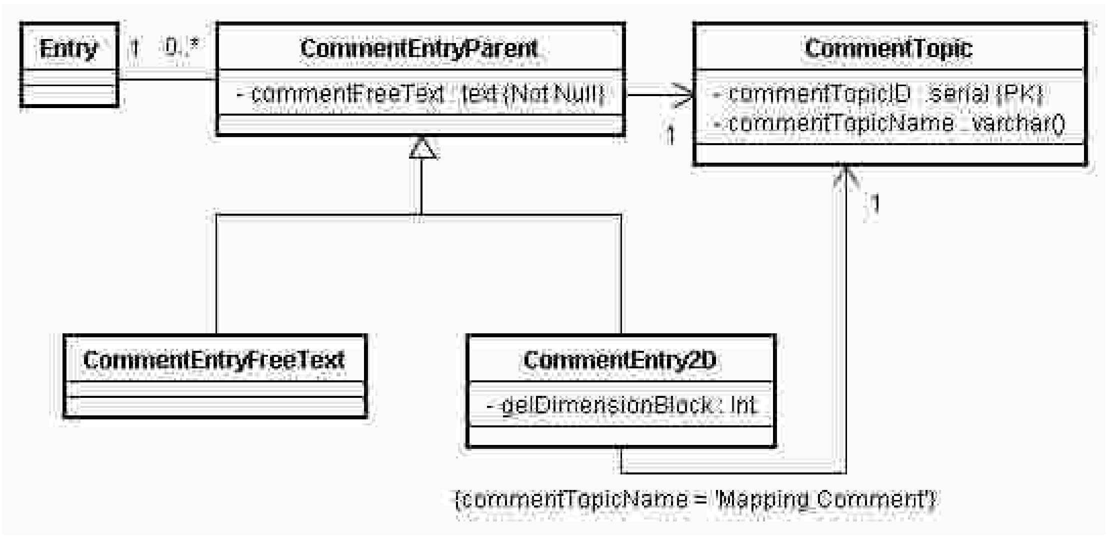

Entry
1
0..*
CommentEntryParent
- commentFreeText : text {Not Null}
1
CommentTopic
- commentTopicID : serial {PK}
- commentTopicName : varchar()
1
CommentEntryFreeText
CommentEntry2D
- gelDimensionBlock : Int
{commentTopicName = 'Mapping Comment'}

Figure E.V-26: Data Model – The entry comments.

[1] http://www.expasy.org/sprot/userman.html#CC_line

The Gene subsystem

The Gene subsystem (Figure E.V-27) encloses all the information related to the genes coding for the proteins of the database. A *Gene* object in the model is composed of one "official" gene name and/or one or several ordered locus names (*i.e.*, OLN or ORF numbers) and/or one or several ORF (open reading frames) names. Ordered locus names are names used to represent ORF in a completely sequenced genome or chromosome, while ORF names are names temporarily attributed by a sequencing project to an ORF[1]. A gene "official" name may have several synonyms. In addition, each Gene object is linked to a specific organism, thus overcoming any homonymy ambiguity with gene names. In most cases, the gene reference to an organism is inherited from the protein that is related to this gene within the database (or in case of splicing, from a set of proteins, assuming that only one organism is concerned).

During the database installation process, genes are extracted from the protein annotations and/or from external data (UniProtKB). The *synchroniseGeneNamesWithGeneSubsystem()* external method symbolises the extraction of the gene data and its reorganisation into the subsystem. As the database content is continuously updated with regard to the external data, the *geneNames* content in *Entry* may change. By default, we do not continuously synchronise the new content with the Gene subsystem, and only the new content is displayed to end-users. Users who wish to personally manage the gene annotations of their databases may be interested in using the Gene subsystem more extensively.

[1] http://www.expasy.org/sprot/userman.html#GN_line

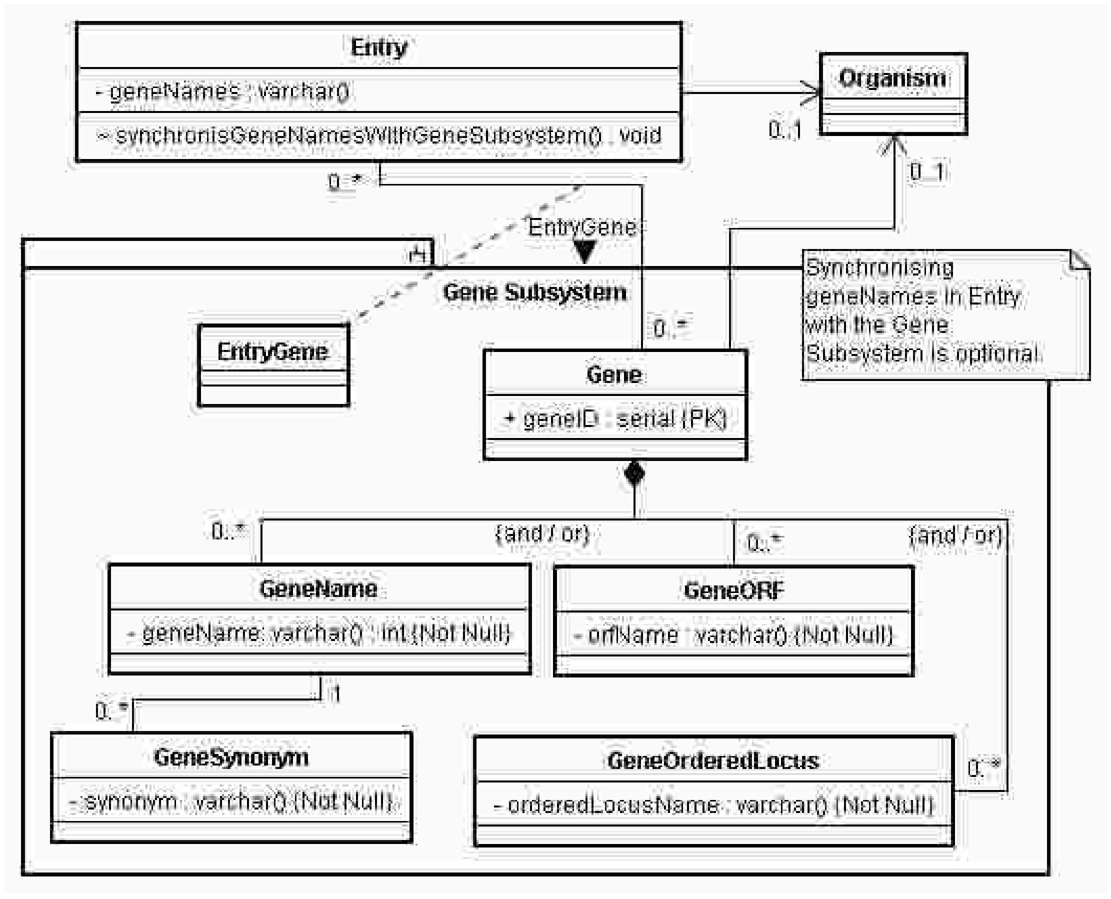

Entry
- geneNames : varchar()
~ synchronisGeneNamesWithGeneSubsystem() : void
Organism
0..1
0..1
0..*
EntryGene
Gene Subsystem
Synchronising geneNames in Entry with the Gene Subsystem is optional.
EntryGene
0..*
Gene
+ geneID : serial {PK}
0..*
{and / or}
0..*
{and / or}
GeneName
- geneName: varchar() : int {Not Null}
GeneORF
- orfName : varchar() {Not Null}
1
0..*
GeneSynonym
- synonym : varchar() {Not Null}
GeneOrderedLocus
- orderedLocusName : varchar() {Not Null}
0..*

Figure E.V-27: Data Model – The Gene Subsystem.[1]

Gene ontology classification

The GOA project[2] (Camon et al. 2004) aims to "*provide high-quality Gene Ontology (GO) annotations to proteins in the UniProt Knowledgebase (UniProtKB) and International Protein Index (IPI) and is a central dataset for other major multi-species databases; such as Ensembl and NCBI*".

The GO classification (C.III) is presented as a graph that is subdivided into three categories: biological processes (*e.g.*, protein folding[3]), molecular functions (*e.g.*, transferase activity[4]) and cellular components (*e.g.*, mitochondrion[5]). The gene ontology classification subsystem in our model is based on the mapping between the GO terms on the one hand, and the UniProtKB entries, keywords annotations[6] and

[1] The Gene subsystem was slightly different until Make2DB-DB II version 2.50.2 included, where the *GeneName* class was at the top of the subsystem.

[2] http://www.ebi.ac.uk/GOA/

[3] http://www.ebi.ac.uk/ego/DisplayGoTerm?id=GO:0006457

[4] http://www.ebi.ac.uk/ego/DisplayGoTerm?id=GO:0016740

[5] http://www.ebi.ac.uk/ego/DisplayGoTerm?id=GO:0005739

[6] http://www.geneontology.org/external2go/spkw2go

enzyme commission codes on the other hand. This mapping is reflected in the 2-DE database through the *EntryGeneOntology* class[1], which links the entries to the Gene ontology subsystem. This can be achieved by extracting the GO terms matching the UniProtKB entries or their keywords, as well as any explicit GO cross-references and enzyme code annotations.

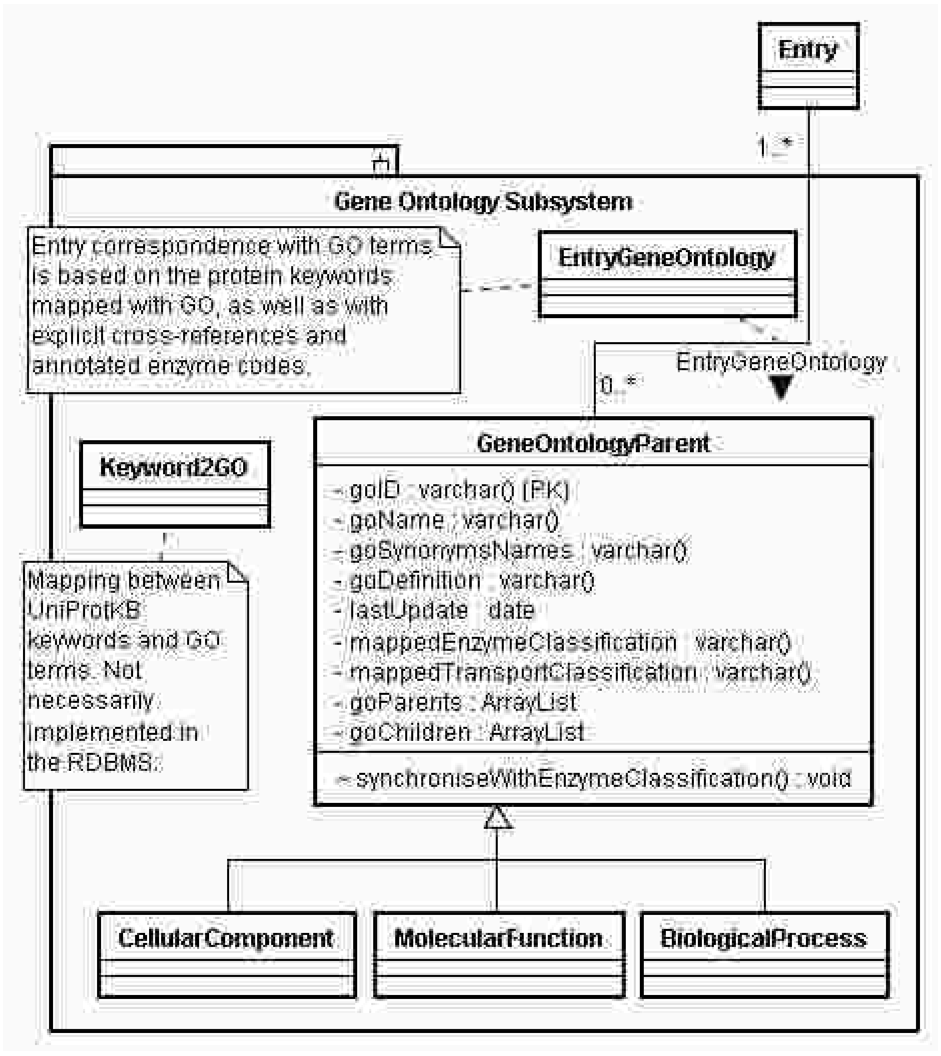

Entry
1..*
Gene Ontology Subsystem
Entry correspondence with GO terms is based on the protein keywords mapped with GO, as well as with explicit cross-references and annotated enzyme codes.
EntryGeneOntology
0..*
EntryGeneOntology
Keyword2GO
Mapping between UniProtKB keywords and GO terms. Not necessarily implemented in the RDBMS.
GeneOntologyParent
- goID : varchar() {PK}
- goName : varchar()
- goSynonymsNames : varchar()
- goDefinition : varchar()
- lastUpdate : date
- mappedEnzymeClassification : varchar()
- mappedTransportClassification : varchar()
- goParents : ArrayList
- goChildren : ArrayList
~ synchroniseWithEnzymeClassification() : void
CellularComponent
MolecularFunction
BiologicalProcess

Figure E.V-28: Data Model – The gene ontology classification.

Details of the system are depicted in Figure E.V-28. The three gene ontology categories are subclasses of the superclass *GeneOntologyParent*. The parent class includes a detailed description of the GO terms[2]: the GO term identifier, its descriptive name, definition and synonyms, as well as the date of the last update of the GO term.

[1] Until Make2D-DB II version 2.50.2, the Gene ontology subsystem had a many-to-one association with *Entry*, which has been replaced by a many-to-many association, via an association class, to eliminate redundancy.

[2] *e.g.*, http://www.ebi.ac.uk/ego/DisplayGoTerm?id=GO:0006525

The hierarchy of the term is limited to its direct parents and direct children identifiers[1] (*goParents* and *goChildren* attributes). Gene ontology data is easily accessed through a variety of available interfaces and web services. The *mappedEnzymeClassification* attribute contains the enzyme commission codes that are mapped to the GO term, while the *mappedTransportClassification* is related to the Transporter Classification (TC) system (Saier, Jr. et al. 2006). TC is a IUBMB approved classification system for membrane transport proteins. It is analogous to the enzyme commission system for classification of enzymes, but incorporates phylogenetic information additionally. TC codes[2] can be extracted from the comment annotations (the CC lines) of a UniProtKB protein.

The mapping between the UniProtKB keywords and the GO terms can be included into the RDBMS implementation, although this is not necessary, as the mapping can be easily managed from outside the relational system. Currently, no such *Keyword2GO* relation is physically implemented in the tool.

During the development phase of the tool, the mapping between GO and UniProtKB entries and keywords was not satisfactory. The use of the Gene ontology subsystem was therefore delayed. We believe that taking advantage of the ontology classification would represent a significant gain in data interpretation and data integration between distributed 2-DE databases. This would expand the prospect of linking related proteins, not only within one 2-DE database, but also across several remote 2-DE databases. One application is to know which related proteins implicated in a pathway have been identified by other users in order to perform quantitative comparison analysis. Currently, the UniProtKB mapping with GO terms is becoming increasingly approved, and we estimate that one of the most important future developments of the tool is to fully exploit this part of our model.

Bibliographic references

The *Entry* class is linked to the bibliographic references' package in a many-to-many association through the *ReferencedEntry* class. Bibliographic references must include all publications that describe the experimental and identification processes that led to protein identification. Whenever several publications are available for the same entry, the 2-DE experimental annotations within the entry must include local references to the appropriate publications. This explains the fact that bibliographic references within an entry have local identifiers (corresponding to the SWISS-2DPAGE '*RN*' line). The *RNdisplayedValue* attribute in Figure E.V-29 represents the local identifier of the bibliographic reference within a specific entry.

[1] For a more extended hierarchy, it would be more reasonable to access the GO database on the fly, rather than saturating the 2-DE database with a huge amount of gene ontology classification.

[2] *e.g.*, http://www.tcdb.org/tcdb/index.php?tc=1.B.3.1.1&Submit=Lookup

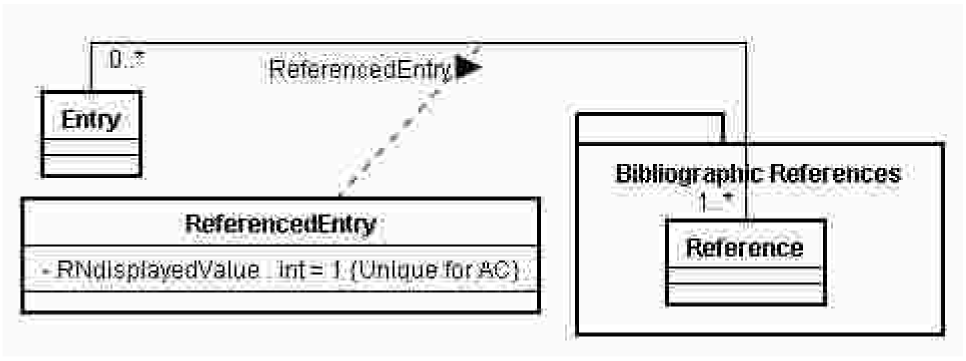


Figure E.V-29: Data Model – Bibliographic references for Entry.

Entry versions

In addition to the creation date, an entry displays information about the last time it has been subject to a relevant modification. This information is a combination of a sequential version number and a date, and it is presented separately for 2-DE related data and for general protein annotations (*e.g.*, http://www.expasy.org/swiss-2dpage/ac=O00299). The *EntryVersionParentClass* in Figure E.V-30 is the superclass of two distinct subclasses, *EntryVersionGeneral* and *EntryVersion2D*, which manage general and 2-DE entry versions independently from each other. The classes include the current version for each entry and the date of the last modification. The date does not reflect the moment at which the modification has been carried out internally, but rather the moment when the database administrator decides to consolidate his/her changes in order to publish a new database release.

Entry versions are automatically managed by the system using a mechanism as follows: a signal tells the entry version classes if a relevant annotation has been modified, added or erased. This automatically triggers the *annotationChanged* boolean attribute to be set to 'true'. An increase of the version number and an update of the version date are performed only when the database administrator decides to make the changes permanently available and to update the materialised views that are presented to end-users. This operation is typically executed before publishing a new database release. *annotationChanged* is then set back to 'false'. Optionally, the materialised views of the protein entries have to be reconstructed at this point and may be limited to the modified entries only.

We have selected a set of attributes that, when changed, cause an entry version to increase. The content of some of these attributes may not be part of the materialised views of the entries as presented to end-users. For example, up to Make2DB-DB II version 2.50.2, the keywords were not displayed within the protein entries, although they were among the set of attributes that could cause an entry version to increase. If the administrator prefers to strictly limit entry version modifications to changes that are visible in the entry materialised view, an alternative procedure may replace the standard one: the use of the *annotationChecksum* attribute. This attribute contains the computed CRC value of the entry text (separated into general annotations and 2-DE annotations) in its last publication. The idea is that when the entries are reconstructed, it is possible to compare their new CRC values with the older ones and to modify the entry version number if they differ. However, this method requires more computer resources and computational time, since before each new database release the CRC values have to be computed over the entire set of entries. A better solution, to strictly limit entry version modifications to visible changes, would be to combine the two mechanisms. This means that only entries with an *annotationChanged* set to 'true' should undergo the CRC verification to decide whether their version number should be increased.

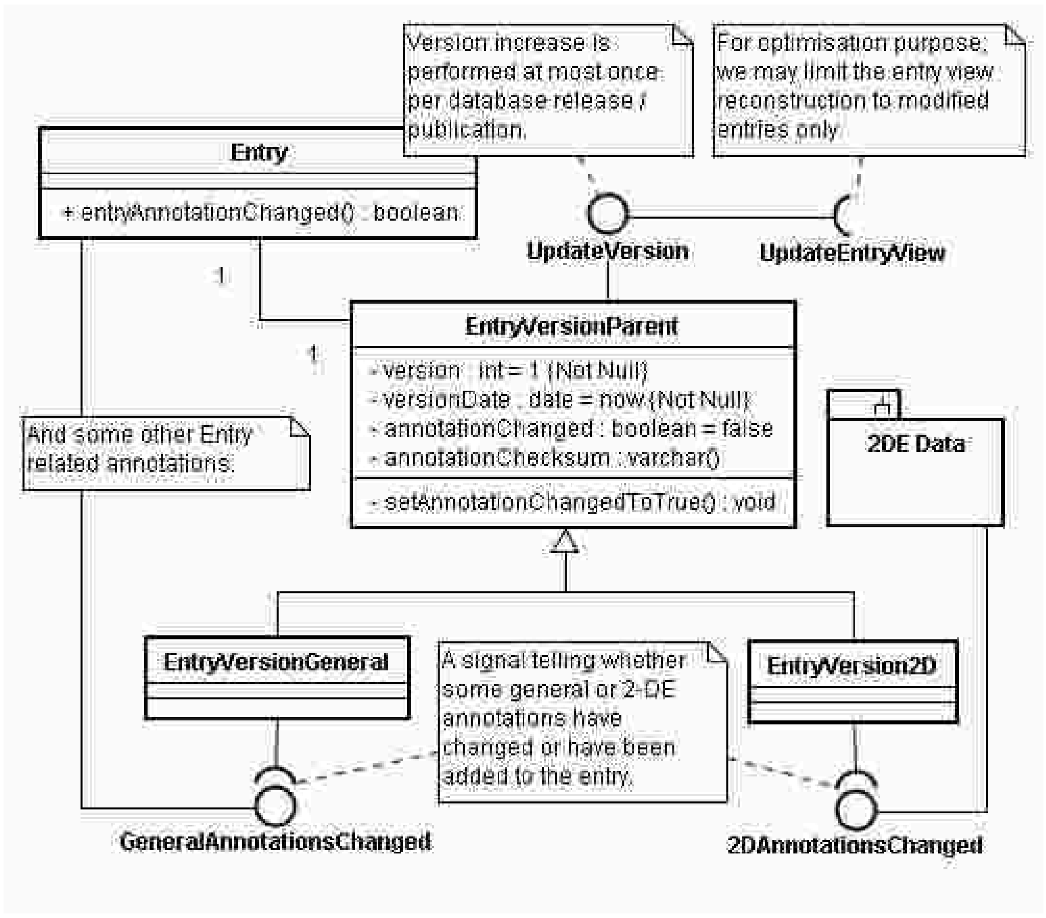

Version increase is performed at most once per database release / publication.
For optimisation purpose; we may limit the entry view reconstruction to modified entries only
Entry
+ entryAnnotationChanged() : boolean
UpdateVersion
UpdateEntryView
1
1
EntryVersionParent
- version : int = 1 {Not Null}
- versionDate : date = now {Not Null}
- annotationChanged : boolean = false
- annotationChecksum : varchar()
- setAnnotationChangedToTrue() : void
And some other Entry related annotations.
2DE Data
EntryVersionGeneral
A signal telling whether some general or 2-DE annotations have changed or have been added to the entry.
EntryVersion2D
GeneralAnnotationsChanged
2DAnnotationsChanged

Figure E.V-30: Data Model – The entry version management.

Initialising entry versions for a new dataset is trivial, as all versions are set to one. However, converting databases that have an earlier publication history in some other format, like SWISS-2DPAGE, into our system is less evident. We resort to generate an initial version number equal to the database release number when an entry was last modified. This guarantees the consecutiveness of versions, but may result in an entry version number higher than the number of times the entry has effectively undergone a modification in the previous releases of the database old format[1].

[1] For SWISS-2DPAGE, it was still possible to track entries' history by analysing all the earlier flat file releases of the database. However, we could not generalise this process to other datasets that are not under our control

Archiving modified entries

In addition to plain database exports in text format as the standard way to archive the database entries on a regular basis, we have also recently opted to slightly expand our implementation to include within the relational database an archive of entries in their older versions. This new feature will be available in the most recent version of our tool that is being finalised at the time of writing[1]. End-users will then be able to easily display an entry in any of its previous versions and to track all modifications it has undergone.

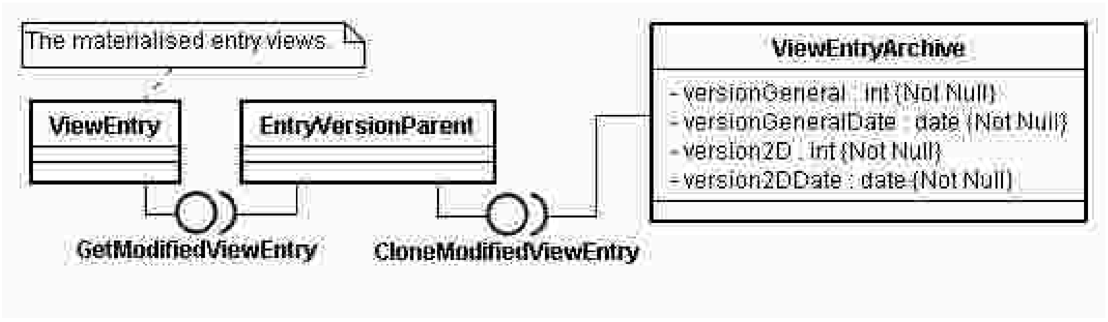

The materialised entry views
ViewEntry
EntryVersionParent
ViewEntryArchive
- versionGeneral : int {Not Null}
- versionGeneralDate : date {Not Null}
- version2D : int {Not Null}
- version2DDate : date {Not Null}
GetModifiedViewEntry
CloneModifiedViewEntry

Figure E.V-31: Data Model – Archiving modified entries.

The process is quite simple. The entries, which are stored as materialised views in a dedicated relation, are cloned into a similar archive relation whenever their annotations change. The cloned relation has two extra fields, general and 2-DE annotation versions, as well as the related dates. Primary accession number modifications (due to a possible merge or split) are accurately managed.

Cross-references of protein entries

Protein entries are cross-linked to external data by means of cross-references. Cross-references are URL pointers that direct the end-user to additional resources relevant to an entry. These resources vary in number and in nature between the different entries. They can be DNA or amino acid sequence databases, domain classification, structure or pathway databases and, evidently, other 2-DE PAGE databases where the same protein has been identified.

We may distinguish between static cross-references that are explicitly given by the user, and dynamic cross-references that the system automatically incorporates, manages and verifies their relevance and current accessibility. Some static cross-references are also controlled by the system to ensure that they are up-to-date.

A cross-reference between a local entry and another resource necessitates, besides a clear definition of the remote database, an unambiguous primary identifier (or pointer) to the information entry on the remote database. Other additional identifiers may be needed to complement the information given by the primary identifier[2] (*e.g.*, specify

[1] Most probably version 2.60.1

[2] http://www.expasy.org/sprot/userman.html#DR_line

the species for a given gene name). The *EntryXrefDbParent* class in Figure E.V-32 lists up to four attributes to cover all potentially needed identifiers for a specific cross-reference. This class is an association class between *Entry* and the *XrefDbParent* class; the latter is the place where metadata about the remote databases is defined (*e.g.*, their location, precise name, etc.). Using a maximum of four atomic attributes for identifiers is sufficient; we do not need to insert an additional intermediate class that may theoretically contain any indeterminate number of identifiers.

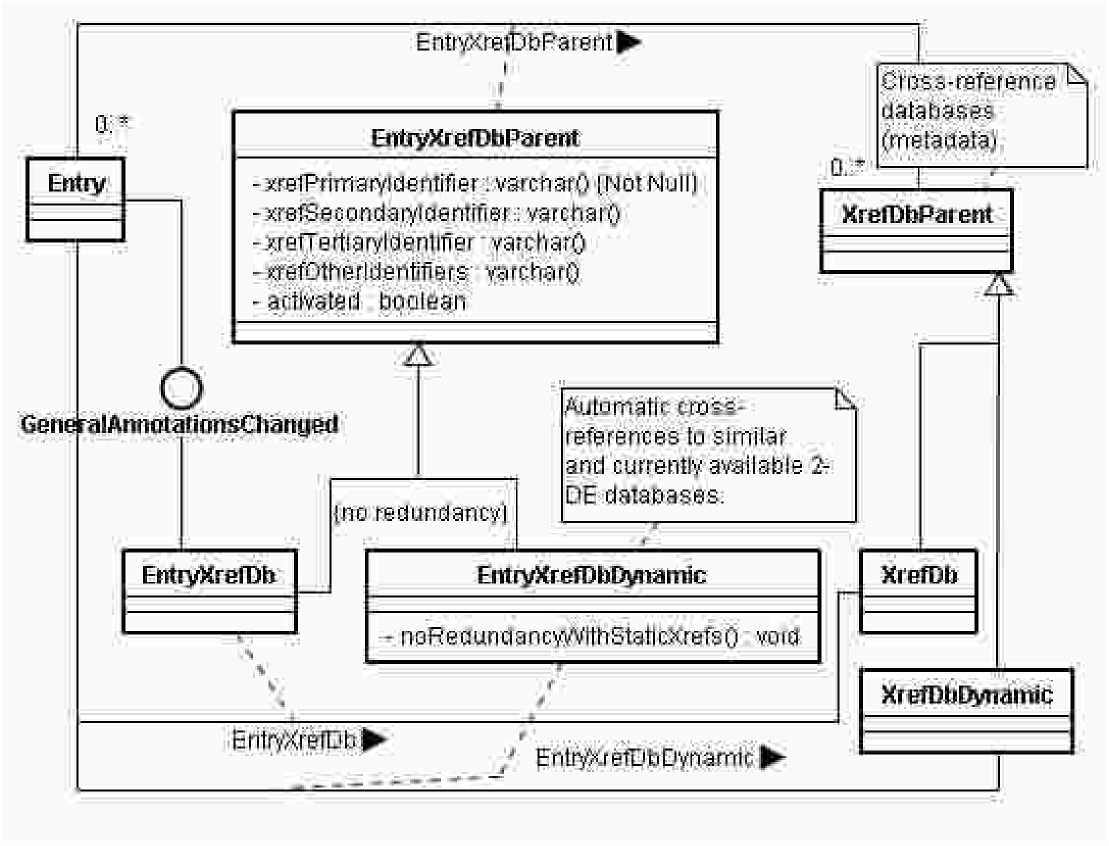

EntryXrefDbParent ▶
0..*
Entry
EntryXrefDbParent
- xrefPrimaryIdentifier : varchar() {Not Null}
- xrefSecondaryIdentifier : varchar()
- xrefTertiaryIdentifier : varchar()
- xrefOtherIdentifiers : varchar()
- activated : boolean
Cross-reference databases (metadata)
0..*
XrefDbParent
GeneralAnnotationsChanged
Automatic cross-references to similar and currently available 2-DE databases.
{no redundancy}
EntryXrefDb
EntryXrefDbDynamic
+ noRedundancyWithStaticXrefs() : void
XrefDb
XrefDbDynamic
EntryXrefDb ▶
EntryXrefDbDynamic ▶

Figure E.V-32: Data Model – Cross-references of protein entries.

XrefDbParent, as its name suggests, is a parent class of two children subclasses: *XrefDb* and *XrefDbDynamic*. *XrefDb* is the subclass where databases that are explicitly given in the cross-references provided by the user are listed. This class also contains a set of additional resources that we will describe in the next section. The other subclass, *XrefDbDynamic*, is exclusively managed by the tool. The user may request to automatically establish cross-references to other remote 2-DE databases (built with the same tool); information about such databases is integrated and kept locally only if these remote 2-DE databases are Web-accessible at the time of the request. Since *XrefDbParent* is a superclass of two subclasses, the association class *EntryXrefDbParent* is, in turn, also a super class of two subclasses relating *Entry* with the cross-reference database subclasses. Similarly, *EntryXrefDb* lists the explicit user-defined cross-references, while *EntryXrefDbDynamic* those that are not considered "permanent". We ensure that non-permanent cross-references are not redundant with the others. If the user defines a cross-reference to a remote 2-DE database, his/her cross-reference will become permanent. The *activated* attribute in the *EntryXrefDb* classes has therefore the role of temporarily deactivating the display of such cross-references to end-users if the remote databases are temporarily inaccessible. Only changes of identifiers, a deletion or an insertion in entry explicit cross-references will imply an entry version modification.

The tool has the ability to switch cross-references from UniprotKB/TrEMBL to UniProtKB/Swiss-Prot when the referenced TrEMBL entry is integrated into Swiss-Prot. If a UniProtKB referenced entry is merged with another entry, the tool updates the cross-reference identifiers to suit the new UniProtKB accession number. In case of a split of a cross-reference entry into several new ones, the tool retains only the entry that belongs to the same species of the gel containing the protein. In case no correspondence between species is detected, the user is invited to choose which one to retain[1].

Cross-reference administration in Make2D-DB II is slightly more complex than described until this point. The next section gives more details on the subject.

E.V.7 Cross-references' management

To be able to uniformly express, homogenise, and verify Web-accessibility of cross-referenced resources, we needed to couple our distributed management process for database metadata - to be used by the tool users - with a centrally harmonised and semi-controlled management process. The first step was to tune this metadata to the different databases available from the ExPASy server, especially those shared between the main index databases, UniProtKB/Swiss-Prot and UniProtKB/TrEMBL, given their relevance to our work, as well as SWISS-2DPAGE and any other non-ExPASy 2-DE PAGE database built with our tool. We have therefore proposed to centralise this information in a local text document to be shared by all the ExPASy databases (DbCrossRefs.txt). For high consistency of data, an autonomous module has been conceived to manage the document content: the DbCrossRefs Perl module, which is also distributed with the Make2D-DB II package:

- ❖ http://world-2dpage.expasy.org/make2ddb/DbCrossRefs.html

The module provides methods to:

- Create, manage and read files listing cross-reference database links in both text and DBM format.

- Check the availability of the links.

- Export and import files.

Databases are exhaustively listed by their common name, the URL to display a cross-reference, and an optional comment /description text. The URL contains placeholders (sequential numbers between brackets) where the given identifiers of the

[1] Choosing one entry among several split ones is currently only implemented for the database installation process, but not for the administration Web interface.

cross-reference will be inserted on the fly[1]. This document is continuously maintained by the Swiss Institute of Bioinformatics in Geneva, and is consequently an appropriate master document for the Make2D-DB II users. The document is distributed with our tool, and is transparently updated on the remote 2-DE PAGE database installations.

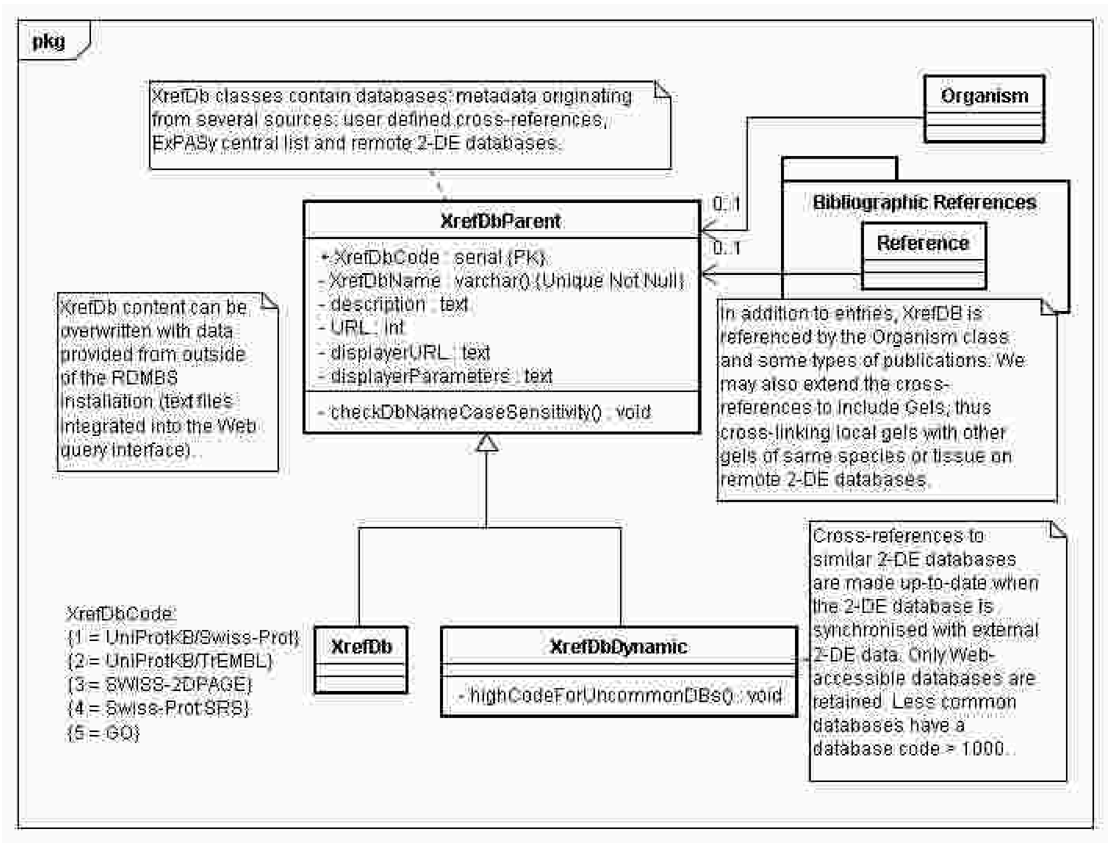

pkg
XrefDb classes contain databases' metadata originating from several sources: user defined cross-references, ExPASy central list and remote 2-DE databases.
Organism
Bibliographic References
Reference
0..1
0..1
XrefDbParent
+ XrefDbCode : serial {PK}
- XrefDbName : varchar() {Unique Not Null}
- description : text
- URL : int
- displayerURL : text
- displayerParameters : text
- checkDbNameCaseSensitivity() : void
XrefDb content can be overwritten with data provided from outside of the RDMBS installation (text files integrated into the Web query interface).
In addition to entries, XrefDB is referenced by the Organism class and some types of publications. We may also extend the cross-references to include Gels, thus cross-linking local gels with other gels of same species or tissue on remote 2-DE databases.
XrefDbCode:
{1 = UniProtKB/Swiss-Prot}
{2 = UniProtKB/TrEMBL}
{3 = SWISS-2DPAGE}
{4 = Swiss-Prot SRS}
{5 = GO}
XrefDb
XrefDbDynamic
- highCodeForUncommonDBs() : void
Cross-references to similar 2-DE databases are made up-to-date when the 2-DE database is synchronised with external 2-DE data. Only Web-accessible databases are retained. Less common databases have a database code > 1000.

Figure E.V-33: Data Model – The Cross-reference database classes.

XrefDb class is populated with databases that are extracted from the protein annotations and the central text document available from the ExPASy server. Additional registered Make2D-DB II databases are also included, but only if they are accessible when the system is updated (*XrefDbDynamic*). *XrefDB* contains a set of permanent databases that are predefined by the tool. This set includes the protein main index databases (UniProtKB) and some other major resources that are essential to the database implementation. Taxonomy and bibliographic databases are also included, as organisms and bibliographic references have as well cross-references to external resources. In practice, it is also possible to automatically establish cross-references between local gels and gels that are published in other remote 2-DE databases built or that are compatible with our tool. In order to achieve meaningful gel cross-referencing, we will first need a significant number of 2-DE databases to become public. Gel cross-references will be based on common species and/or tissues.

[1] http://mordor.isb-sib.ch/make2ddb/text/DbCrossRefs.txt

Attributes of the *XrefDb* classes are given in Figure E.V-33. Beside the database identifier and common name, a short description of the database can be given (or read from the DbCrossRefs central list). *URL* is the main server address where the database is located, while the *displayerURL* is the URL address of the viewer that will display the cross-referenced data. *displayerURL* and *displayerParameters* are given in the same syntax as the one used in the DbCrossRefs central list.

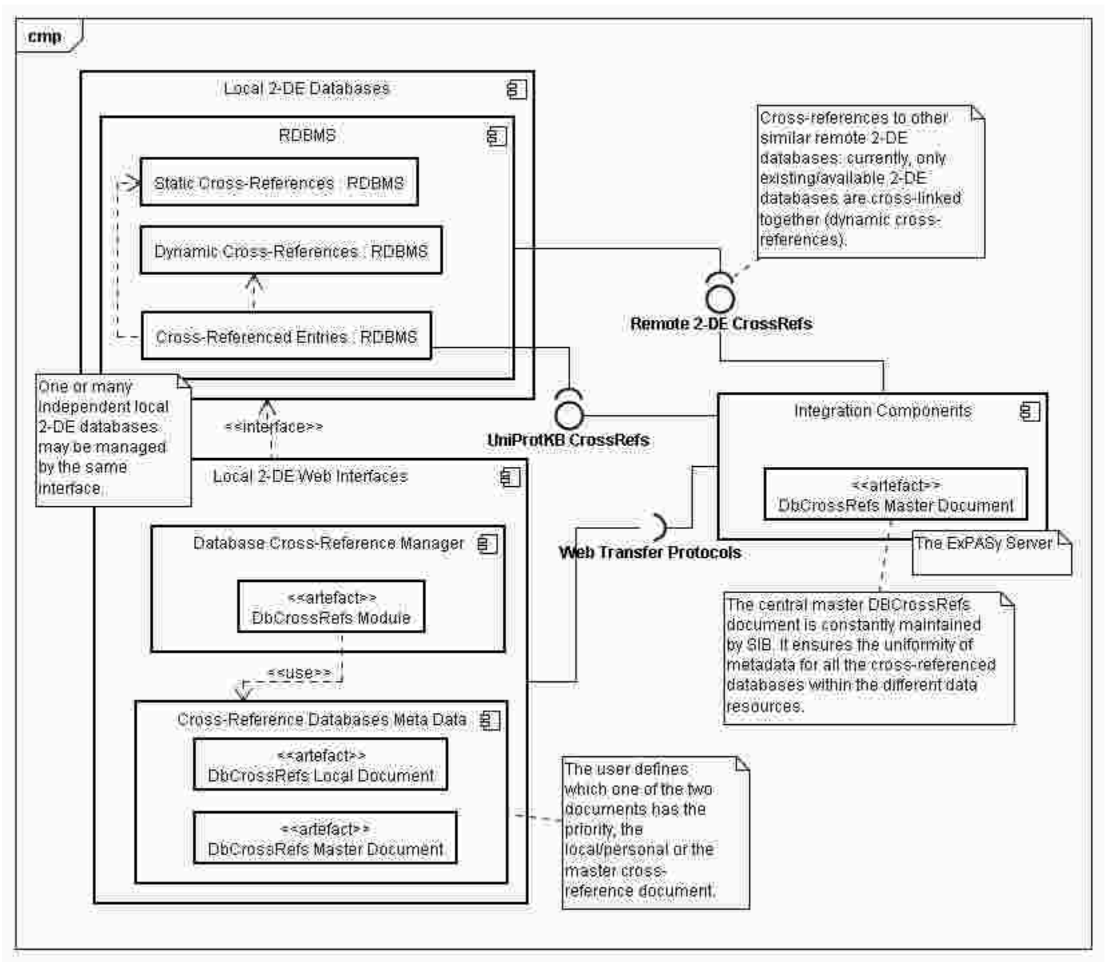

cmp
Local 2-DE Databases
RDBMS
Static Cross-References : RDBMS
Dynamic Cross-References : RDBMS
Cross-Referenced Entries : RDBMS
Cross-references to other similar remote 2-DE databases: currently, only existing/available 2-DE databases are cross-linked together (dynamic cross-references).
Remote 2-DE CrossRefs
One or many independent local 2-DE databases may be managed by the same interface
<<interface>>
UniProtKB CrossRefs
Integration Components
<<artefact>> DbCrossRefs Master Document
Local 2-DE Web Interfaces
Web Transfer Protocols
The ExPASy Server
Database Cross-Reference Manager
<<artefact>> DbCrossRefs Module
The central master DBCrossRefs document is constantly maintained by SIB. It ensures the uniformity of metadata for all the cross-referenced databases within the different data resources.
<<use>>
Cross-Reference Databases Meta Data
<<artefact>> DbCrossRefs Local Document
<<artefact>> DbCrossRefs Master Document
The user defines which one of the two documents has the priority, the local/personal or the master cross-reference document.

Figure E.V-34: The mechanism of integrating and managing cross-reference metadata.

The tool, which integrates locally data from UniProtKB annotations, adds to the local entries the list of cross-references given to the corresponding UniProtKB entries (if a mapping between local entries and UniProtKB entries is possible). UniProtKB cross-references displayed from within the local 2-DE databases will only rely on the central DbCrossRefs document. A simple description of the cross-reference management, intended to the tool users, is given at:

- http://world-2dpage.expasy.org/make2ddb/1.Readme_main.html#cross-references

An overview of the mechanism of integrating and managing cross-references after the local 2-DE database has been installed is given in Figure E.V-34.

E.V.8 External general and 2-DE data

Data integration in Make2D-DB II is performed during the 2-DE database initial installation, and it continues to be performed throughout the database lifetime by means of the administration interface. The integration includes data imported from the ExPASy and the EBI servers, from UniProtKB, from NCBI / NEWT taxonomy and from an unlimited number of remote 2-DE databases built with our tool. The number of resources can be extended to include a variety of other relevant data, like, as already mentioned, the GO classification. Data integration is controlled by users through configurable parameters that define the level of integration.

We have already come through several classes that are dedicated to external data. The *XrefDbDynamic* class, responsible for storing metadata of remote 2-DE databases, and the *EntryXrefDbDynamic* class that establishes dynamic cross-references between remote 2-DE entries and local entries, are both part of the data integration system. *TissueSP* and *TissueSPAliase* classify tissue names as defined and imported from Swiss-Prot. The Gene system, the Gene ontology classification system and the organism classification get essentially their content from external resources. Other classes that include or rely on external data are listed in this section.

The main index: UniProtKB protein annotations

Besides being the main index[1] of protein entries in our tool, UniProtKB is a resource for protein annotations. A dedicated class, *ExternalMainXrefData*, contains UniProtKB annotations for the proteins of the local 2-DE database if these proteins are mapped to UniProtKB. The class (Figure E.V-35) lists a number of selected annotations: the protein identity (accession numbers, ID, description, enzyme codes, the general and sequence versions and their corresponding dates), but also the list of keywords, gene names and the organelle / plasmid annotations (the '*OG*' line[2]). Most importantly, the complete list of explicit cross-references to relevant resources is also incorporated. The UniProtKB release version and the date of the extraction are included within the class. An additional attribute, *spOrTrembl*, tells if the UniProtKB entry is a Swiss-Prot entry or is still a TrEMBL entry.

[1] The index that allows unambiguous designation of individual proteins.

[2] http://www.expasy.org/sprot/userman.html#OG_line

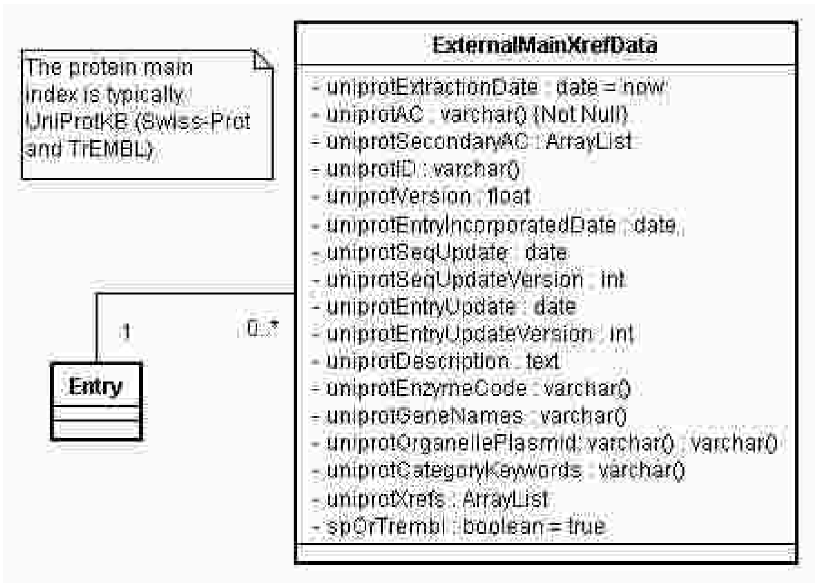

The protein main index is typically UniProtKB (Swiss-Prot and TrEMBL)
ExternalMainXrefData
- uniprotExtractionDate : date = now
- uniprotAC : varchar() {Not Null}
- uniprotSecondaryAC : ArrayList
- uniprotID : varchar()
- uniprotVersion : float
- uniprotEntryIncorporatedDate : date
- uniprotSeqUpdate : date
- uniprotSeqUpdateVersion : int
- uniprotEntryUpdate : date
- uniprotEntryUpdateVersion : int
- uniprotDescription : text
- uniprotEnzymeCode : varchar()
- uniprotGeneNames : varchar()
- uniprotOrganellePlasmid: varchar() : varchar()
- uniprotCategoryKeywords : varchar()
- uniprotXrefs : ArrayList
- spOrTrembl : boolean = true
1
0..*
Entry

Figure E.V-35: Data Model – The main index / UniProtKB protein annotations.

UniProtKB annotations can substitute local entry annotations upon user's request. The user chooses a level of substitution, which can range from partial to full replacement. Only a change of UniProtKB accession numbers and a switch from TrEMBL to Swiss-Prot will unconditionally imply an update of the local entries cross-references to UniProtKB. The major part of the internal procedures controlling the mechanism of local data substitution with external data is listed at[1]:

- http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_update_internal.pgsql
- http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html#core.function.make2db-update-internal-de-integer

Connecting to remote 2-DE databases

Any 2-DE database built with the Make2D-DB II tool can register on the ExPASy server, and therefore becomes part of a 2-DE net where nodes (remote databases) can interconnect with each other. In order to contact another 2-DE resource, a 2-DE database must catch the required information about the other node locations, information that is centralised by the Make2D-DB II integration components on the ExPASy server. This process is currently in use for the interconnection of 2-DE databases. To give nodes more independence in the future, we have included a relation within the database implementation that can be used to store the required metadata needed to contact other remote 2-DE databases without systematically going through the ExPASy server. The *dynamicRemoteMake2DDBInterface* class (Figure E.V-36) includes information about the Web addresses to access other nodes, as well as specific identifiers given for each node. The identifiers (*interfaceID*) should be defined once by a central authority (for example, by our group), and they should then serve to clearly

[1] A set of procedures labelled "core.make2db_update_internal*()".

identify a specific database, even if its location, its name or the local *dbNumber* change[1]. The information contained in *dynamicRemoteMake2DDBInterface* is not supposed to be edited by users, but it must be sent by the ExPASy server to update the relation content when an update is requested. By convention, the local database will have an *interfaceID* equal to zero, so that the tool knows which address corresponds to the local interface, thus avoiding contacting it.

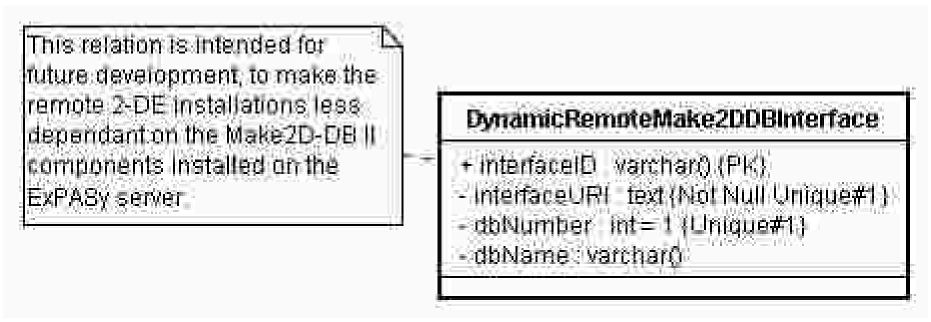

This relation is intended for future development, to make the remote 2-DE installations less dependant on the Make2D-DB II components installed on the ExPASy server.
DynamicRemoteMake2DDBInterface
+ interfaceID : varchar() {PK}
- interfaceURI : text {Not Null Unique#1}
- dbNumber : int = 1 {Unique#1}
- dbName : varchar()

Figure E.V-36: Information about the remote Make2D-DB II interfaces / databases.

A Make2D-DB II query interface is able to manage simultaneously several local databases at the same time. Hence, each local database is given by the interface a distinct sequential number starting from one. The *dbNumber* attribute serves to define this local database number when several databases are managed by the same interface.

Remote gels

In addition to information about remote 2-DE databases, the system optionally stores data on the gels found in these remote databases. Actually, this task is partially managed by the Web interface, independently of the RDBMS implementation. When the interface acts as a 2-DE portal, thus accessing itself many other remote interfaces - a feature that will be discussed hereafter – information about the gels managed by the remote interfaces are collected and stored locally in the portal directories. For completeness, and for more autonomy of the RDBMS implementations, with regard to the Web interfaces, a dedicated class is included in the data model to store the same kind of data. *GelDynamic* contains therefore detailed information about the gels that belong to other remote 2-DE databases. Figure E.V-37 lists the elements used to define and access these gels. Besides storing the gel related data (names, organism and tissue), the relation also stores the name of the remote database and the URL to access the gel. The association between *GelDynamic* and *DynamicRemoteMake2DDBInterface* is logical but is not physically implemented. Since storing information in *DynamicRemoteMake2DDBInterface* is optional, we needed to make sure that *GelDynamic* would still be operational even in the absence of related data in *DynamicRemoteMake2DDBInterface*.

[1] When a Web interface is managing several local databases, each of the local databases gets a unique local number.

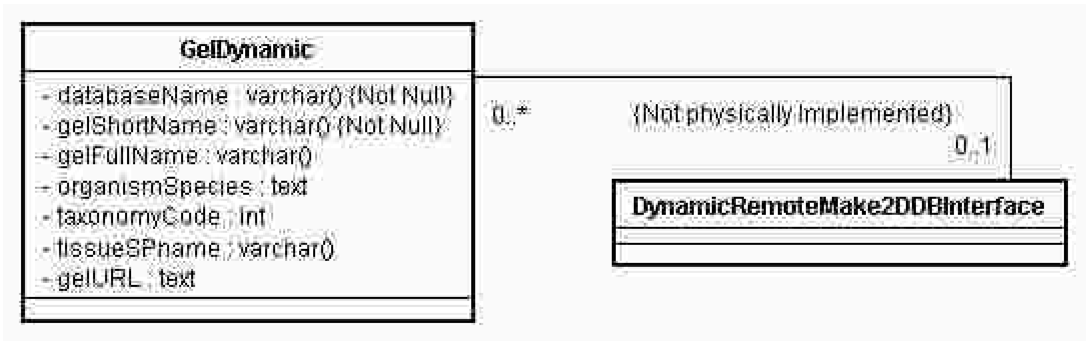

GelDynamic
- databaseName : varchar() {Not Null}
- gelShortName : varchar() {Not Null}
- gelFullName : varchar()
- organismSpecies : text
- taxonomyCode : int
- tissueSPname : varchar()
- gelURL : text
0..*
{Not physically implemented}
0..1
DynamicRemoteMake2DDBInterface

Figure E.V-37: Information about gels on remote databases.

Estimated location of a protein on remote gels

SWISS-2DPAGE has an add-on feature that evaluates the theoretical location of any UniProtKB protein over its various maps, provided the whole amino acid sequence is available in UniProtKB. The estimation is obtained according to the computed pI and Mw of the protein sequence and is graphically displayed to end-users as a region where the protein is expected to be[1], *e.g.*, http://www.expasy.org/cgi-bin/ch2d-compute-map?ECOLI,P02760.

Make2D-DB II users can access this feature thanks to the *GelComputableDynamic* class. This class includes information about available remote gels and the add-on URL that computes the protein location over these remote gels. In theory, this class can include any available tool performing the computation of the location of a protein with a known sequence (*e.g.*, VIRTUAL2D, C.IV.6), as well as the list of gels that the tool can display. For the moment, only the SWISS-2DPAGE add-on and maps are incorporated. The *gelComputableURL* attribute contains, in addition to the address of the tool that performs the computation, a set of placeholders to be substituted on the fly with the appropriate parameters needed by the computational tool (a substitution operation that is performed by the Make2D-DB II query interface).

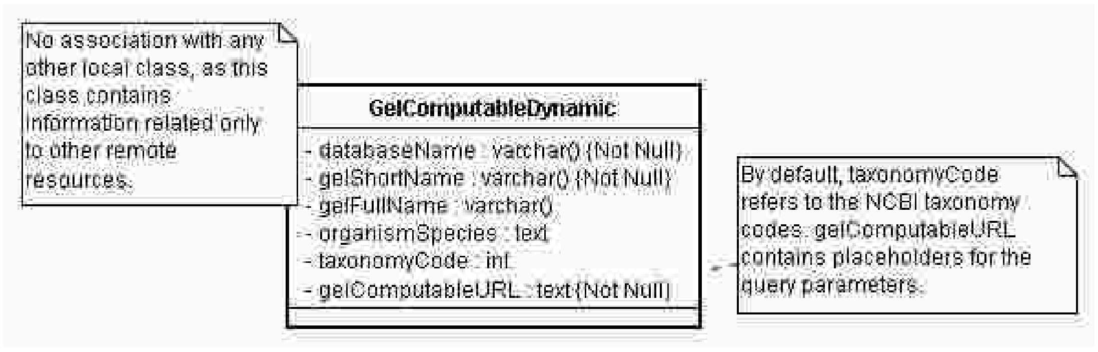

No association with any other local class, as this class contains information related only to other remote resources.
GelComputableDynamic
- databaseName : varchar() {Not Null}
- gelShortName : varchar() {Not Null}
- gelFullName : varchar()
- organismSpecies : text
- taxonomyCode : int
- gelComputableURL : text {Not Null}
By default, taxonomyCode refers to the NCBI taxonomy codes. gelComputableURL contains placeholders for the query parameters.

Figure E.V-38: Data Model – Computed location of a protein on remote maps.

[1] http://www.expasy.org/ch2d/expl-gifs.html

E.V.9 Bibliographic references

Bibliographic references are the literature citations, the sources from which the data has been abstracted. The Make2D-DB II "Bibliographic references" package is an autonomous component of the model that manages exhaustively different types of literature publications. The model is close to the UniProtKB and the SWISS-2DPAGE representation for references (the '*R**' lines)[1].

The *Reference* class is at the top of the package. It defines the identity of the bibliographic reference and gives its title and any related comment. The checksum attribute is intended for comparison of consolidated references[2]. The class combines other classes that are aggregated to it (Figure E.V-39).

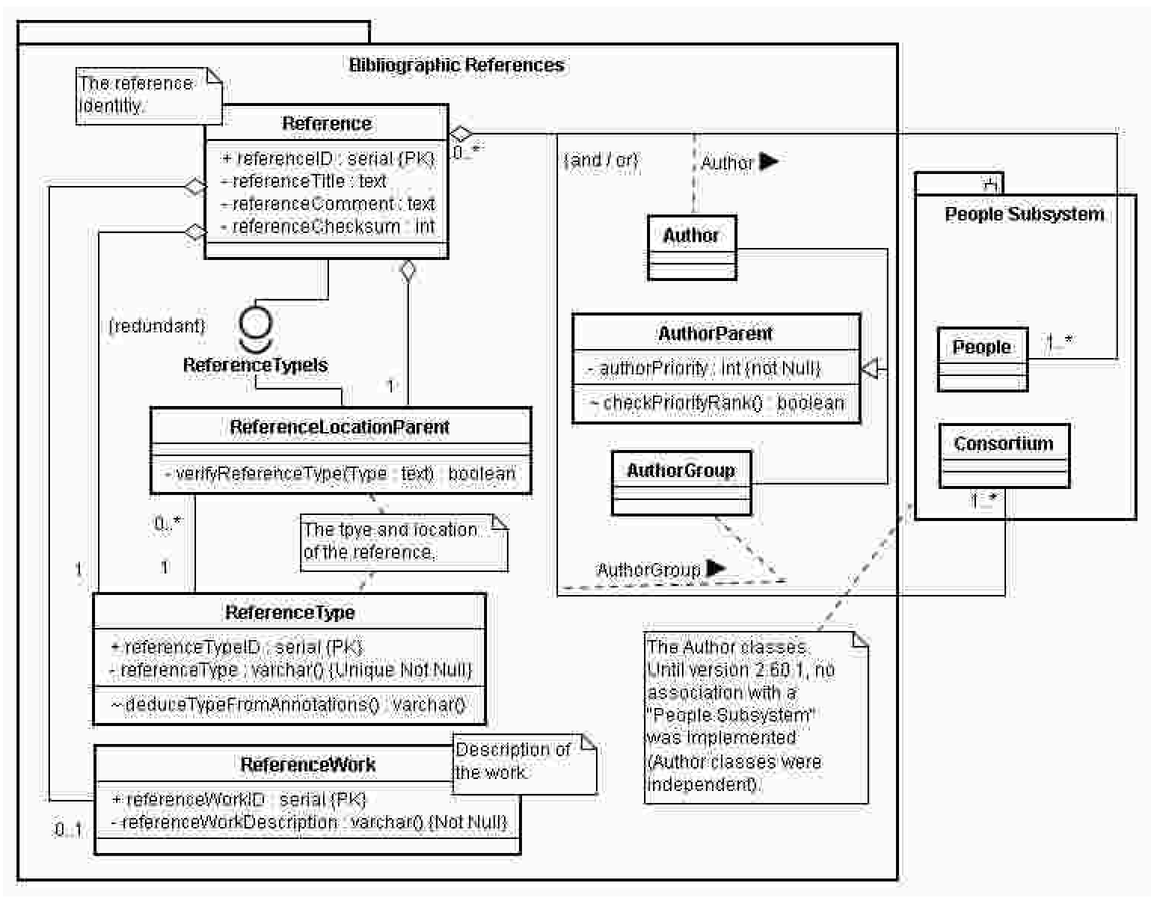

Bibliographic References
The reference identity.
Reference
+ referenceID : serial {PK}
- referenceTitle : text
- referenceComment : text
- referenceChecksum : int
0..*
{and / or}
Author ▶
Author
People Subsystem
{redundant}
ReferenceTypeIs
1
AuthorParent
- authorPriority : int {not Null}
~ checkPriorityRank() : boolean
People
1..*
ReferenceLocationParent
- verifyReferenceType(Type : text) : boolean
AuthorGroup
Consortium
1..*
0..*
The tpye and location of the reference.
1
1
AuthorGroup ▶
ReferenceType
+ referenceTypeID : serial {PK}
- referenceType : varchar() {Unique Not Null}
~deduceTypeFromAnnotations() : varchar()
The Author classes. Until version 2.60.1, no association with a "People Subsystem" was implemented (Author classes were independent).
Description of the work.
ReferenceWork
+ referenceWorkID : serial {PK}
- referenceWorkDescription : varchar() {Not Null}
0..1

Figure E.V-39: Data Model – The Bibliographic References package.

We explain the extent of the bibliographic work carried out by the authors of the reference: the *ReferenceWork* class describes the information that has been propagated into the database entry (*e.g.*, protein sequence, review, mapping on gel, etc.). A

[1] http://www.expasy.org/sprot/userman.html#Ref_line and http://www.expasy.org/ch2d/manch2d.html#Heading14

[2] The consolidation of a reference is part of the materialised views procedures.

reference type is assigned to the reference by the *ReferenceType* class. The type of a reference is controlled, and is one of the following pre-selected types:

- Journal	- Book	- Submitted
- Unpublished results	- Unpublished observations	- Personal communication
- Thesis	- Patent Number	

An external method, *deduceTypeFromAnnotations()*, is associated to the class. Its role is to parse the reference annotations given by the user and to deduce which type of reference they match (the user is free to define the keywords he/she uses to indicate a specific type of work). Any unrecognised or non-listed annotation is categorised as being 'Other' (currently, the new UniProtKB type 'electronic publications' fall under this category). The detailed information necessary for the citation and the localisation of the reference is stored in the Reference locations subsystem, presented in Figure E.V-39 by the *ReferenceLocationParent* superclass (more details are given afterwards).

In Make2D-DB II version 2.50.2 and earlier, we had a direct association between *ReferenceType* and the *Reference* class, which conceptually represents a redundancy at the relational level. We believed that such a direct association would be useful is some particular queries. Therefore, we have implemented a mechanism that guarantees that a reference type associated with a *Reference* instance is matching the type that corresponds to the reference location (the *verifyReferenceType()* method and the *ReferenceTypeIs* interface). The redundant association had no negative effect on data consistency. However, we now believe that this process is not necessary, and we will be able to remove it from the database implementation in version 2.60.1.

Any reference should have a list of authors. There should be at least one author or a group of authors (*e.g.*, a consortium) linked to the reference. The *Author* and the *AuthorGroup* classes are subclasses of the more general *AuthorParent* superclass. Both of them define a rank position for the author (or the author group) in the authors' list, using the *authorPriority* attribute. For each reference, the uniqueness of a position over the whole list of authors is controlled by the *checkPriorityRank()* external method. We have recently decided to include a 'People' subsystem, and to relate the previously independent authors' classes to it. This subsystem is presented in Figure E.V-40 where a superclass is parent of three distinct subclasses: a *Contact* subclass that we have already encountered in previous sections of the model, as well as a *People* and a *Consortium* subclass. The *Author* subclass becomes thus an association class between *Reference* and *People*, while the *AuthorGroup* subclass becomes an association class between *Reference* and *Consortium*. The *Contact* subclass is distinct from the *People* subclass in the sense that it may also point to non-physical persons, or design people by their function instead of their name.

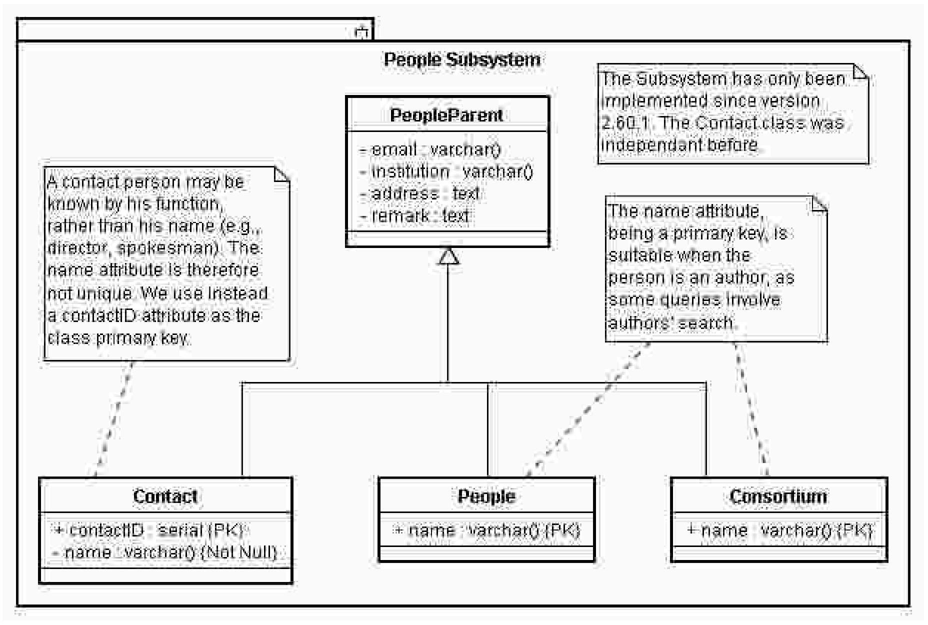

People Subsystem
PeopleParent
- email : varchar()
- institution : varchar()
- address : text
- remark : text
The Subsystem has only been implemented since version 2.60.1. The Contact class was independant before
A contact person may be known by his function, rather than his name (e.g., director, spokesman). The name attribute is therefore not unique. We use instead a contactID attribute as the class primary key.
The name attribute, being a primary key, is suitable when the person is an author, as some queries involve authors' search.
Contact
+ contactID : serial (PK)
- name : varchar() (Not Null)
People
+ name : varchar() (PK)
Consortium
+ name : varchar() (PK)

Figure E.V-40: Data Model – The People subsystem.

Full details covering the localisation of the references are included within the Reference locations subsystem (Figure E.V-41). The subclasses express the categories of the different reference types and their relationship. We have omitted here to represent a class called *Citer* and its association with *ReferenceLocationJournal*. Until recently, UniProtKB included in bibliographic references "citers", who are authors citing unpublished works carried out by others. This subcategory has been abandoned by UniProtKB and consequently we prefer not to promote it any longer[1].

[1] However, the mechanism for recognising this subcategory is still active within the tool.

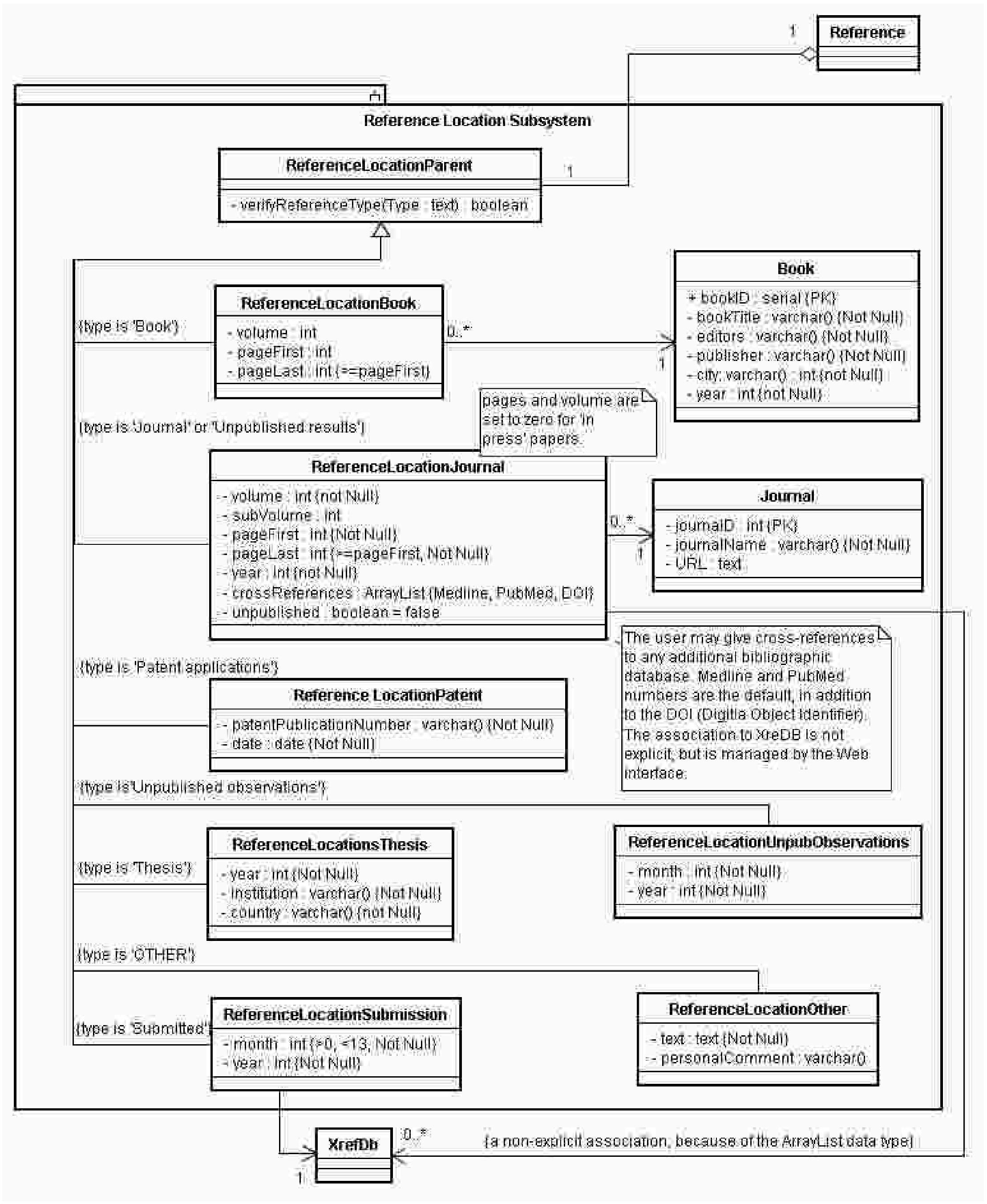

1
Reference
Reference Location Subsystem
ReferenceLocationParent
1
- verifyReferenceType(Type : text) : boolean
Book
+ bookID : serial {PK}
- bookTitle : varchar() {Not Null}
- editors : varchar() {Not Null}
- publisher : varchar() {Not Null}
- city: varchar() : int {not Null}
- year : int {not Null}
ReferenceLocationBook
{type is 'Book'}
- volume : int
- pageFirst : int
- pageLast : int {>=pageFirst}
0..*
1
pages and volume are set to zero for 'in press' papers.
{type is 'Journal' or 'Unpublished results'}
ReferenceLocationJournal
- volume : int {not Null}
- subVolume : int
- pageFirst : int {Not Null}
- pageLast : int {>=pageFirst, Not Null}
- year : int {not Null}
- crossReferences : ArrayList {Medline, PubMed, DOI}
- unpublished : boolean = false
0..*
1
Journal
- journalID : int {PK}
- journalName : varchar() {Not Null}
- URL : text
The user may give cross-references to any additional bibliographic database. Medline and PubMed numbers are the default, in addition to the DOI (Digitla Object Identifier). The association to XrefDB is not explicit, but is managed by the Web interface.
{type is 'Patent applications'}
Reference LocationPatent
- patentPublicationNumber : varchar() {Not Null}
- date : date {Not Null}
{type is'Unpublished observations'}
ReferenceLocationsThesis
{type is 'Thesis'}
- year : int {Not Null}
- institution : varchar() {Not Null}
- country : varchar() {not Null}
ReferenceLocationUnpubObservations
- month : int {Not Null}
- year : int {Not Null}
{type is 'OTHER'}
ReferenceLocationSubmission
{type is 'Submitted'}
- month : int {>0, <13, Not Null}
- year : int {Not Null}
ReferenceLocationOther
- text : text {Not Null}
- personalComment : varchar()
XrefDb
0..*
1
{a non-explicit association, because of the ArrayList data type}

Figure E.V-41: The Data Model – The Reference location subsystem.

We have already mentioned association classes linking specific classes to the bibliographic reference system. This situation is generalised with the *ReferencedObjectParent* association superclass linking the Reference class to the symbolic *Object* class in a many-to-many association (Figure E.V-42).

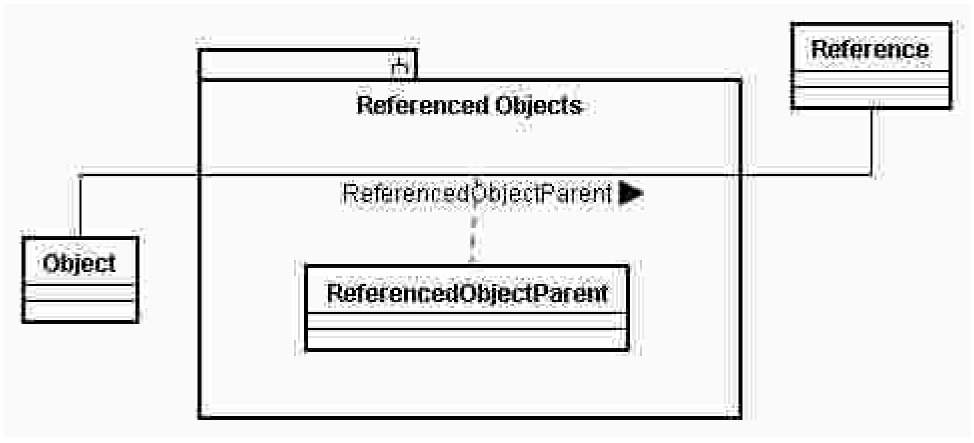


Figure E.V-42: The referenced objects subsystem.

The *ReferencedObjectParent* association superclass is parent of the *ReferencedProject*, *ReferencedSample*, *ReferencedGel* and *ReferencedEntry* subclasses. Any object that will be added to the model in future developments will have to use a similar association subclass in order to be linked to the Bibliographic subsystem.

E.V.10 Materialised views and related operations

Many materialised views are present in our data model. They offer an efficient access to pre-formatted data views. Some of them are final perspectives that are presented (after some additional Web reformatting) to end–users, while others serve mainly to construct other views or to optimise common queries. A full collection of management operations and interfaces help to assemble these views, as well as to keep them the most up-to-date. These procedures are server-side functions, which means they are part of the RDBMS implementation. They are nested at several levels, with the uppermost level offering a high degree of abstraction[1].

Figure E.V-43 shows how the different components interact in order to materialise different views that, in turn, contribute in materialising other views. For example, bibliographic references are assembled in the *ViewRef* tables (*ViewRef* and *ViewRefLast*)[2] in order to make text parsing easier An interface accesses *ViewRef* to provide other procedures with the materialised bibliographic references specific to each protein entry (the *Entry Reference Blocks* interface). Similarly, the spot identifications of each entry are gathered and stored in *ViewSpotEntry* (or *ViewSpotEntryLast*) from where they become available to other procedures.

[1] http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_final_views.pgsql

[2] For technical reasons, some materialised views have a similar cloned view (*ViewObjectLast*) containing a single instance. These cloned views are used when only one single specific element to assemble is needed on the fly.

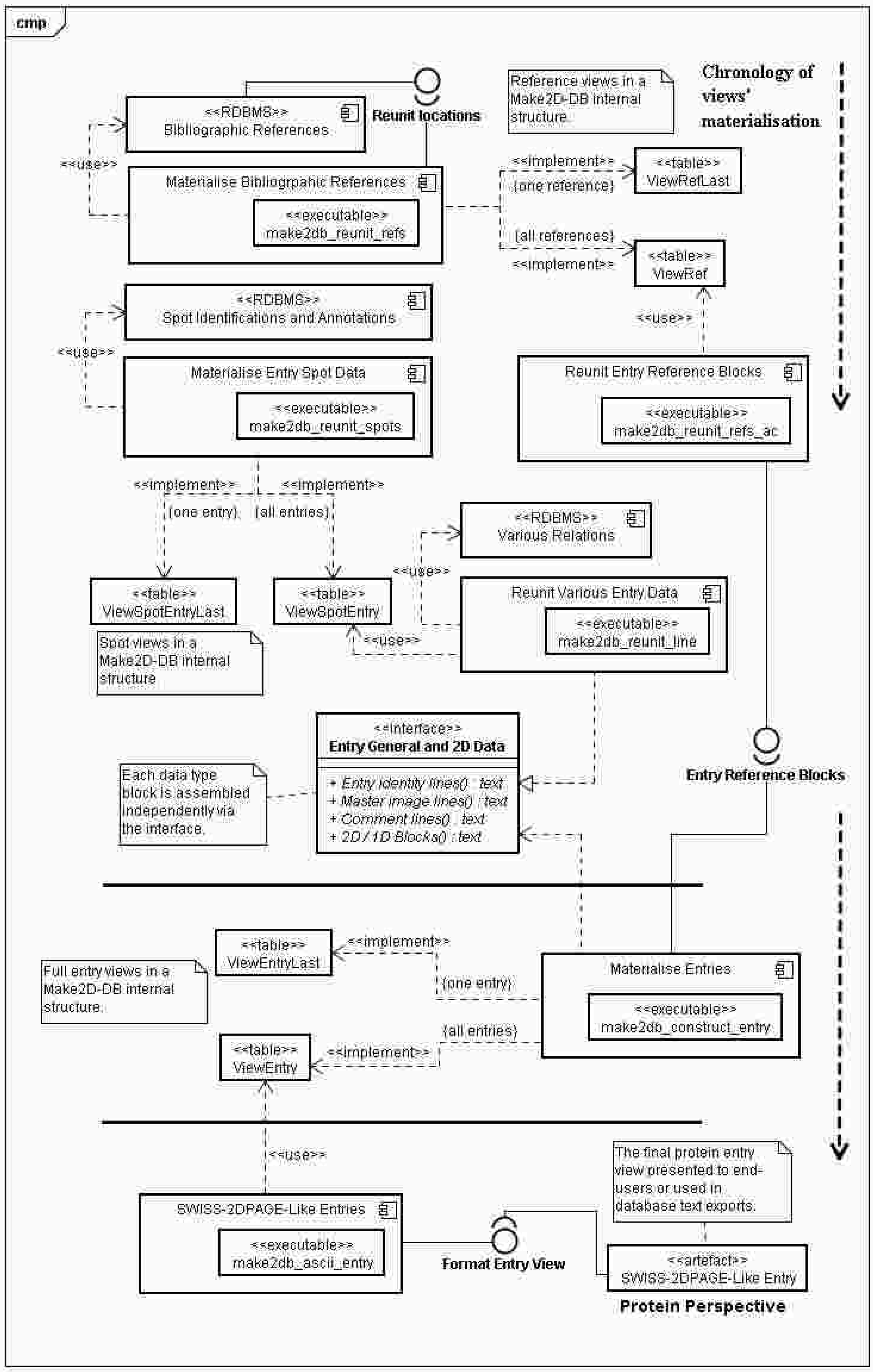

cmp
<<RDBMS>> Bibliographic References
Reunit locations
Reference views in a Make2D-DB internal structure
Chronology of views' materialisation
<<use>>
Materialise Bibliogrpahic References
<<executable>> make2db_reunit_refs
<<implement>>
{one reference}
<<table>> ViewRefLast
{all references}
<<implement>>
<<table>> ViewRef
<<RDBMS>> Spot Identifications and Annotations
<<use>>
Materialise Entry Spot Data
<<executable>> make2db_reunit_spots
Reunit Entry Reference Blocks
<<executable>> make2db_reunit_refs_ac
<<implement>>
{one entry}
{all entries}
<<RDBMS>> Various Relations
<<table>> ViewSpotEntryLast
<<table>> ViewSpotEntry
Reunit Various Entry Data
<<executable>> make2db_reunit_line
Spot views in a Make2D-DB internal structure
<<interface>> Entry General and 2D Data
+ Entry identity lines() : text
+ Master image lines() : text
+ Comment lines() : text
+ 2D / 1D Blocks() : text
Each data type block is assembled independently via the interface.
Entry Reference Blocks
<<table>> ViewEntryLast
{one entry}
Full entry views in a Make2D-DB internal structure
Materialise Entries
<<executable>> make2db_construct_entry
{all entries}
<<table>> ViewEntry
The final protein entry view presented to end-users or used in database text exports.
SWISS-2DPAGE-Like Entries
<<executable>> make2db_ascii_entry
Format Entry View
<<artefact>> SWISS-2DPAGE-Like Entry
Protein Perspective

Figure E.V-43: The materialised views components (the protein perspective).

All these views, in addition to the output of a procedure that assembles all the other categories of data within the entries, are consolidated to form raw views of protein entries. *ViewEntry* (and *ViewEntryLast*) are consolidated entries in a suitable structure for quick text parsing. To represent entries to end-users in the familiar SWISS-2DPAGE-like text format, an ultimate procedure transforms entries from *ViewEntry* into the appropriate output, *e.g.*

- http://www.expasy.org/swiss-2dpage/ac=P05090&format=raw

The Web interface is then in charge of rendering this output user-friendlier, and of enriching it with additional related data when presented to end-users.

Another mechanism is in charge of constructing materialised views in a gel perspective. The *ViewMapEntryList* assembles all identified spots, their physical properties and identification methods, as well as the corresponding proteins and their annotations for every available gel. Bibliographic references are also included in the list by their identifiers (Figure E.V-44).

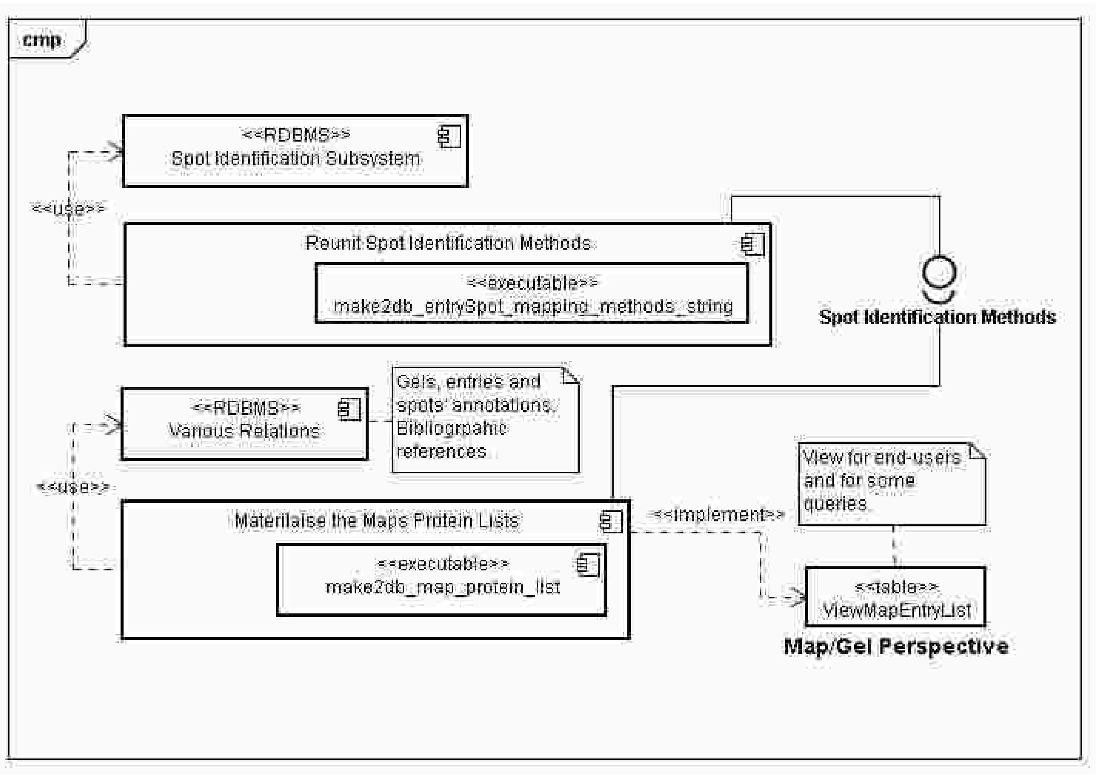

cmp
<<RDBMS>>
Spot Identification Subsystem
<<use>>
Reunit Spot Identification Methods
<<executable>>
make2db_entrySpot_mapping_methods_string
Spot Identification Methods
<<RDBMS>>
Various Relations
Gels, entries and spots' annotations, Bibliogrpahic references
<<use>>
Materilaise the Maps Protein Lists
<<executable>>
make2db_map_protein_list
<<implement>>
View for end-users and for some queries
<<table>>
ViewMapEntryList
Map/Gel Perspective

Figure E.V-44: The materialised views components (the map perspective).

Materialised views have fields in text format that are easily interpreted by humans. The fields in *ViewRef*[1], *ViewSpotEntry*[2], *ViewEntry*[3] and *ViewMapEntryList*[4] are self-explanatory, and are often very close to the familiar SWISS-2DPAGE entry structure.

Besides being a human-readable presentation of data, the materialised views are also helpful in querying the database. They reduce the complexity and the time required to perform a number of queries. For example, the SRS-like search interface at:

- http://www.expasy.org/swiss-2dpage?full

is a text-based search engine. It lets the user express SRS-like queries that are directly performed over the materialised entry views, rather than over the more complex inner schema. This is another reason why all the materialised views must be thoroughly synchronised with the database content.

E.V.11 Batch operations – An example

As already mentioned, server-side functions are nested at several levels. They constitute a network of batch operations. Figure E.V-45 illustrates the Global Update operation, which is at the top of this network. It is performed by the server-side function *make2db_update(integer,integer)*. This operation commands many other operations, which in turn command other operations. The interface presented by the Global Update operation offers the caller abstracted commands to simultaneously:

- Integrate the external data - previously collected - within the local data, and based on a user-defined level of integration (low, partial or full level).
- Update all the materialised views on the core schema (inner database content) and adapt the entries' versions.
- Export the core schema content to the public schema. The data that is made public is thoroughly filtered from any data marked private. The exported materialised views are also filtered from any such data.

Calling the Global Update operation with the appropriate parameters is as easy as executing a simple SQL command. "***SELECT make2db_update(3,1)***", for example, performs all the operations listed above, including a full level of data integration. For parameters' details, one may refer to the function documentation at:

- http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html#core.function.make2db-update-integer-integer

[1] http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html#core.table.viewref

[2] http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html#core.table.viewspotentrylast

[3] http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html#core.table.viewentry

[4] http://world-2dpage.expasy.org/make2ddb/database_schema/core_schema.html#core.table.viewmapentrylist

Only database administrators, and not the less privileged *select users*[1], have permission to execute batch operations.

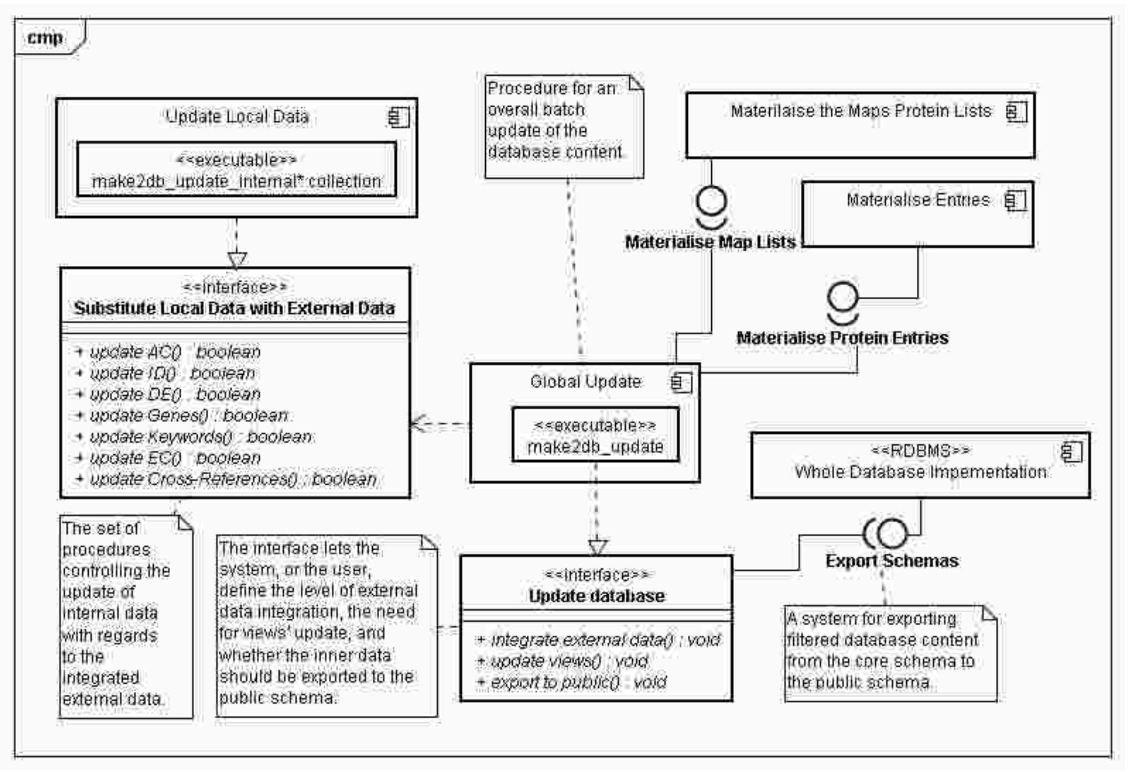

cmp
Update Local Data
<<executable>>
make2db_update_internal* collection
<<interface>>
Substitute Local Data with External Data
+ update AC() : boolean
+ update ID() : boolean
+ update DE() : boolean
+ update Genes() : boolean
+ update Keywords() : boolean
+ update EC() : boolean
+ update Cross-References() : boolean
The set of procedures controlling the update of internal data with regards to the integrated external data.
The interface lets the system, or the user, define the level of external data integration, the need for views' update, and whether the inner data should be exported to the public schema.
Procedure for an overall batch update of the database content
Materilaise the Maps Protein Lists
Materialise Entries
Materialise Map Lists
Materialise Protein Entries
Global Update
<<executable>>
make2db_update
<<RDBMS>>
Whole Database Impementation
Export Schemas
<<interface>>
Update database
+ integrate external data() : void
+ update views() : void
+ export to public() : void
A system for exporting filtered database content from the core schema to the public schema.

Figure E.V-45: The global update, an example of batch operations.

Many batch operations, such as the operation illustrated above, are accessible to the database administrators through the tool administration interface where they can be performed by simple clicks and appropriate parameter choice. The administration interface will be illustrated in the next chapter.

E.V.12 Additional ontology and controlled vocabularies

Many terms are defined and controlled inside the relational implementation by means of data types, constraints and relationships. A considerable amount of used terms is also defined and controlled outside of the relational implementation, using the Data Analyser component[2] (*cf.* F.I).

We have already come across predefined and user-defined topics, identification methods, bibliographic categories, etc.

In addition to the classical flat file format as one format for data input for the tool, other flexible formats let the user define some of his/her personal terms under some

[1] Users with no rights except performing search queries.

[2] Part of the script at http://mordor.isb-sib.ch/make2ddb/lib2d/make2db_ASCII_TABLES.pm

predefined categories. Examples of terms' definition on how data is prepared with predefined and user-defined annotations are given at:

- http://world-2dpage.expasy.org/ make2ddb/2.Readme_preparation.html#spreadsheetsMode
- http://world-2dpage.expasy.org/make2ddb/2.Readme_preparation.html#flatFile

The basic configuration file (baic_include.pl) also offers some control of vocabulary to be defined by the user.

- http://world-2dpage.expasy.org/ make2ddb/3.Readme_configuration.html#basicConfigurationFile

The mapping method description can be altered or extended to include additional identification methods (the "*mapping_methods_description*" parameter). The user can also define which terms in his/her free text annotations correspond to which predefined or user-defined identification method (the "*mapping_methods_containing*" and the "*mapping_methods_not_containing*" parameters). He/she also defines what are the main topics in his/her general annotations to be recognised as 2-DE topics (the "*SERVER::two_d_MAIN_TOPICS*" parameter).

Some other definitions require an adaptation of the relational implementation, and are therefore not directly editable by users. We may for example alter the bibliographic categories defined in the Analyser component[1] to exclude or to add new categories (any new category will require the implementation of a new dedicated class and an appropriate management). But we may also simply redefine the used vocabulary that represents each of these categories.

Finally, it is important to be aware that finding a balance between flexibility and consistency in a database implementation depends on the balance between the flexibility and the rigidity of terms that are used to define objects and concepts.

E.V.13 Metadata and technical records

Technical records are classes that contain metadata relative to the database implementation. They store metadata about the database and the Make2DDB II version in use, as well as precise information about any operation modifying the database content, including history of all data modifications.

The Release class

We have already come across the *Release* class, the class to which the *Entry* class refers when defining an entry creation release number. Let us remind the reader that the *Release* class is an independent class containing the release numbers of the successive publications of the database along with their corresponding dates, and that a database release is formally made of a release number and a sub-release number (Figure E.V-25,

[1] The "*%main::rl_categories*" variable defined within the script.

the *Release* class). For a newly created Make2D-DB II database, either the database is initialised with version 1.0, or the user explicitly defines a list of previous database publication dates if the database has been previously published in another format. Alternatively, for previously published databases, a list of prior database releases can be extracted from a flat file by parsing the date information of the different protein entries[1]. The information in this case may sometimes be incomplete, but this does not have any side effect on data consistency.

A database release is principally meant for public end-users as a milestone to signal significant changes in database content. Therefore, after a global database update and an export of the inner data to the public schema, the administrator is asked to choose to perform an increment, either on the database release, or on the database sub-release, depending on the importance of the changes[2]. Any new major release will necessarily imply an initialisation of the sub-release counter to one. The increment operations are generally managed from outside the RDBMS implementation through the administration interface.

For statistical purposes, it is suitable to display entry modifications grouped by database releases[3].

The Database class

The database class is the sole class that is not defined in the core schema (and therefore, not cloned in the public schema). Being in the common schema, it is directly accessed by both administrators and non-privileged users with no write permissions. The class has one unique instance, which contains metadata about the database (Figure E.V-46). The content of the class is used to display general facts about the database content within the associated Web interface. The class is also meant, theoretically, to unambiguously identify the database to other remote interfaces.

Much of the data within this class initially originates from the configuration parameters the user defines when he/she is installing a new database. The content is then regularly updated when the related data are modified. The administrator can also manually modify part of the class content at any moment using the administration interface.

The ***databaseIdentifier*** attribute is an identifier that is intended to be unique across all Make2D-DB II distributed databases. We already came across its signification when we described the *DynamicRemoteMake2DDBInterface*. For the moment, this attribute is initialised without checking if the same identifier is already used by another database. In the future, additional developments of the system should include a mechanism to automatically assign a unique database identifier to each 2-DE database. The advantage is to be able to clearly distinguish and to identify remote databases within the 2-DE

[1] For SWISS-2DPAGE, information on the prior database releases has been extracted this way.

[2] The administrator still has the choice of not incrementing the database release number.

[3] http://world-2dpage.expasy.org/make2ddb/database_schema/common_schema.html#common.function.make2db-release-of-date-date-boolean-boolean

network, even if some databases change their location or their name. Another advantage is to make the 2-DE network far less dependent on a central location where metadata about remote 2-DE databases have to be constantly consulted in order to maintain links between the nodes (the *DynamicRemoteMake2DDBInterface* local class will thus be in charge of the linkage). The idea is that, once the system is installed, it will send an electronic signal to the centralised integration components of Make2D-DB II located on the ExPASy server to declare itself, and to ask to be assigned a unique identifier. This identifier will therefore be sent back and will be permanently attached to the database class[1]. If the database changes its location, an export of the database content into a new location will still keep this identifier. If the database is reinstalled on the same location using the right “update option” parameters offered by the tool, the database identifier will also be safely kept, independently of any change on the database name. Technically, this mechanism is relatively easy to set up. Its implementation will depend on the usefulness that depends on the number of publicly accessible 2-DE databases, as well as on some political considerations[2]. The *databaseIdentifier* may be subject to data type change.

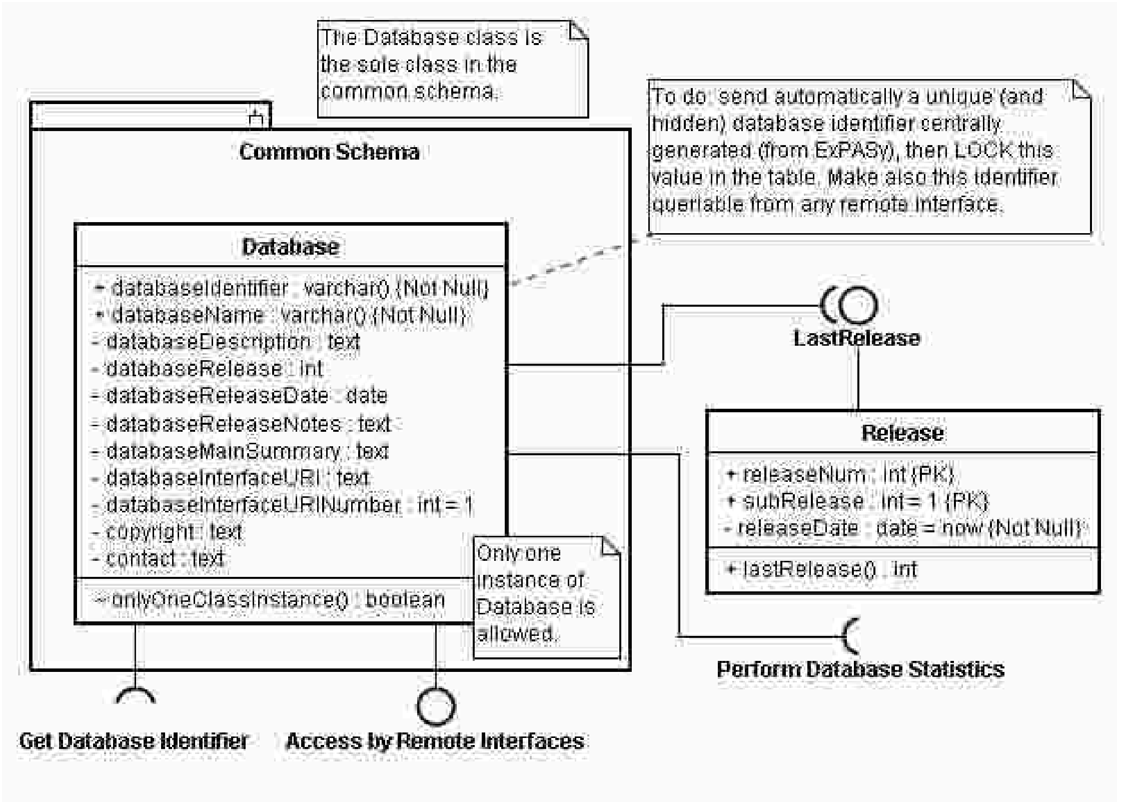

The Database class is the sole class in the common schema.
To do: send automatically a unique (and hidden) database identifier centrally generated (from ExPASy), then LOCK this value in the table. Make also this identifier queriable from any remote interface.
Common Schema
Database
+ databaseIdentifier : varchar() {Not Null}
+ databaseName : varchar() {Not Null}
- databaseDescription : text
- databaseRelease : int
- databaseReleaseDate : date
- databaseReleaseNotes : text
- databaseMainSummary : text
- databaseInterfaceURI : text
- databaseInterfaceURINumber : int = 1
- copyright : text
- contact : text
~ onlyOneClassInstance() : boolean
Only one instance of Database is allowed.
LastRelease
Release
+ releaseNum : int {PK}
+ subRelease : int= 1 {PK}
- releaseDate : date = now {Not Null}
+ lastRelease() : int
Perform Database Statistics
Get Database Identifier
Access by Remote Interfaces

Figure E.V-46: The Database common class.

[1] Technically, it is even possible to “LOCK” the value within the table, so it will not be altered afterwards.

[2] Like agreeing on one representative authority to definie proteomics database identifiers.

In addition to the *databaseIdentifier*, the *Database* class contains a *databaseName* attribute, which is the name given by the database administrator to his database, a short free text description of the database, as well as the database release number and date, extracted from the *Release* class, and with optional comments regarding the current release. Additional information covers the database associated Web interface URL, and its assigned identifier.

The administrator interface generates statistical facts about the database content exported in the form of a HTML document. This document is stored in the *databaseMainSummary* attribute. We have recently decided to include a simple approach of querying a database about its statistics in order to automatically import them in any 2-DE portal built with our tool[1]. As we will see later, a Make2D-DB II portal is a Web interface that interacts with any number of remote 2-DE interfaces. The portal sends queries to the remote interfaces, all at once, and displays the results consolidated to the end-user. Displaying metadata and statistical facts about remote databases within a portal homepage is a very convenient way to describe the origin of the data being collected by the portal.

The Make2DDBTool class

This class contains technical information about the Make2D-DB II version used in building the system: the version, the subversion and the version date of the tool, as well as the PostgreSQL version managing the system. The *action* attribute is one of three possible actions performed by the tool: <*create*> (create a new database from scratch), <*transform*> (convert data from another format) or <*update*> (update data or tool on an already installed database). The class plays an important role in interconnecting remote 2-DE databases, as it informs any calling interface about the version of the local database. The structure of the database is version dependent. Even if remote interfaces never access directly the inner structure of a database - remote queries are addressed in an abstract manner – knowledge of the database version informs if a specific query can or can not be performed, as older versions may not support some newer queries.

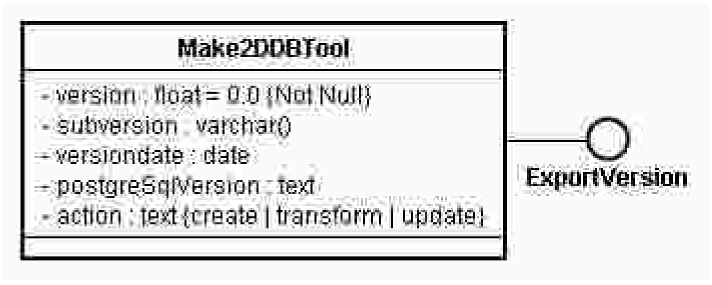

Make2DDBTool
- version : float = 0.0 {Not Null}
- subversion : varchar()
- versiondate : date
- postgreSqlVersion : text
- action : text {create | transform | update}
ExportVersion

Figure E.V-47: Make2D-DB II tool information class.

[1] Until Make2D-DB II version 2.50.2, statistics and database metadata were only displayed interactively. Starting from version 2.60.1, we will be able to export them into a remote interface using the generic URL syntax: http://someServer.org/databaseName?stats&extract

When connecting to a distant database, an interface interrogates at first the database version, so it knows which queries it can correctly handle, *e.g.*,

- http://www.expasy.org/swiss-2dpage?make2d&extract

Operation dates and history of data modifications

We have previously mentioned that each relation in the core schema, except the materialised views, has two extra attributes: the *userStamp* and the *update* fields, giving the name and the date when a tuple has been inserted or modified within the relation.

We have also mentioned that an automatic mechanism records into the log schema every modification that occurs in the core schema. All the core relations are automatically cloned in the log schema with three additional attributes that record the modification date, the modifier name and whether the tuple has been modified or deleted. Having a place to store every modification within the database is an additional security feature to the classic regular database backups. However, it will require a person familiar with relational systems and with the data model of the tool to be able to manipulate and restore the data

E.VI. Implementing a working system out of the concepts

We have presented in this chapter the main motivations and the ideas that have led us to conceptualise and implement the Make2D-DB II tool. The ideas have been translated in terms of a physical and evolving data model interacting with its environment.

The next chapter is dedicated to the description of the pieces of the puzzle, that when joined together, constitute the whole image we have in mind of a global integrative 2-DE network. We will be able to describe in details the major components of the tool, how they are implemented and how they interact with each other. We will describe how the tool can be configured and installed, how data is converted into a local database, how distributed data is integrated into the system and how it is managed by users. We will illustrate afterwards the many functionalities of the Web interface, which acts both interactively, as a search engine to query local and remote 2-DE databases, as well as a Web node capable of interconnecting with any similar node to import and export data between remote installations, thus forming a global virtual 2-DE database.

Chapter **F**

CHAPTER F. MAKE2D-DB II ENVIRONMENT: COMPONENTS AND IMPLEMENTATION

This chapter describes how Make2D-DB II is physically implemented, how the distributed systems interconnect, and how data is integrated.

F.I. The complete image

Make2D-DB II can be schematically divided into two major parts (Figure F.I-1):

- **The Make2D-DB II distributed package**: This package implements the "Installation and Data Conversion" components, the "Local 2-DE Databases" components and the "Local 2-DE Web interfaces" components. The package ensures the interconnection between remote 2-DE implementations, and helps to set up personalised 2-DE Web portals anywhere on the Web.

- **The central data integration mediator**: Integration of remote non 2-DE data is administered by a mediator residing on the ExPASy server. This is also where remote 2-DE databases can be "registered" to become automatically visible to each other.

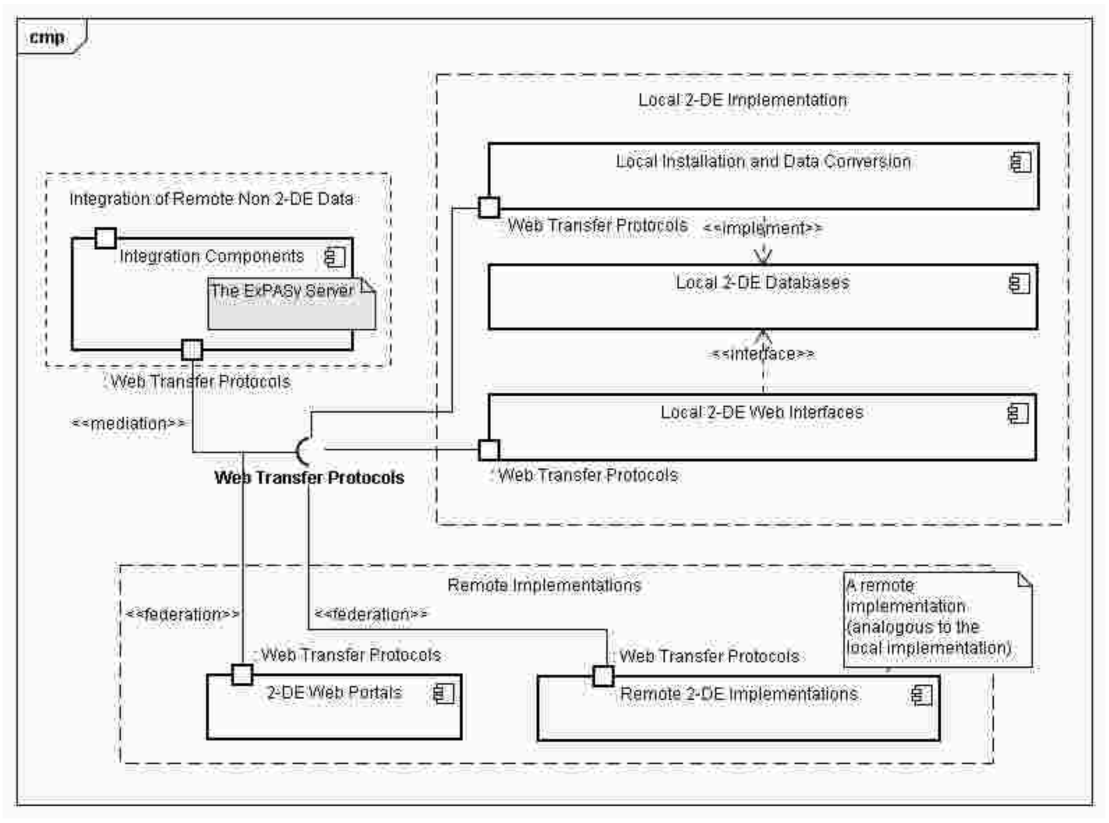


Figure F.I-1: The Make2D-DB II environment.

A closer examination of the major elements gives us a larger insight into the inner components of the global environment and their interactions (Figure F.I-2).

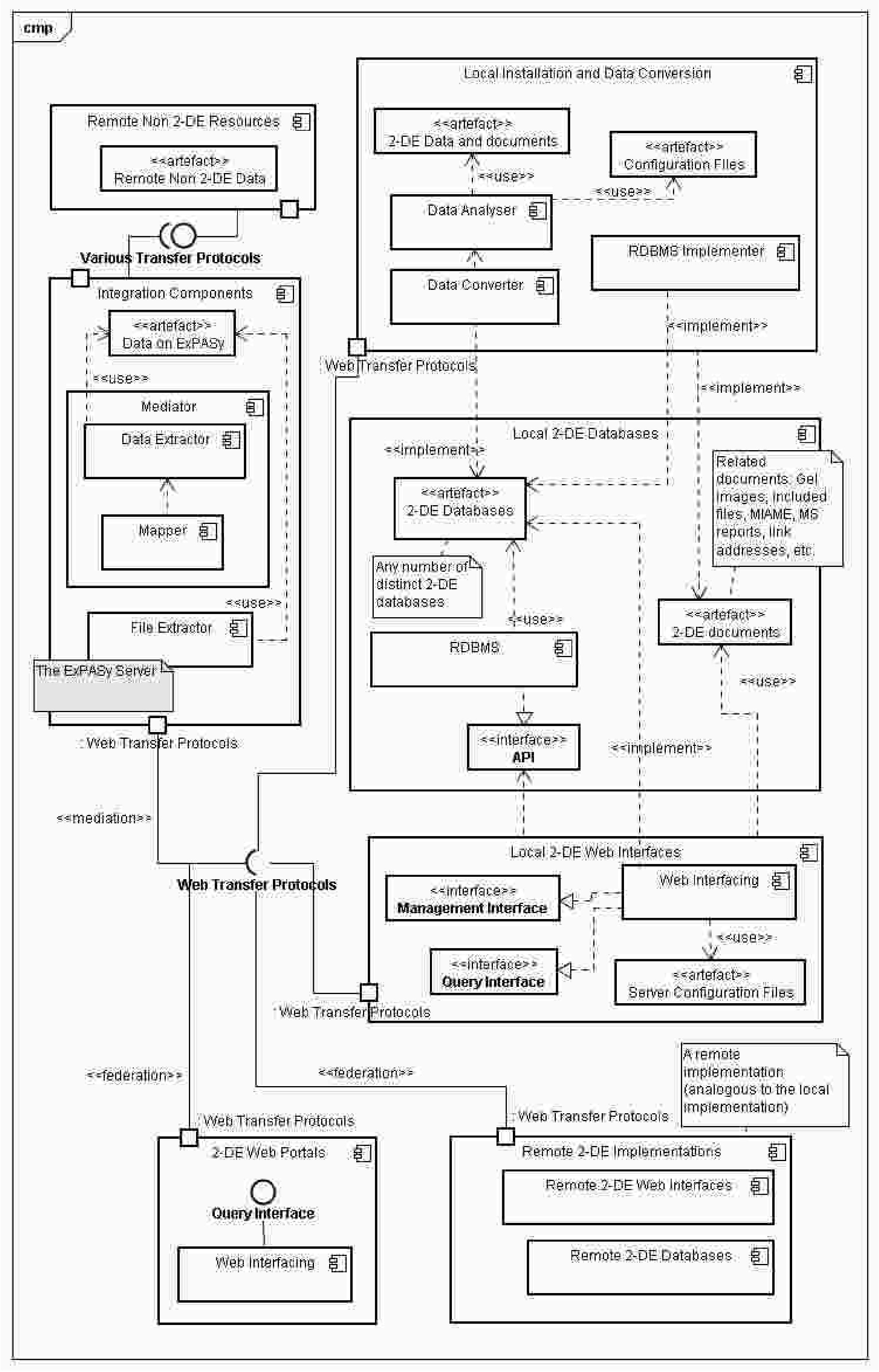

cmp
Local Installation and Data Conversion
<<artefact>> 2-DE Data and documents
<<artefact>> Configuration Files
<<use>>
<<use>>
Data Analyser
RDBMS Implementer
Data Converter
<<implement>>
Remote Non 2-DE Resources
<<artefact>> Remote Non 2-DE Data
Various Transfer Protocols
Integration Components
<<artefact>> Data on ExPASy
<<use>>
Mediator
Data Extractor
Mapper
<<use>>
File Extractor
The ExPASy Server
Web Transfer Protocols
Web Transfer Protocols
<<implement>>
Local 2-DE Databases
<<implement>>
<<artefact>> 2-DE Databases
Related documents: Gel images, included files, MIAME, MS reports, link addresses, etc.
Any number of distinct 2-DE databases
<<use>>
RDBMS
<<artefact>> 2-DE documents
<<use>>
<<interface>> API
<<implement>>
<<mediation>>
Local 2-DE Web Interfaces
Web Transfer Protocols
<<interface>> Management Interface
Web Interfacing
<<use>>
<<interface>> Query Interface
<<artefact>> Server Configuration Files
Web Transfer Protocols
<<federation>>
<<federation>>
A remote implementation (analogous to the local implementation)
Web Transfer Protocols
Web Transfer Protocols
2-DE Web Portals
Query Interface
Web Interfacing
Remote 2-DE Implementations
Remote 2-DE Web Interfaces
Remote 2-DE Databases

Figure F.I-2: Details of the Make2D-DB II environment.

F.II. The Make2D-DB II distributed package

The components contained in the Make2D-DB II distributed package are sufficient for a complete local 2-DE database installation. All the elements needed for the conversion of data, the database implementation and its publication on the Web are supplied, along with the elements responsible for the interconnection with other similar remote 2-DE databases and the ExPASy-resident components.

Details on the package are provided on the package Web server at:

- ❖ http://world-2dpage.expasy.org/make2ddb/

F.II.1 Installation process

In this section, we will explain how a user prepares his/her system, how he/she writes data and how he/she launches the appropriate commands to convert this data into a Make2D-DB II local database. The installation process is fully described at:

- ❖ http://world-2dpage.expasy.org/make2ddb/1.Readme_main.html#installation

and

- ❖ http://mordor.isb-sib.ch/make2ddb/readme/Technical-Implementation.doc

Prerequisites

Make2D-DB II runs on any Unix or Unix-type operating system (Linux, Solaris/SunOS, BSD, Irix). The environment relies on some few free components that must be installed on one's system[1]. The prerequisite components are: a Perl[2] interpreter, a PostgreSQL server[3] and an Apache HTTP server[4]. All these components are present by default in many operating systems, such as standard Linux distributions. Some few free Perl modules are also required. Other additional components (C libraries and Java interpreter) are optional and serve to extend some features of the tool. The user must prepare the PostgreSQL server as described in the tool's manual.

Overview of the tool's options

The tool runs with several different options:

[syntax: perl make2db.pl –m *option*]

- ➢ ***<config>***: To set up the configuration files for both the database installation and the Web interfaces.

[1] http://world-2dpage.expasy.org/make2ddb/1.Readme_main.html#before

[2] http://www.perl.org/

[3] http://www.postgresql.org/

[4] http://www.apache.org/

- ***<check>***: To check the syntax and the consistency of the provided data. Corrections are proposed when an error is encountered. The process is interrupted when a major error is met.

- ***<check-report>***: Same as the <*check*> option, except that the process continues even if a major error is encountered. This option is helpful to report all detected errors in a final report without any interruption of the program.

- ***<create>***: To implement the relational schema for a new database. A void database is created if no data is provided.

- ***<update>***: For updates of the database schema structure, the interfaces and the database content when adding extra data or updating to a Make2D-DB II newer version.

- ***<server>***: To prepare and install the files for the Web server. This option can also be used independently, without any 2D-PAGE data, in order to host a local Web-portal to query remote databases.

- ***<transform>***: A shortcut to combine the <*check*>, the <*create*> and the <*server*> options.

Information regarding the tool is obtained using the syntax:

[perl make2db.pl –*info*], where *info* is:

- ***<help>***: To display the list of possible options and the package's metadata.

- ***<version>***: To display the current version of the tool.

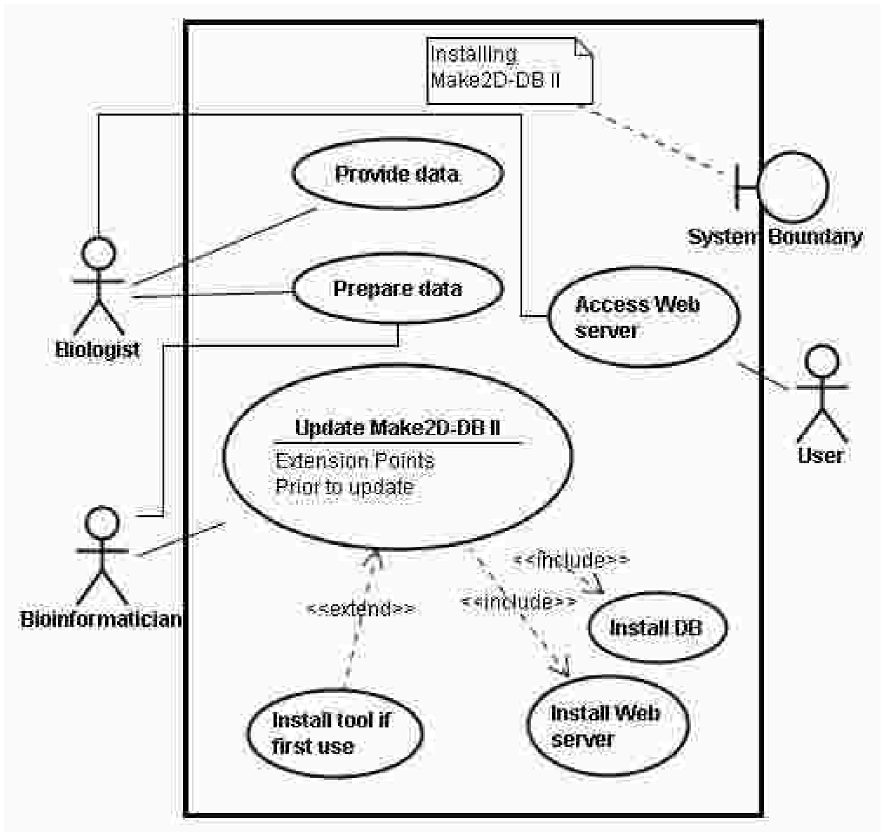


Figure F.II-1: Installing the Make2D-DB II distributed package.

Schematically, for a standard installation, the user provides his/her data in a recognised format. He/she configures the many parameters that define how the tool should behave, how the data should be interpreted, which external data should be integrated, and how the integration should be performed. In Figure F.II-1, the person that is in charge of the installation process is referred to by b*ioinformatician*. This person is responsible for preparing the data provided by the *biologists* in a format that is understood by the system. He/she is also responsible for configuring the system parameters to properly suit the desired installation. Subsequent tasks are the conversion of data into the relational database, the installation of the Web interfaces and, in some cases, the update of the whole system. These tasks are essentially managed by the tool itself, according to the configuration parameters given by the bioinformatician. An *end-user* interacts with the system only by way of the Web server to address his/her queries.

F.II.2 Data preparation: formats and annotations

Preparing the data

Except for void databases, the user must prepare his/her data conform to the instructions given in the data preparation document at:

- ❖ http://world-2dpage.expasy.org/make2ddb/2.Readme_preparation.html

This document describes in details all the different input formats, including the annotation structure and syntax to use. A brief summary of how data is prepared as described in the readme document is given hereafter.

Maps and their annotations

Images of maps and their corresponding thumbnails must be provided. Each map is annotated following a precise format. Some of the annotations are mandatory, while some are optional. Annotations cover the following topics:

- Map identifiers and descriptive names
- Image dimension (including potential axis and origin's shifting)
- pI and Mw ranges
- Species and tissue definition
- List of identification methods applied to the whole map
- URL addresses and/or local file paths describing the preparation protocol and the informatics part
- The software used for the detection of the spots and the number of detected spots
- Comments

Map annotations are in text format[1]. They can be generated interactively by running the tool configuration utility (using the <*config*> option).

[1] Example: http://world-2dpage.expasy.org/make2ddb/examples/existing.maps.detailed.example

An example of generated map annotations:

```
PLASMA
longname:Plasma Test Gel
x:424
y:431
x-shift:0
y-shift:0
pi-start:4.0
pi-end:8.0
mw-start:15000
mw-end:40000
taxid:9986
strain:Oryctolagus cuniculus crossed
tissue:Cervical
preparation-uri:http://www.expasy.org/ch2d/protocols/protocols.fm2.html
informatics-uri:http://www.expasy.org/ch2d/protocols/protocols.fm4b.html
preparation-doc:/Make2D-DB_II/data_test/gel_doc/SDSprotocol.html
informatics-doc:/Make2D-DB_II/data_test/gel_doc/GelInformatics.gif
comment:The preparation procedure is fictive (for demonstration)
informatics-comment:The spots phyiscal properitites are fictive
software:ImageMaster 2D Platinum and Melanie 6.0
mapping:Gm,PMF
```

The spot and the protein annotations

There are actually three manners to provide spot and protein annotations. The user can choose between:

- **Flat file**[1] (listing sequentially the protein entries)
- **Spreadsheet**[2] (CSV / tab-delimited text files, *e.g.*, Excel exports)
- Melanie / ImageMaster™ 2D Platinum **XML exports**[3]

Both the flat file and the spreadsheet modes offer a high degree of annotation that we fully describe in the data preparation document. The flat file format, a protein-centric perspective, is a SWISS-2DPAGE-like format with extended annotations specific to Make2D-DB II. The spreadsheet mode, a spot-centric perspective, has a wide list of predefined annotation categories, in addition to any number of user-defined categories. Melanie XML exports (Figure B.III-4) are essentially lists of spot identifiers, with appropriate coordinates, intensities, and identified proteins. The Melanie exports may lack high-level annotations, but users can combine Melanie exports with flat files to extend annotation. A flat file non-combined with a Melanie XML export requires associated reports that define the spots' positions, their physical properties and their relative intensities. In both the spreadsheet and the Melanie XML modes, the tools generates an intermediate flat file that users can manually edit in order to refine their original annotations or to add extra annotations that are specific to the flat file mode[4].

[1] Example: http://mordor.isb-sib.ch/make2ddb/data_test/examples/test.dat

[2] Examples: http://world-2dpage.expasy.org/make2ddb/examples/PLASMA.excel_example.txt and http://world-2dpage.expasy.org/make2ddb/examples/PLASMA2.excel_example.txt

[3] *cf.* Figure B.III-4

[4] By editing the generated *last_generated_flat_file.dat* file in the *data* directory, and then switching to the flat file mode.

By providing various identifiers that are recognised by the tool - such as UniProt-like accession numbers, cross-links to UniProtKB entries, biosource definition and NCBI taxonomy values - users ensure their annotations are enriched with additional information, which the tool automatically collects and integrates into the database. The same type of identifiers makes the database content also accessible and queriable by other remote 2D-PAGE databases.

The flat file mode

The flat file is structured to be readable by humans, as well as by computer programs. It is a protein-centric perspective. The different lines describing one entry begins with a two-character line code, which indicates the type of data contained in this line. The remaining part of the line follows some specific rules that depend on the type of data. Specifications to build a flat file are fully described at:

- ❖ http://world-2dpage.expasy.org/make2ddb/2.Readme_preparation.html #flatFileMode

These specifications extends the ones described in the SWISS-2DPAGE user manual:

- ❖ http://www.expasy.org/ch2d/manch2d.html

The flat files may vary greatly regarding the quality and the quantity of the annotations they contain. Many data types are not explicitly required. Default values for missing mandatory data types can be set up within separate configuration files (e.g., bibliographic references, studied species, etc.). Some annotations, in particular those covering the identification methods, belong to a controlled vocabulary and are based on user-defined definitions.

In addition, users are asked to provide spot reports that give, for each spot, an identifier and a location on a gel. The physical properties of the spots are given either within these reports or within the flat file. The spots' relative optical densities and volumes are optional. Most 2D-PAGE image analysis software are able to export this kind of reports. Table F.II-1 is an example of a flat file[1].

[1] A more extended example of a flat file is given at the address:
http://world-2dpage.expasy.org/make2ddb/examples/test.dat

Table F.II-1: An example on an annotated Make2D-DB II flat file.

```
TEST_2DPAGE Release 3
Feb 2005

AC   P12345;
2D   -!- MASTER: PLASMA;
2D   -!-   PI/MW: SPOT 397=4.43/32375;
//
AC   ZI|GI.MINIMAL;
2D   -!- MASTER: PLASMA;
2D   -!-   PI/MW: SPOT 397=4.43/32375;
DR   Swiss-Prot; P00853; -
DR   NCBI_Protein; 31543902; -
//
AC   Z02760;
OX   NCBI_TaxID=9606;
IM   PLASMA; PLASMA-MAP1; PLASMA-MAP2; LIVER; LIVER-MAP1; LIVER-MAP2.
RN   [1]
RP   MAPPING ON GEL.
RX   MEDLINE; 78094420.
RA   Anderson N.L., Anderson N.G.;
RT   "High Resolution 2-DE of human Liver";
RL   Proc. Natl. Acad. Sci. U.S.A. 74:5421-5425(1977).
2D   -!- MASTER: PLASMA;
2D   -!-   PI/MW: SPOT 111=4.89/32122;
2D   -!-   PI/MW: SPOT 112=4.97/33422;
2D   -!-   MASS SPECTROMETRY: SPOT 112: [2] 669.468 (1.09); 324.448 (2.67);
2D         435.708 (3.17); 512.129 (1.2); 517.129 (1.9), 598.345 (2.17) :
2D         469.468 (1.59), 524.448 (2.17), 571.432 (1.08);
2D         file:/some_path/msms.pkl ident-file:/some_path/msIdentResults.dat
2D         uri:http://www.ebi.ac.uk/pride/experimentAccessionNumber=someID
2D         ident-uri:http://www.ebi.ac.uk/pride/experimentAccessionNumber=ID
2D   -!-   PEPTIDE SEQUENCES: SPOT 112: (R)SLDMoxDSIIAEVK(A),252-263;
2D         (R)ASLEAAIADAEQR(G),328-340; (R)LEGLTDEINFLR(Q),213-224 {private}.
2D   -!-   PATHOLOGICAL LEVEL: Increased during renal insufficiency [1];
2D         SPOT 112: Decreased during renal insufficiency.
2D   -!- MASTER: LIVER;
2D   -!-   PI/MW: SPOT 100=5.05/42200;
DR   UniProtKB/Swiss-Prot; P02760; -
//
ID   My Favourite Protein; PRELIMINARY 2DG.
AC   P02990;
2D   -!- MASTER: PLASMA;
2D   -!-   PI/MW: SPOT 397=4.43/32375;
//
…
```

The spreadsheet mode

By using the spreadsheet mode, users have the choice to work with a large range of pre-defined annotations, as well as with any number of user-defined free text annotations. The spreadsheet mode means any text report with fields separated by tabulators (tab-delimited files / CSV), such as spreadsheet software exports (*e.g.*, Excel exports). It is a spot-centric perspective, as opposed to the protein-centric perspective of the flat file mode. Details on the spreadsheet mode are given at:

❖ http://world-2dpage.expasy.org/make2ddb/2.Readme_preparation.html #spreadsheetsMode

A separate report should be given for each map. There are three distinguishable categories of data. A category is recognised thanks to the header that is given on top of each column. The three categories are:

- The *mandatory* types (required)
- The *pre-defined* types (optional and syntactically controlled data)
- The *free text* types (types that are defined by the user)

Various simple rules apply in order to append, for example, several distinct annotations of a similar type to the same spot, or to affect the same annotation to all spots belonging to the same entry or to all spots belonging to the whole map.

The spreadsheet mode is much more flexible than the flat file mode. For example, users may list their bibliographic references in a separate file[1], and refer to these references by the identifiers they are given (*e.g.*, Table F.II-2). The following three Web links provide an outline of the different possibilities to annotate a map in this mode:

- http://world-2dpage.expasy.org/make2ddb/examples/PLASMA.excel_example.txt
- http://world-2dpage.expasy.org/make2ddb/examples/PLASMA2.excel_example.txt
- http://world-2dpage.expasy.org/make2ddb/examples/PLASMA3.excel_example.txt

[1] Example: http://world-2dpage.expasy.org/make2ddb/examples/reference.txt

Table F.II-2: An example of annotations in Make2D-DB II spreadsheet mode.

SPOT	89	111	112
X	400	450	390
Y	420	390	690
pI	4.12	5.76	4.78
Mw	25,400	27,100	33,100
Od	0.992	0.656	0.356
Volume	1.231	0.567	0.498
AC	P02760	Q99TY2	P13693
Mapping Methods	PMF, Im, AA, Co	MS/MS, Gm	MS/MS, Gm
PMF	769.46 (1.09); 824.48 (2.67)… TRYPSIN (private)		
MS		[1001.5:2] 669.46 (4.09); 324.48 (2.67)…: 512.19 (1.2);…	
MS FILE		Some/path/msms.mgf	mzdata: mzdata1.05.xml
MS IDENT-FILE		Some/path/msIdentResults.dat	MascotReport.dat
MS URI		www.ebi.ac.uk/pride/directLink.do?experimentAccessionNumber=11	
Peptides		(R)SLDMoxAEVK(A), 254-263 (S)RLDMoxDVA(E), 225-233	
Amino Acid	B=9,Z=6.7,S=3.7, …		
Expression	Increased During the Acute-Phase	*High expression in strain	
***Positional Variants**		Some rare variants	
Comment: Subunit	Homodimer (Similarity)		
REFERENCE	1	1,2	1

The Melanie / ImageMaster^{TM} 2D Platinum XML exports

The tool will parse all found XML files generated by the Melanie software in order to extract spot locations, shapes, physical properties, and corresponding identified proteins. This is described at the following address:

❖ http://world-2dpage.expasy.org/make2ddb/2.Readme_preparation.html #MelanieMode

To enhance the annotation possibilities, relatively limited XML exports can be combined with a highly annotated flat file.

An example of an annotated Melanie XML export:

```
<Gels>
      <Fileinfo>
            <Type>Export</Type>
            <Version>1.0</Version>
            <User>mostaguir</User>
            <Date>17/02/2005 18:54:06</Date>
            <Application_Name>ImageMaster & Melanie Viewer</Application_Name>
            <Application_Version>5.0.2.0</Application_Version>
            <Organisation/>
      </Fileinfo>
      <Gels_Data>
            <Version>1</Version>
            <Gel Id="2a9aecf7-b491-4bd1-9d07-78529675b0f2" Cols="424" Ref.="2" Rows="431" Class=""
                  Spots="516" Caption="a" GelName="Y:\Make2d\Make2D-DB_II\data_test\PLASMA"
                  MaxGray="2189.00" MinGray="90.0000" PixWidth="175" PixHeight="175" Gray_Slope="1.00000"
                  Annotations="10" Gray_Offset="0" IsSynthetic="0" Selected_Spots="0" Selected_Annotations="0">
                  <Gel_Properties/>
                  <Spots>
                        <Spot X="38" Y="5" Id="1" Mw="247654" Pi="3.54634" Vol="223.042" Area="0.704375"
                              Flag="0" X_Align="0" Y_Align="0" Saliency="0" Intensity="611.000"
                              percentVol="0.318571" percentIntensity="0.548996"/>
                        <Spot X="46" Y="103" Id="111" Mw="98320" Pi="3.68293" Vol="90.8031" Area="0.765625"
                              Flag="0" X_Align="0" Y_Align="0" Saliency="0" Intensity="261.000"
                              percentVol="0.129694" percentIntensity="0.234514">
                          <Annotations>
                              <Annotation Id="4">
                              <Label Value="P02760" Category="Ac"/>
                              <Label Value="HC_HUMAN" Category="ProteinName"/>
                              </Annotation>
                           </Annotations>
                        </Spot>
                  ....
                  </Spots>
                  <Annotations>
                        <Annotation X="6" Y="4" Id="1">
                              <Label Value="l1" Category="Landmark"/>
                              <Label Value="3.00 250000" Category="pI_MW"/>
                        </Annotation>
                  </Annotations>
            </Gel>
      </Gels_Data>
</Gels>
```

Comments on annotations

In all the three previous modes, users have the possibility to include additional local documents and Web addresses (pointers) for external documents to enrich their database annotations. Documents and pointers will be available from within the tool's Web interface. Users also control which data is public and which data is restricted to privileged users For MS analysis, the tool directly extracts the peak list values from a range of various output document formats[1].

F.II.3 The configuration files

The user has a large control over the behaviour of the tool regarding the databases and the Web servers' installations. The configuration process is launched by running

[1] Various examples at http://mordor.isb-sib.ch/make2ddb/data_test/ms_files/

the tool's configuration utility[1] (using the <*config*> option). This interactive process guides the user to generate two distinct configuration files: the main configuration file ***include.cfg***, and the server configuration file ***2d_include.pl***. Both files are required to install a new database and to set up a corresponding Web server. A third configuration file, named ***basic_include.pl***, is distributed with the package and do not need to be generated. This file contains some global definitions and controls the behaviour of the tool. It is generally not necessary to edit this file, except for non-standard installations. The full description of the configuration process and all related parameters is given at:

- ❖ http://world-2dpage.expasy.org/make2ddb/3.Readme_configuration.html

The configuration process is also helpful to annotate the different maps, to modify or to update an already installed database, to reconfigure an already running Web server, as well as to create a new Web portal (an entry point to access several remote interfaces and databases at once).

The main configuration file: *include.cfg*

This file defines how to perform the database creation process.

- ❖ http://world-2dpage.expasy.org/make2ddb/3.Readme_configuration.html #mainConfigurationFile

The user gives all technical parameters regarding the PostgreSQL database installation. He/she also defines the kind of data to be converted, the files' location, and the mode to use. A whole range of particular data types can be attributed default values to use with the source data. This file also defines where to install the Web server interfaces.

An example of a generated main configuration file, listing the different parameters that can be configured, is given at:

- o http://world-2dpage.expasy.org/make2ddb/examples/example.include.cfg

The server configuration file: *2d_include.pl*

Each Web server needs a specific server configuration file that controls its behaviour.

- ❖ http://world-2dpage.expasy.org/make2ddb/3.Readme_configuration.html #serverConfigurationFile

This file defines what the Web interface should look like, and how it should act. Titles, shape, colours, icons, and contact information can be defined, as well as map images' shifting, private data passwords, copyright messages and external data visibility. The user also defines technical parameters related to the Apache installation,

[1] http://mordor.isb-sib.ch/make2ddb/lib2d/make2db_CFG.pm

and he/she states if desired to activate the URL redirection rules[1]. More importantly, as several local or remote databases can be simultaneously accessed by the same Web interface, the user defines a list of databases to connect to, along with their connection parameters.

An example of a generated server configuration file, listing the different parameters that can be configured, is given at:

- http://world-2dpage.expasy.org/make2ddb/examples/example.2d_include.pl

After the installation of the Web interface, the user is free to modify this configuration file. For example, if he/she wishes to adjust or alter the look and feel of the Web interface. At any moment, he/she may also add any newly created database (or remove an already accessible database) to the already running Web server. This is useful since a Make2D-DB II interface has the ability to simultaneously access several databases.

The basic configuration file: *basic_include.pl*

In addition to the previous two user-generated configuration files, a third configuration file defines additional parameters that control the overall behaviour of the tool.

❖ http://world-2dpage.expasy.org/make2ddb/3.Readme_configuration.html #basicConfigurationFile

The file is subdivided in three editable sections: the ‘General Behaviour’, the ‘Global Definitions’, and the ‘Query remote interfaces’ sections. In most cases, the user does not need to modify the proposed values.

The ‘General Behaviour’ section

This is where specific remote resources can be included or excluded from the data integration process. The user also defines the level of integration and data replacement. Specific features, essentially technical issues controlling the Web interface behaviour, can be turned on or off from within this section.

The ‘Global Definitions’ section

In this section, the identity of various components of the package is defined. This includes URL addresses for some external components, and a list of different patterns and keywords that are used when reading and generating data in text formats. More important, the user may redefine the different ontologies and controlled vocabularies that cover the identification methods, the main annotation topics and the bibliographic categories, in order to make them correspond to his/her personal definitions. Finally,

[1] To rewrite URL addresses at the Web server level based on a few transformation rules that let the server understand meaningful addresses, *e.g.*,: http://some-domain/my_2dpage/map=plasma

many technical database optimisation parameters can be controlled from this particular section.

The 'Query remote interfaces' section

Any Make2D-DB II Web server has the ability to access and query, in addition to local databases, any public remote Make2D-DB II Web server (or interface) available on the Web. The interconnection is performed by means of simple HTTP protocols. This section offers the user the possibility to define his/her personalised list of remote servers and databases. The basic configuration file ends with a fourth section that only developers and experimented users may edit.

A copy of a standard basic configuration file is given at:

- http://world-2dpage.expasy.org/make2ddb/examples/example.basic_include.pl

After the initial installation process, all the configuration files are copied into the Web server directories. By modifying the files within the server directories, a user may change - at any moment - many of the initial definitions.

F.II.4 The Data Analyser and Converter[1]

Analysing and converting the data provided by users is a complex process. The first step is the analysis of the data at the syntax and lexical levels. During the process of analysing the data significance and consistency, The Make2D-DB II tool employs an inner data conversion. This conversion transforms the user-provided data, coming either from flat files or from spreadsheets, into a format that is a combination of a plain text format - similar to a flat file like format that we keep for historical reasons - and a Perl data structure. This kind of structure is convenient to carry out the analysis process, due to the complexity of the data we are dealing with, and due to the type of verifications that we have to perform over this data.

The Data Analyser component and the Data Converter component are Perl modules working concurrently (Figure F.II-2).

Reading the configuration files and controlling the presence of the required data

The tool starts by reading the configuration parameters defined by the user. It then makes sure all the needed files are present and follow the expected formats.

Updating the external documents

The distributed package contains a set of supporting documents, among which a list of tissue names and their aliases, as well as a list of cross-reference databases and their contact parameters. The tool verifies that these documents are recent enough. If they are not, a signal is sent to the integration mediator on the ExPASy server, and up-to-date versions of these documents are imported locally.

[1] http://mordor.isb-sib.ch/make2ddb/lib2d/make2db_ASCII_TABLES.pm

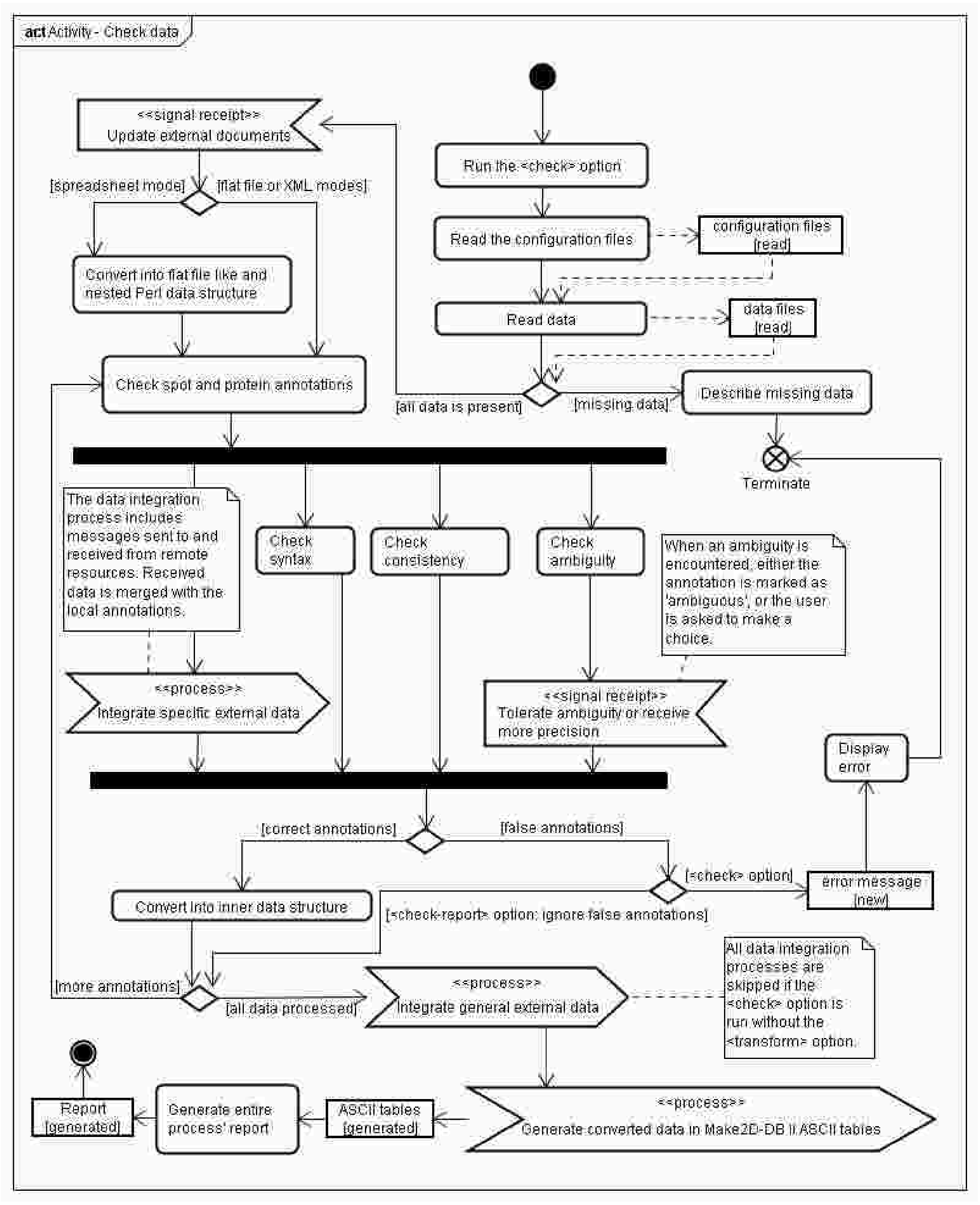

act Activity - Check data
<<signal receipt>> Update external documents
[spreadsheet mode]
[flat file or XML modes]
Convert into flat file like and nested Perl data structure
Check spot and protein annotations
Run the <check> option
Read the configuration files
configuration files [read]
Read data
data files [read]
[all data is present]
[missing data]
Describe missing data
Terminate
The data integration process includes messages sent to and received from remote resources. Received data is merged with the local annotations.
Check syntax
Check consistency
Check ambiguity
When an ambiguity is encountered, either the annotation is marked as 'ambiguous', or the user is asked to make a choice.
<<process>> Integrate specific external data
<<signal receipt>> Tolerate ambiguity or receive more precision
Display error
[correct annotations]
[false annotations]
[<check> option]
error message [new]
Convert into inner data structure
[<check-report> option: ignore false annotations]
[more annotations]
[all data processed]
<<process>> Integrate general external data
All data integration processes are skipped if the <check> option is run without the <transform> option.
Report [generated]
Generate entire process' report
ASCII tables [generated]
<<process>> Generate converted data in Make2D-DB II ASCII tables

Figure F.II-2: Data analyser and converter - the <*check*> option.

Generating structured entries

Before analysing the data content, and in order to process the entries sequentially, the tool rearranges and merges the original data. It then converts it into a rich plain text format combined with a nested Perl data structure. Processing the entries sequentially is a historical choice that originates from the SWISS-2DPAGE-like protein perspective. This process also includes the integration of the content of some optional user-provided documents, *e.g.,* the mass spectrometry files, whose content is parsed and integrated within the structured entries[1]. In the spreadsheet mode, the generated entries are physically stored in the work directory so that the user may intervene on them and apply any additional modification.

At this point, the data is progressively checked, entry after entry, for its syntax and its consistency with previously analysed entries. Some flexibility is considered regarding the syntax and the completeness of the information.

Checking the syntax

The data is syntactically controlled thanks to an extended syntax checker. Except for the content of free text annotations, each type of information must be understood by the tool. If the tool fails to recognise the nature of any data or if it finds that a type of data is not given in its expected format, it displays an error message, along with a help proposal (whenever possible).

Checking the consistency

All the relationships that are part of the data model described in the previous chapter must be established and all the constraints must be verified. This is the heaviest task of the data analysis process, as any error or inconsistency should be detected before attempting to physically implement the relational database. The process fully analyses the significance of each type of information and its consistency and then generates the relationships this information has with the rest of the provided data. A large set of notifications, warnings and error verbose messages is associated with the process to help the user and notify him about any action taken by the data checker, as well as to help him to detect the source of any data inconsistency.

Incomplete data

Some provided data may be incomplete or may lack expected relationships with other types of data. If the missing information does not affect the consistency of the database and does not cause a critical ambiguity, then the tool accepts it, explicitly marks it as being incomplete or ambiguous, and reports it to the user so that he/she is aware of the incident. Ambiguities will always be visible to end-users who query the database, so that they are also aware of the incompleteness of the information.

In some situations, the user is required to interactively intervene during the analysis process to make a choice between several proposals. The most critical situation is when a referenced unique identifier has been demerged (split) in the referenced resource. This

[1] In order to parse mass spectrometry files, we have adapted an up-to-date version of the file parser of Phenyx, the InSilicoSpectro module (http://insilicospectro.vital-it.ch/), to work with the Make2D-DB II tool. Phenyx is the software platform for the identification of proteins and peptides from mass spectrometry data (http://www.phenyx-ms.com/).

is common with many UniProtKB accession numbers that have recently been demerged into several new entries[1].

Integration and merging of external data

Throughout the data conversion process, external data of interest is continuously integrated and merged with the user-provided data. While processing the local data, the tool collects external related information from the central data integration mediator, residing on the ExPASy server, as well as from many remote federated 2-DE systems built with the Make2D-DB II tool (Figure F.II-3). The integration is processed at different levels: the database, the map and the protein levels.

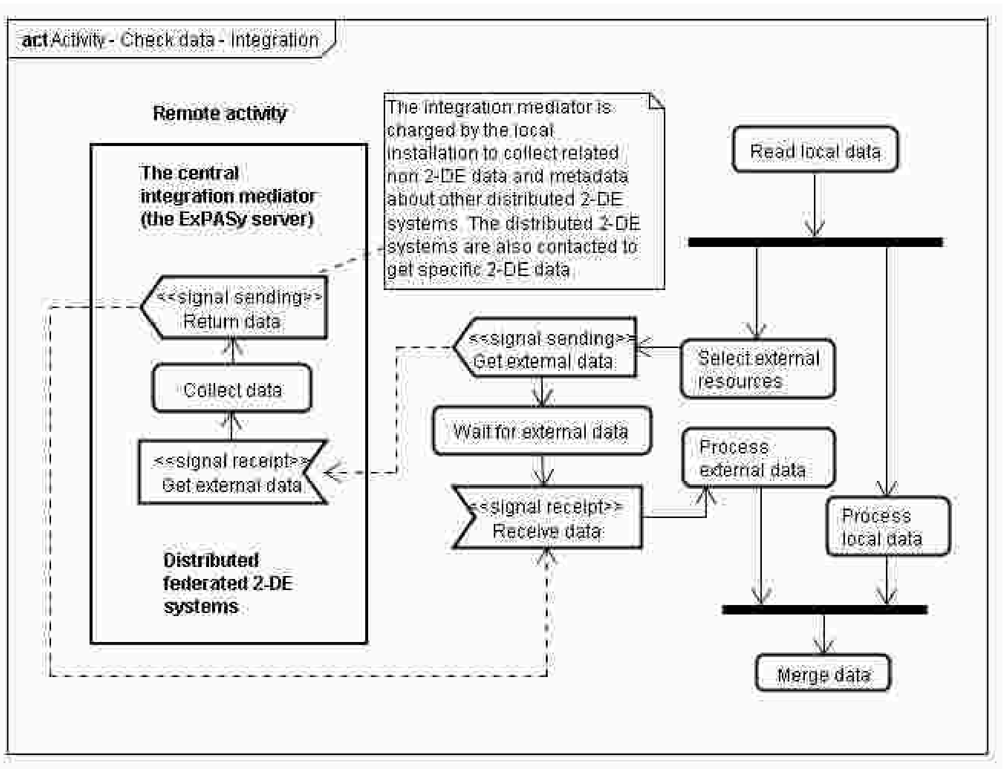

act Activity - Check data - Integration
Remote activity
The central integration mediator (the ExPASy server)
The integration mediator is charged by the local installation to collect related non 2-DE data and metadata about other distributed 2-DE systems. The distributed 2-DE systems are also contacted to get specific 2-DE data
Read local data
<<signal sending>> Return data
<<signal sending>> Get external data
Select external resources
Collect data
Wait for external data
<<signal receipt>> Get external data
Process external data
<<signal receipt>> Receive data
Process local data
Distributed federated 2-DE systems
Merge data

Figure F.II-3: Data analyser and converter – Integrating data from external resources.

Generate converted data in Make2D-DB II ASCII tables and in plain text files

The final step of the conversion process is the preparation of the data to be imported into the relational system. Technically, the tool generates ASCII (text) files reflecting the relational model. The ASCII files are intended to directly populate the relational implementation.

[1] If a UniProtKB entry has been demerged, the tool first tries to associate the local entry with the new UniProtKB entry that refers to the same organism of the local entry. If none satisfies this condition, then a user intervention is reclaimed to choose between the new entries, or to simply skip any association with UniProtKB.

Verbose reports

The user is notified about every single action taken by the analyser and the converter to inspect, rearrange, ignore or modify the data. Notification messages and error messages are interactively displayed and also reported in a final detailed log file, like for example the report at:

- http://mordor.isb-sib.ch/make2ddb/temp/last_STDOUT.log.

Many situations cause the tool to engender a notification, a warning, or an error message. Given the complexity of the analysis and the conversion tasks, it was important to report all the actions taken with as much precision as possible. We have therefore provided the tool with an extended set of verbose messages to ensure that all particular actions, all ambiguities, and all minor or major inconsistencies are accurately detected and clearly explained to the user. Table F.II-3 lists a brief sample of the kind of messages the user is expected to receive whenever an unusual situation occurs.

Table F.II-3: A sample of user-addressed messages during the conversion process.

Type	Message	Explanation

warning	*WARNING: one of your UniProtKB accession numbers (P31059 / TaxID=83333) has been demerged!
	[1] P0A7C0; P31059; P97542; [TaxID:83334] {Escherichia coli O157:H7.}
	[2] P0A7B8; P31059; P97542; Q2M8M8; [TaxID:83333] {Escherichia coli (strain K12).}
	
	assigning the following UniProtKB entry for P31059 => P0A7B8*
		The tool detects that a referred UniProtKB accession number has been demerged, and automatically assigns the one that shares the same taxonomy identifier. User intervention is required if none of the organisms of the demerged entries corresponds to the one of the local entry.

warning	*A pair of spots in entry $ac / map: $master share exactly the same pI/MW values. Please, verify the consistency of their values! Choose to continue or to stop the process. If you choose to continue, the Mw of one of the 2 spots will be shifted by +1 Dalton for internal rearrangement, but will remain as given for external display.*	Two different spots should never share the same physical properties.
warning	*.. ambiguous reference(s) in a free text topic found for map '$master' in entry $ac. The topic(s) will be marked as being ambiguous.*	Annotations in an entry that contains several bibliographic references are expected to indicate to which reference they belong. Otherwise, they are marked ambiguous.
warning	*Warning: the SRS mapper (ExPASy) returned a void response!*	A data integration operation received an unexpected void response from the data integration mediator on ExPASy.
warning	*Warning: the InSilicoSpectro MS converter cannot correctly run! No MS conversion will be processed!*	The mass spectrometry file parser could not run correctly. The tool skips the extraction of the spectra peak lists.
error	*The same RX line contains more than one reference to MEDLINE / PubMed, is duplicated or is badly written.*	A bibliographic reference is badly formatted.
error	*You refer to a spot by an abbreviation while there are more than 1 spot that could fit this abbreviation.*	Abbreviations are managed by the tool (SWISS-2DPAGE had initially some free text annotations that used to truncate the spot identifiers, *e.g.*, 11V instead of 2D-00011V). If a truncated identifier may possibly refer to several spots, an error is generated.

error	*Could not create a user to perform queries! Please, check that you have permissions to create postgreSQL users, or ask the postmaster owner to create a user called $select_user for you.*	Make2D-DB II requires specific system permissions to install its various components (system files, PostgreSQL databases, Web server directories,..). The tool is prepared to manage many conflicting situations, but may sometimes lack the appropriate permissions.
note	*AS you have selected the 'check-report' method option, the process will be resumed and will not halt -- The current entry will be escaped at this point and the checker will go through the next one. -- At the end of the operation, check the 'last_STDOUT.log' file in the main directory for a complete report of encountered errors...*	The tool encounters a fatal error, but as it runs under the <*check-report*> option, it simply skips the false entry and resumes the data analysis.

F.II.5 The relational implementation

Three possible options

The conversion process produces all the necessary ASCII tables that are imported, almost as-is, into the relational database. Three installation options launch the relational database installation: the <*transform*> option, the <*create*> option and the <*update*> option.

The <transform> option

The <*transform*> option is the one that is most often chosen by users. This option implies an initial analysis of provided data and prior conversion of this data into Make2D-DB II ASCII tables. The relational implementation builds up the relational database and imports the content of the ASCII tables into it.

The <create> option

Though this option is implied whenever the <*transform*> option is selected, it can also be selected independently. If only the <*create*> option is selected (*i.e.*, explicitly choosing the <*create*> option without the <*transform*> option), then a new void database is implemented. A void database has the structure and all the features of a standard Make2D-DB II database, except that it does not contain any data.

The <update> option

This option is needed to perform batch updates on an existing database. The batch updates include adding maps, entries and annotations, as well as modifying or erasing parts of the database’s content. The same option is also useful to upgrade an already existing database to a newer version of the Make2D-DB II tool, which implies a restructuring of the relational schema. The *<update>* option acts as a regular *<transform>* option[1], except that it starts by extracting relevant data and metadata from the existing database before completely replacing it with a new installation. The ultimate task is the integration of these extracted data and metadata into the new database. As a result, the user is presented with a new database that is an update of his/her former database[2].

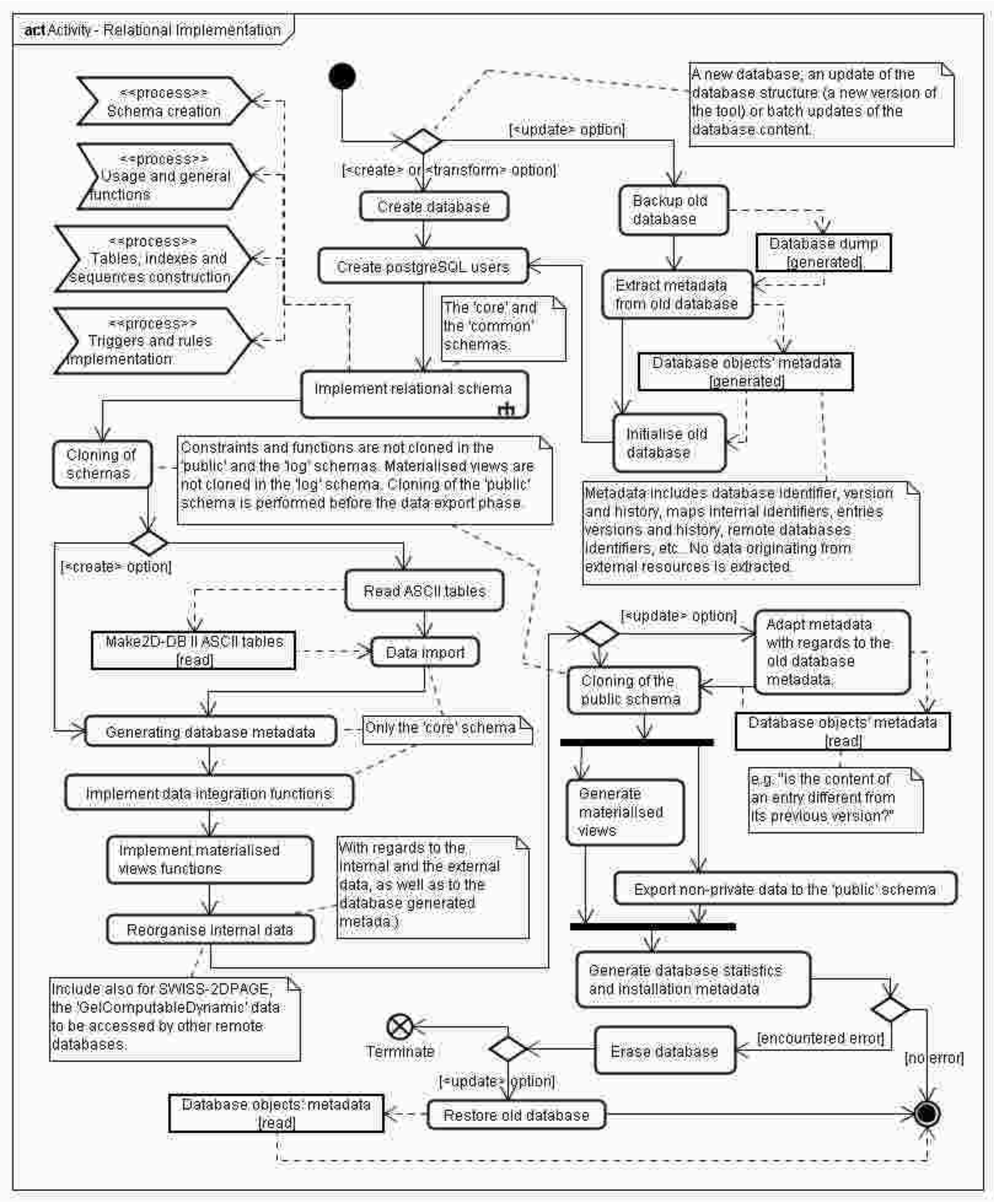
The RDBMS implementer[3]

Figure F.II-4 describes the process of the relational implementation covered by the three previous options. The process is controlled by a Perl module (the RDBMS implementer) that manages a set of different files containing SQL commands and PL/pgSQL scripts. The first step is to define whether the implementation should create a new database, or update an existing one.

1 This implies that we must provide the entire set of data and annotations, as if the *<transform>* option was the selected option..

2 Since version 2.60, users are no longer required to provide any data when upgrading to a newer version of the tool, as the tool is in charge of exporting, updating with regard to external resources, and importing back the entire database content.

3 http://mordor.isb-sib.ch/make2ddb/lib2d/make2db_POSTGRES_DB.pm


act Activity - Relational Implementation
<<process>> Schema creation
<<process>> Usage and general functions
<<process>> Tables, indexes and sequences construction
<<process>> Triggers and rules implementation
[<update> option]
[<create> or <transform> option]
Create database
Create postgreSQL users
Implement relational schema
Backup old database
Database dump [generated]
Extract metadata from old database
Database objects' metadata [generated]
Initialise old database
Cloning of schemas
[<create> option]
Read ASCII tables
Make2D-DB II ASCII tables [read]
Data import
Generating database metadata
Only the 'core' schema
Implement data integration functions
Implement materialised views functions
Reorganise internal data
[<update> option]
Adapt metadata with regards to the old database metadata
Cloning of the public schema
Database objects' metadata [read]
Generate materialised views
Export non-private data to the 'public' schema
Generate database statistics and installation metadata
[encountered error]
[no error]
Erase database
Terminate
[<update> option]
Restore old database
Database objects' metadata [read]

A new database; an update of the database structure (a new version of the tool) or batch updates of the database content.

The 'core' and the 'common' schemas.

Constraints and functions are not cloned in the 'public' and the 'log' schemas. Materialised views are not cloned in the 'log' schema. Cloning of the 'public' schema is performed before the data export phase.

Metadata includes database identifier, version and history, maps internal identifiers, entries versions and history, remote databases identifiers, etc. No data originating from external resources is extracted.

e.g. "is the content of an entry different from its previous version?"

With regards to the internal and the external data, as well as to the database generated metada.)

Include also for SWISS-2DPAGE, the 'GelComputableDynamic' data to be accessed by other remote databases.

Figure F.II-4: The relational implementation of a local database.

Updating an existing database

When a user updates an existing database, the tool starts by dumping the running database (generating a backup of its entire structure and content). It then extracts all relevant data and metadata that will ensure continuity between the new database and the former one. Such data include the database identifier, which has been once generated by the tool to unequivocally identify the database by the other remote Make2D-DB II nodes. The tool also extracts the database current version and the meta-annotations related to the last release. The same procedure applies to the map identifiers, as well as to the protein entry versions and their history[1]. Make2D-DB II also uses some sequences[2] to generate unique identifiers for experiments and identifications, which values must be transmitted to the updated database. Annotations that originate from external resources are ignored, since these annotations are updated during the installation procedure of the new database.

Once the relevant data is extracted and physically stored for subsequent integration into the new database, the entire old database is erased from the system. If the process of setting up an updated database fails, then the tool restores entirely the erased database.

Building up a new database

The tool connects to the PostgreSQL server to build up a new database. It also creates two users, with appropriate permissions, to manage the database. The first user owns the database with absolute control over it: he is the *owner* user. The second user is only allowed to query the database without the right to modify it or to access its private content: he is the *select* user.

Implementing the relational schema

The implementation of the relational schema reflects perfectly the data model that is described in Chapter E. The tool starts with defining the four distinct schemas: the *core* schema, the *common* schema, the *log* schema and the *public* schema (*cf.* paragraph E.IV).

The tool then uploads a set of general functions that are used in managing the relational installation. These functions are read from a PL/pgSQL file:

- http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_functions.pgsql

It then implements the tables belonging to the *core* and the *common* schema, as well as their related indexes. The sequences needed to generate unique identifiers are also initialised during this step. The relational implementation relies on a specific file containing all the SQL commands to achieve this task:

- http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_tables.pgsql

While it implements the different objects, the tool grants them the appropriate usage, so as to define the permissions given to the *owner* and the *select* users to access or to modify each object.

The final action performed at this stage is the implementation of the different triggers and rules associated with the tables. These triggers and rules are defined in a separate file:

[1] It is only since version 2.60 that the tool stores the entire history of protein entries.

[2] Sequences are special tables used to generate integer sequences. Typically, they are used to create a unique record ID (or key) for each row in a table.

- http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_triggers.pgsql

Cloning of schemas

As soon as the relational schema and its related objects (functions, triggers, rules, etc.) have been implemented into the core and the common schemas, the tool performs an adapted cloning of the structure of the core schema into the log schema. This operation starts with parsing the core schema, filters all defined constraints, associations and existing functions out of it, and then implements the remaining elements into the log schema. Each table implemented into the log schema is augmented by three additional attributes to contain information on data modification. The cloning of the core schema into the public schema is performed at a later stage.

Importing data (reading the ASCII tables)

The ASCII tables that have been built during the <*transform*> or the <*update*> phases contain all user data. They reflect perfectly the physical relational module and respect all the module's constraints. The tool imports progressively the content of the ASCII tables into the core schema. The RDBMS implementer includes a loader that offers the developer a simple way to quickly define which relational table corresponds to which Make2D-DB II ASCII file, as well as the order in which the tables should be populated[1].

Implementing data integration and materialised views functions

At this point, the functions responsible for the integration of data are implemented within the core schema. These functions control how data collected from external resources is merged with the local data, including the extent of data replacement. Data integration functions are active during the database installation and during any subsequent update of the database content.

The materialised views functions (as described in section E.V.10) are implemented too, and like data integration functions, have also a key role during subsequent updates of the database content.

The two following files contain all the definitions of data integration and materialised views functions, as well as the functions responsible for showing and hiding private data:

- http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_update_internal.pgsql
- http://mordor.isb-sib.ch/make2ddb/pgsql/make2db_final_views.pgsql

[1] The *LOADER_FILE* and the ORDER_TABLES procedures within the RDBMS implementer module make2db_POSTGRES_DB.pm.

Generating database metadata and reorganising internal data

At this stage, metadata are evaluated and inserted into the database (*cf.* Figure E.V-46: The Database common class.). Some specific data that has been imported into the database is also revaluated.

For SWISS-2DPAGE, we also populate the *GelComputableDynamic* table with data that will be accessed by remote databases via the Make2D-DB II Web services (*cf.* Figure E.V-38: Data Model – Computed location of a protein on remote maps.). During this stage, any other database that offers a means to compute the location of amino acid sequences on its maps, like SWISS-2DPAGE, may also insert the information needed to make this feature available (map annotations, URLs and parameters to access the computation program)[1].

Adapting metadata for updated databases

If a user is running an update process to an existing database, the tool performs a special treatment to ensure continuity between his databases. The metadata that has been extracted at a previous stage is incorporated into the newly built database. Database release and its related dates are updated. The unique identifiers previously attached to the former database and to its maps are kept, so that any established links between other remote databases and the local database are not broken. In addition, the protein entries' versions are adapted by tracking any annotation change. This is done by comparing the *annotationChanged* and the *annotationCheckSum* attributes of the *EntryVersionGeneral* and the *EntryVersion2D* tables (described in section E.V.6, under "Entry versions") between the old and the updated database. Moreover, the tool detects if an entry belonging to the old database has been demerged in two or in more new entries within the updated database (either locally, or by getting the information from UniProtKB). In case an entry demerge is detected, the demerged entries inherit their annotation versions from the former entry to which they belonged. Such a behaviour requires that merging several entries must not generate a new accession number.

Cloning of the public schema and exporting of non-private data into it

The tool parses each table's structure and its related indexes from the core (and the common) schema and rebuilds the same structure into the public schema. This operation results in a cloned schema that does not contain any of the constraints or the associations present in the core schema. All data, except non-private data, is exported into the public schema.

Generating the materialised views

All the materialised views described in section E.V.10 (protein entries, map lists, bibliographic references and identification methods) are then assembled. Technically this operation is performed while data is exported from the core schema into the public schema. The tool generates two versions of materialised views, one that includes all

[1] The administrator of such a database should modify slightly the code of the RDBMS implementer to include his/her database identifier and related data in the section following the "*populate the GelComputableDynamic table accessed by remote databases*" commentary, and contact the tool's developers to include an access to his program in the master distribution.

public and private data, reserved for privileged users having access to the core schema, and one that excludes all private data, exported into the public schema for public access.

Generating statistics and installation's metadata

The last steps in the database implementation include statistics generation covering the database content, the recording of technical information about the database installation, as well as a final analysis of the entire database for access efficiency.

Exceptions / errors

As illustrated in Figure F.II-4, if the tool encounters an error at any moment during the process of database implementation, then it erases completely the database from the system. In an update process, the former database is immediately restored back to the system.

F.II.6 Installation of the Web server components

Running the Web server process installs all the Web server components responsible for making the relational database(s) and all its related documents accessible, either locally or through the Internet. The components include:

- A main search interface to query the database(s)
- An administration interface to control and update the database(s)
- A graphical viewer / navigator for the maps
- A set of associated modules and routines
- The map images
- Documents defined by users and related to the tool, such as the private data lists
- All provided protocol documents
- All provided identification documents and mass spectrometry peak lists
- Mass spectrometry peak viewers (the light and the extended viewers)
- The Web server configuration files that control the behaviour of the Web interfaces
- Additional data documents extracted from remote UniProtKB
- Information about the species and the maps of the remote Make2D-DB II Web servers with which the Web server is linked

- Generated Apache mode rewrite rules to activate logical URLs
- A set of icons
- Readme files
- Database backups (relational dumps and extended flat files), plus several external documents

The server process is in charge of generating the needed directory structure within the Apache directory tree, and of allocating the various components based on user preferences. When a Web server is managing several local databases, each of them will have separate sub-directories to hold its related data and documents. The process also engenders the rules that make Apache understand and deal with logical URLs.

Connection between a Web server and a relational database

A Make2D-DB II Web server (or interface) is independent of any specific database, since it accesses and administers any number of ***local*** **databases**. By *local* database, we mean a relational database that is connected to a local Web server through direct **TCP/IP**[1]connection. At the same time, a local database can be concurrently accessed and managed by more than one Web server.

[1] *cf.* glossary.

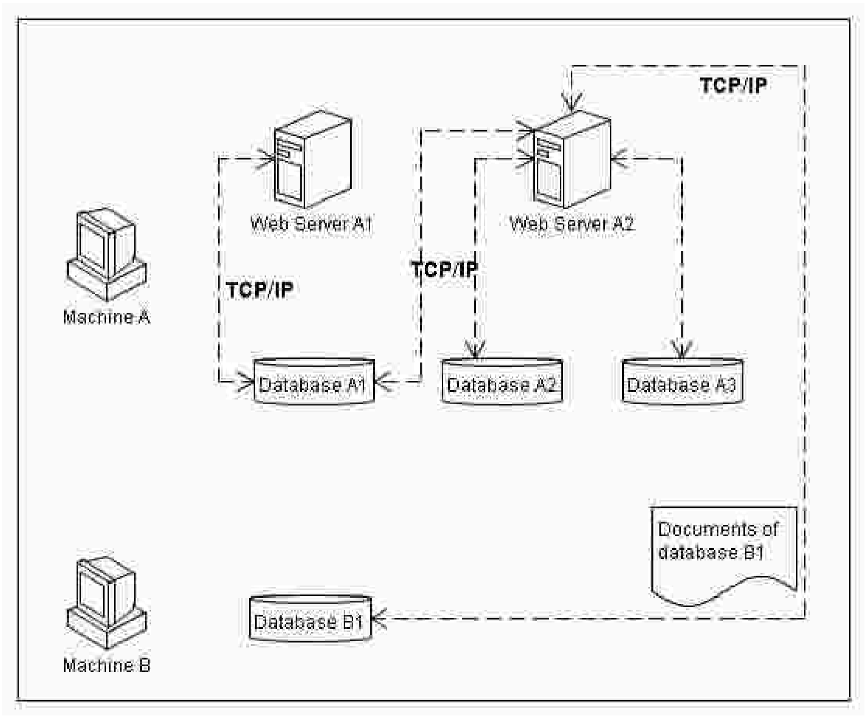


Figure F.II-5: Web servers connecting to various databases via TCP/IP connections.

Figure F.II-5 illustrates this situation. Two distinct 2D-PAGE Web servers (servers A1 and A2) are running on the same machine (Machine A), where three relational Make2D-DB II databases are installed (databases A1, A2 and A3). These databases can either be completely independent or distinct projects that are related. Web server A1 connects to the local database A1 via a TCP/IP connection. The two-headed arrow (mutual dependence) indicates that the interface can both access and manage the database. This is the classic situation, in which a Web server is linked to one single database. Web server A2 connects to all three databases through the same type of connection. Databases A1, A2 and A3 can therefore be accessed and managed by this server. A user can access and manage database A1 using either Web Server A1 or Web Server A2. Moreover, Web Server A2 has a direct TCP/IP connection with database B1, which resides on another distant computer (Machine B). Since Web server A2 is able to access and manage the distant database B1, we consider this database local to Web server A2[1].

Make2D-DB II Web servers have the ability to communicate among themselves using a federated approach. A ***remote* database** is a database that is queried by a Web server via **HTTP** / REST connection[2]. In this case, the connection is established between the querying Web server and another remote Make2D-DB II Web server that accesses, in its turn, the remote database. The remote Web server becomes

[1] Documents that are related to database B1 which are not part of the relational implementation (map images, protocols, identification documents, etc.) have to be physically present on machine A in order to be displayed by Web Server A2.

[2] *cf.* glossary.

consequently a client of the querying Web server. Users can easily configure which remote Web servers they would like to integrate within their local Web server.

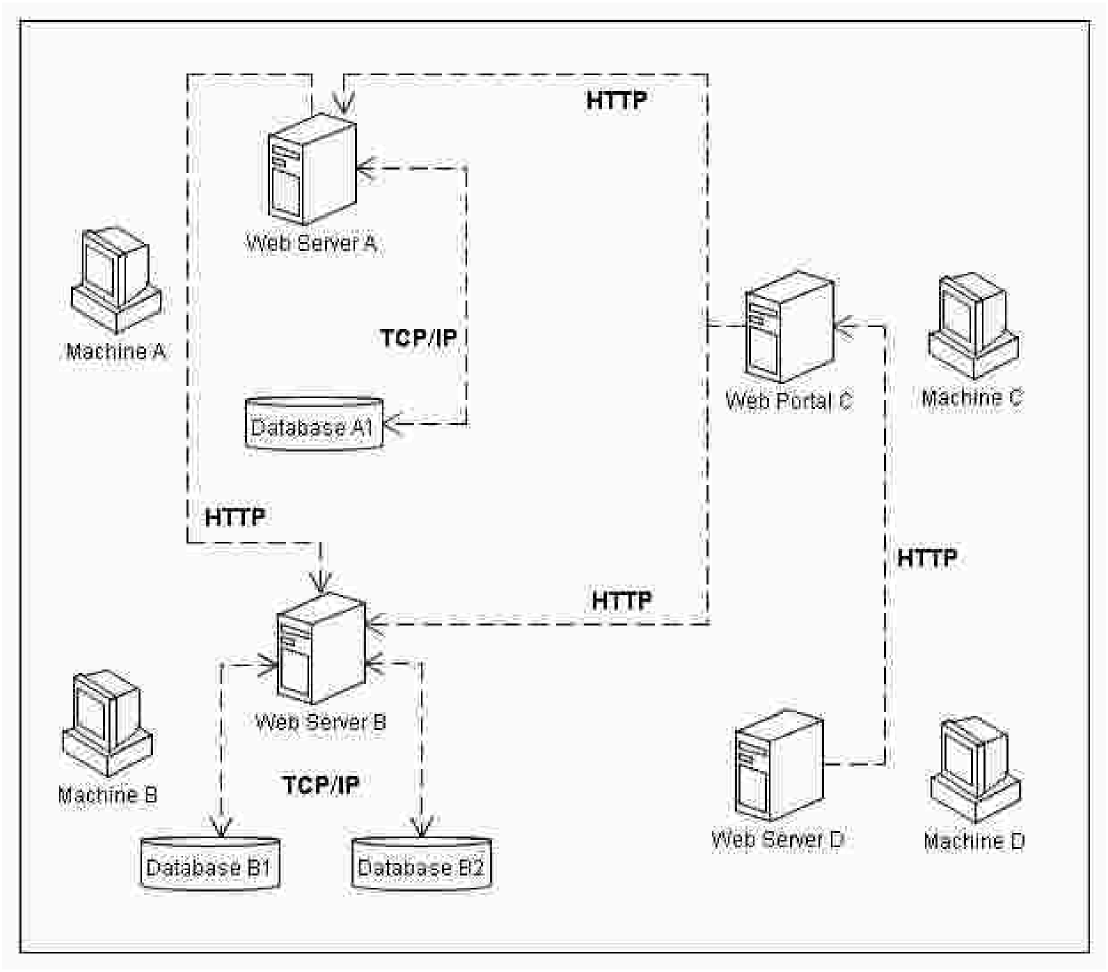


Figure F.II-6: Web servers connecting amid themselves via HTTP (REST) connections.

Let us consider Web Server A in Figure F.II-6. The server accesses a local database (A1) on Machine A via a TCP/IP connection. At the same time, a Web server (B) accesses locally two other databases (B1 and B2) on Machine B. Web server A communicates with Web server B - acting as a client - via a HTTP (REST) connection. This connection is represented by a single-headed arrow (one-way dependence). Web server A becomes therefore able to extract resources from Web server B, thus indirectly interrogating databases B1 and B2. For obvious security reasons, we have limited the interaction between Web servers to the extraction of resources. Web server A can neither alter nor manage the remote databases B1 and B2.

When a Make2D-DB II Web server is remotely contacted by another Web server, the contacted server gives only access to its own local databases. By default, a contacted Web server does not perform any query over its remote databases in response to another Web server's request. This is illustrated in Figure F.II-6. When a third Web server (Web Portal C) contacts Web server A, the latter gives only access to its local database A1, without enabling the HTTP connection that links it with Web server B. However, for contacted Web servers, this default behaviour can be reversed by defining a Web portal.

Web portals[1]

A Web server gives only access to local databases when it is remotely (non interactively) contacted. Nevertheless, A Make2D-DB II user can configure his/her server to give access to all its local and remote databases when it is contacted by another Web server. To distinguish between the two kinds of behaviours, we choose by convention to use the term *Web portal* for servers with the extended behaviour. In Figure F.II-6 we have defined the Web server on Machine C as a Web Portal. Hence, when a Web server (Web server D) queries Web portal C, it also indirectly contacts Web server A and Web server B. All remote databases (A1, B1 and B2) are therefore accessed.

Preventing cyclic loops

Given the fact that Web portals have the ability to bi-directionally contact each other, and that the contact schema is not predetermined, it is important to deal with potential situations in which an infinite loop may result. Figure F.II-7 shows a fictive schema involving several Web portals: Web portal A is configured to query Web portal B, which in its turn is configured to query Web portal C. The latter queries Web portal D, which has been configured to query Web portal C, as well as the initial querying Web portal A!

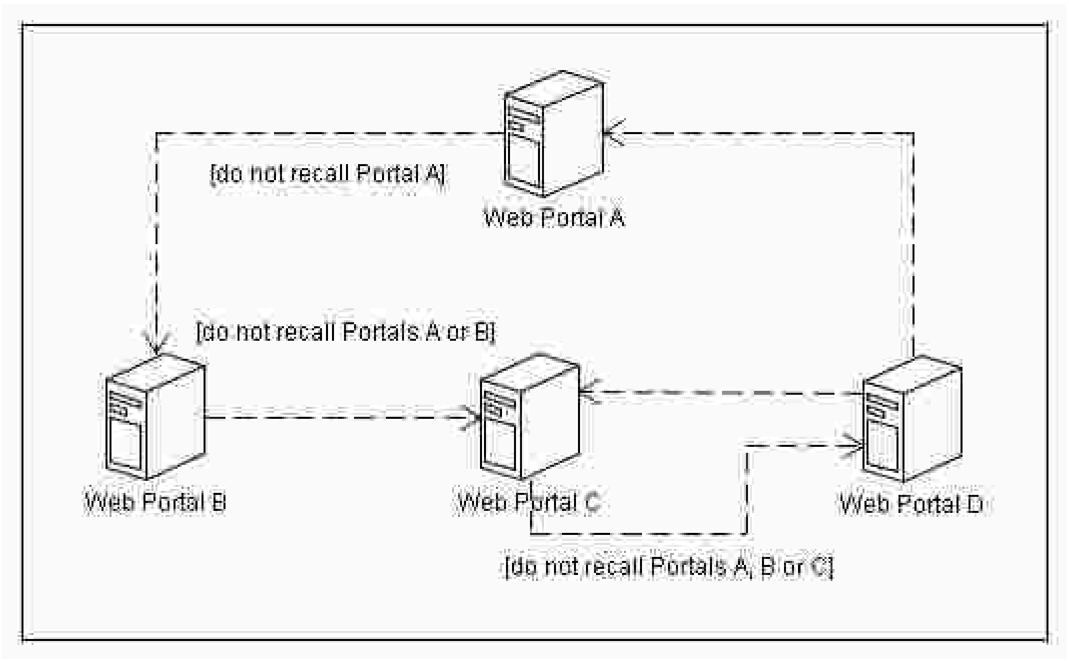


Figure F.II-7: Preventing cyclic loops in extended networks.

[1] *cf.* glossary.

We can represent this network using a directed graph, in which Web portals are the nodes and messages are the edges. To prevent the cyclic situation that Web portal D would cause by contacting back Web portals A and C, messages that are transmitted from a Web portal to another include a list of all previously visited nodes, including the transmitting node. The entire path that has been followed is therefore known by each intermediate or final node. No query is sent again to a node of this path[1].

It is important to note that although this solution eliminates cyclic loops, it does not prevent a node from being part of different paths. In Figure F.II-7, if Web Portal B had a direct connection to Web portal D, the latter would be visited twice, one time by Web portal B, and another time by Web portal C. This might result in Web portal A - the root - receiving twice the same data from Web portal D.

Configuration for a Web server process

As already introduced in section F.II.3, in order to run a Web server users must provide the tool with a suitable server configuration file: the *2d_include.pl* file. Depending on the purpose, they may need either to generate a new configuration file to set up a new server, or to adapt an existing one in order to modify an already running server. In both cases, running the tool in the interactive *<config>* mode simplifies this task. Alternatively, when dealing with an already running server, users may simply edit its associated configuration file using any text editor.

In the server configuration file *2d_include.pl*, each local database must have a separate block that defines all its related parameters[2]. When a user installs a unique database – like in the majority of cases - only one block is needed. If he/she decides afterwards to include other local databases (or projects) within the same Web server, then he/she will need to add a new block for each additional database.

At any moment, a user can also give access to one or several remote databases through his/her Web server. Contacting a remote 2D-PAGE database is physically performed only when end-users[3] explicitly select the Web server associated with the remote database from the list of enabled resources.

Including remote databases within the list of authorised resources is simply achieved by manually adding to the basic configuration file *basic_include.pl* (introduced in section F.II.3) blocks of the format:

```
'Remote Web server name' => {
  URL => 'http://remote-domain/2d-server/',
  URI => 'http://remote-domain/2d-server/',
  database => ['remote database 1', .., 'remote database n'],
```

[1] Web portals are identified by an official name (given by their maintainers). If the need arises, *e.g.*, if two different Web portals share the same "official" name, we may extend this definition to also include Web locations (URLs) to distinguish between them.

[2] Blocks are defined in the database section entitled "*DATABASES SPECIFIC PARAMETERS*" in *2d_include.pl*.

[3] We remind that a user refers to the person who uses the tool to build a database and publish the data, and that an end-user is a person who accesses this data via the Web server.

```
    activated => [1|0]
  },
```

A remote Web server is defined with its common name and its Web location (URL). URI is optional and serves only if the remote Web server does not support the Apache mode rewrite mode[1]. The selected remote databases managed by the remote Web server are either given as a list (separated by commas), or simply by using the keyword '*All*' to include all available remote databases. An activation parameter can be set to 0 or to 1 to temporarily (and rapidly) deactivate and activate a remote resource. Remote Web portals are also included using an identical syntax.

Users also define in *basic_include.pl* whether their Web server should act as a Web portal client, thus contacting all its defined remote resources whenever a remote Web server asks for it (*cf.* F.II.6: Web portals), or limit access to its local databases in response to a remote query[2].

The reason for choosing to define this information in *basic_include.pl*, rather than in the personal configuration file *2d_include.pl*, is that *basic_include.pl* is not frequently modified by average users. This is to prevent an unnecessary over exploitation of remote accesses, since an abuse on the number of simultaneously accessed resources tends to significantly slow down a Web server performance. Since the latest versions of the tool, SWISS-2DPAGE and the World-2DPAGE Portal - and more recently World-2DPAGE Repository - are activated by default.

Running the Make2D-DB II server process

The Web server process is executed by running the tool with the <*server*>, the <*update*> or the <*transform*> options; the two last options acting as a shortcut to execute successively the <*check*>, the <*create*> and the <*server*> options while updating or installing a new local database.

In the majority of cases, users who are using the tool install at the beginning a single database and assign a new Web server dedicated to query the database. Therefore, they create a new server configuration file containing one database block. After the database implementation is over, the tool proceeds to implement the Web server components. If the user has full write permissions into the Apache directories, then the server installation will be carried out without interruption. Otherwise, the installation is not performed and the tool invites the user to enable these permissions, or to login using a user ID that gives write permissions into the Apache directories[3]. To resume the interrupted operation, the user should execute once more the tool, this time using the <*server*> option.

[1] Since for remote Web servers not supporting the Apache mode rewrite rules, the URL must point to the main interface script, *i.e.*, URL => 'http://remote-domain/cgi-bin/2d-server/2d.cgi'.

[2] The $SERVER::onlyInterfaces and the $SERVER::PortalWebService parameters in basic_include.pl.

[3] Typical UNIX users having full write permissions into the Apache directories may be 'http', 'apache' or 'nobody', as well as 'root'.

If a user is updating the content of his/her database or is upgrading to a new version of the tool using the <*update*> option, then he/she does not need to provide any new server configuration file. When the tool does not find a new server configuration file, it looks for the one that is associated with the already running Web server[1]. Nonetheless, the tool recommends the generation of a new configuration file for upgrades between major releases of the tool (from version 0.x to version, 1.x, or from version 1.x to version 2.x).

Adding a new database that has been built using the <*transform*> option to an already running server does not explicitly require generating a new server configuration file. By selecting "Changing a Running Server Parameters" in the <*config*> mode, the previous configuration of the running Web server may simply be interactively modified to include the database as an additional new local database.

The <*server*> option can also be executed independently in order to set up a Web server or a Web portal with no local databases, and that gives a simultaneous access to other remote Web servers. No main configuration file for local databases – as defined in F.II.3 - is therefore required.

Web server components, features and interactions are described in more details in Chapter G.

F.III. The central data integration mediator

Reinforcing the federated approach using a central mediator

Make2D-DB II Web servers communicate autonomously between themselves using a federated approach, in which they exchange data and mutually understand each other. However, data originating from other resources need to be transformed first into a structure that is compliant with the tool. In data integration, the component that is responsible for converting data from a source data format into a destination data format is called a *mediator* (*cf.* Chapter D.). The mediator maps data elements from the source to the destination and performs all required transformations. In bioinformatics, resources are subject to many changes regarding their format, structure and accessibility. The central mediator guarantees continuity of data integration, even when a resource modifies its data exchange format, or when a resource changes its physical location.

Connections between remote Make2D-DB II implementations and the residing ExPASy mediator are activated from the start of the database creation and throughout the database life, assuming that the database maintainers perform regular updates of their database. The mediator understands query messages sent by remote Make2D-DB II servers and knows how to accurately extract the required data from a range of distributed resources. Moreover, it knows how to deal with all versions of the tool that may differ by the messages they are sending to the mediator and the data they expect to

[1] The user can also direct the tool to use an alternative configuration file by giving its path.

receive. Another advantage is the possibility to extend or replace initial resources by analogous ones when appropriate, without the need to update the remote installations' behaviour. In addition, the mediator keeps the remote installations aware of the existence of each other, thus letting them establish direct links between themselves.

Two main components: the file extractor and the data mediator

The central data integration mediator (Figure F.III-1) is composed of two separate components, the *file extractor* and the data *mediator*. We will simply refer to the latter as the mediator. Conceptually, there is no fundamental difference between the two components. They play the same role in interpreting received queries, gathering the corresponding data and sending it back to the caller. The main difference resides in the level of transformation that they apply on the data before sending it back. While the file extractor is mainly concerned by text files, transferring them with minimal restructuring of their content, the mediator deals with much more complex data representations, originating from various sources. Both components are not restricted to work with Make2D-DB II. They can be easily extended in order to perform general data integration mediation and file extraction.

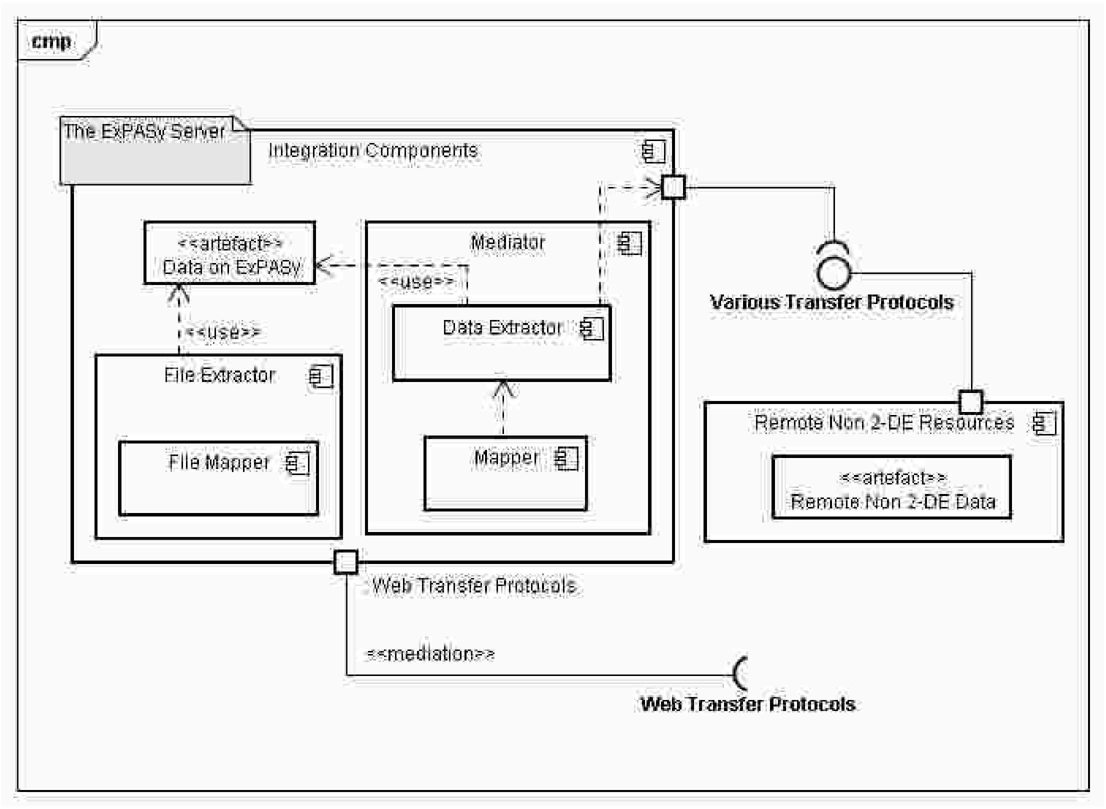


Figure F.III-1: The central data integration mediator and file extractor on ExPASy.

The file extractor[1]

The file extractor is a simple Web script that is triggered by direct HTTP GET queries. It receives a simple argument asking to receive a text file that is present on the ExPASy server. It also can be asked to send a specific portion of a file's content that contain a particular piece of information. The script provides Make2D-DB II remote databases with the most up-to-date versions of a collection of files, in particular the database cross-reference list and the tissue list, both maintained by the Swiss-Prot group. The script is also able to extract some metadata required by the remote installation, which include the tissue list version, the current database release numbers of UniProtKB/Swiss-Prot and UniProtKB/TrEMBL, along with their release dates. An argument non-recognised by the script is assumed a name of a text file that the caller wants to extract from the ExPASy server. The only condition is that the file is present in a directory containing exclusively public data.

Some examples of files and information extraction are given below:

- http://www.expasy.org/cgi-bin/txt_on_web.cgi?DbCrossRefs.txt
- http://www.expasy.org/cgi-bin/txt_on_web.cgi?tisslist.txt
- http://www.expasy.org/cgi-bin/txt_on_web.cgi?UniProt:version
- http://www.expasy.org/cgi-bin/txt_on_web.cgi?Swiss-Prot:date

Make2D-DB II developers must check regularly for any change on ExPASy that would affect file extraction. On several occasions, the file extractor had to be slightly adjusted in order to conform to changes related to file locations. When the raw list of tissues also recently changed the format of its content, the script was adapted to provide older versions of the tool with the same file format as callers expect to receive.

The mediator[2]

Initially, the mediator has been developed to work with SRS - the Sequence Retrieval System (Etzold, Argos 1993) - to extract data from local or remote SRS servers. Consequently, its data exchange protocol reflects a SRS style. The mediator is contacted by simple HTTP POST queries and returns semi-structured text responses. The process makes use of a generic query language. The caller defines a specific database or resource to retrieve data from (*e.g.*, UniProtKB, NEWT, etc.), a method of retrieval from the resource (SRS/getz or SRS/wgetz, SQL, etc.), the list and the type of identifiers to look for and the type of output fields to receive. For example, a caller may ask to receive all cross-references and feature keys of a list of UniProtKB entries by sending to the mediator a list of accession numbers and by selecting UniProtKB and the SRS retrieval method.

[1] http://mordor.isb-sib.ch/make2ddb/expasy/txt_on_web.cgi

[2] http://www.expasy.org/cgi-bin/getz_syntax_mapper.cgi, only POST queries are allowed.

The mediator has the ability to communicate with tools other than Make2D-DB II and therefore expects from the caller to declare its identity - which can be for example a Make2D-DB II server - and to optionally give its version. This helps the mediator to adapt its exchange protocols, and the mapping applied on data, depending on which tool and tool's version is the caller. This also helps the mediator to limit or to assign a list of pre-defined resources to a particular tool. In theory, the list of resources can be unlimited. Nevertheless, for the time being, the choice of accessible resources is limited to those needed by Make2D-DB II. The following resources are currently covered: UniProtKB/Swiss-Prot, UniProtKB/TrEMBL, NEWT and SWISS-2DPAGE, as well as a set of remote 2D-PAGE resources.

The mediator includes a list of registered Make2D-DB II servers and their remote databases. This list is maintained by the Proteome Informatics Group. Any Make2D-DB II Web server is able to retrieve this list in order to automatically establish up-to-date cross-links between its own databases and the remote databases of registered servers.

Adapt to changes concerning data resources

The most vital role of the mediator is to ensure that any change related to the location or to the format of a resource does not affect Make2D-DB II distributed systems. For example, UniProtKB should migrate in 2008 from the ExPASy server to a new dedicated server[1] and should benefit from a renewed interface to access data. This new interface allows direct and refined retrieval of data by using simple HTTP GET queries. Consequently, the mediator will have to adapt its behaviour, replacing all its SRS extraction procedures by GET queries (the UniProtKB SRS server will be no longer maintained). Such changes can be operated without disturbing the good functioning of remote Make2D-DB II servers.

The following address gives access to the mediator implementation:

- o http://mordor.isb-sib.ch/make2ddb/expasy/getz_syntax_mapper.cgi

F.IV. The package content

The distributed package elements are described at the following address:

- ❖ http://world-2dpage.expasy.org/make2ddb/1.Readme_main.html#package

To make easier the parsing of the documents, manuals and scripts for readers of this manuscript, we made them available at the following address:

- ❖ http://mordor.isb-sib.ch/make2ddb/

[1] http://www.uniprot.org

We have also provided a FAQ page for users to post their questions and observations and to report unexpected bugs:

- ❖ http://world-2dpage.expasy.org/make2ddb/faq.html

Make2D-DB II is an open source project. It follows the Artistic Licence directives[1], except that claiming fees out of it is prohibited. The tool's license[2] insists on the fact that modified versions should clearly be labelled as modified. Since each individual database and its associated Web server can be part of the global virtual 2D-PAGE environment, it is imperative to know whether they are perfectly compliant with the other "official" installations or whether data exchange should be treated with caution.

The tool is developed and managed under Concurrent Versions System (CVS)[3] to keep track of changes and consecutive distributed versions of the tool.

Make2D-DB II license:

```
ORIGINAL LICENSE:

The Make2D-DB II tool is
(c) Copyright 2002 Swiss Institute of Bioinformatics
     All Rights Reserved.
     Author: Khaled Mostaguir (khaled.mostaguir@isb-sib.ch)

============================================================

PERMISSION TO USE, COPY AND DISTRIBUTE THIS TOOL, INCLUDING ALL OR ANY OF
ITS COMPONENTS, IS HEREBY GRANTED FOR ANY PURPOSE EXCEPT FOR CLAIMING FEES
OUT OF THE TOOL (AS A WHOLE OR AS COMPONENTS).

THIS ENTIRE NOTICE SHOULD BE INCLUDED IN ALL DISTRIBUTED COPIES AND IN ALL
MODIFIED VERSIONS OF THIS TOOL, AS WELL AS IN ALL TOOLS THAT USE SOME OF ITS
COMPONENTS.

MODIFIED VERSIONS SHOULD CLEARLY STATE BEING MODIFIED IN AN ADDITIONAL NOTE.

THIS TOOL IS PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN
PARTICULAR, THE AUTHOR DOES NOT MAKE ANY REPRESENTATION OR WARRANTY OF ANY
KIND CONCERNING THE MERCHANTABILITY OF THIS TOOL OR ITS FITNESS FOR ANY
PARTICULAR PURPOSE.
```

[1] http://www.perlfoundation.org/artistic_license_2_0

[2] http://www.expasy.org/ch2d/make2ddb/License.txt

[3] http://www.nongnu.org/cvs/, *cf.* glossary.

Chapter G

CHAPTER G. MAKE2D-DB II WEB SERVERS

A data management and integration system relies heavily on suitably designed interfaces in order to be efficient and usable.

Interactively, a Web site giving access to database content must be easy to use and should provide information that is not hard to read. At the same time, data integration operations should be technically simple and intuitive, to guarantee a better interoperability with disparate systems.

G.I. Introduction

A Make2D-DB II Web server consists of a set of user interfaces and software interfaces running on a Web server and accessible over the Internet. The user interfaces are interactive interfaces that enable people to interact with the system, while the software interfaces are in charge of the interoperability of the system with other remote systems.

Here are four duties handled by the interfaces:

- Give interactive access and display the content of local and remote databases
- Extract or refer to objects within local databases
- Administer the Web server and update local databases
- Ensure bi-directional data exchange and data integration with other remote resources

An illustrative Web server, Test-2DPAGE, has been set up for Make2D-DB II users to demonstrate the tool. It has the reference name *2d_test* and it accesses three independent databases / projects, respectively named: Test-2DPAGE I, Test-2DPAGE II and Test-2DPAGE III. All data in Test-2DPAGE is purely fictive and serves only to illustrate the tool. This Web server is accessible at the following address:

- http://mordor.isb-sib.ch/2d_test/ (*cf.* note below)

Two additional Web servers, the SWISS-2DPAGE database and the World-2DPAGE Portal will also help us to illustrate some examples throughout this chapter. These two resources are located at the following address:

- http://www.expasy.org/swiss-2dpage/ (SWISS-2DPAGE search engine)
- http://world-2dpage.expasy.org/portal/ (World-2DPAGE Portal)

- *Note:* In March 2008, the Test-2DPAGE Web server will move to **http://world-2dpage.expasy.org/test-2dpage/**. Consequently, this new domain name should also replace the one used in any subsequent URLs referring to the test database.

G.II. Overview of the Web components and their interactions

Figure G.II-1 gives a schematic overview of the Web server components and their local and remote interactions. The main entry point to a Web server is the main interface, from which one can interrogate local and remote databases, trigger data viewers, activate the privileged users' mode for full private data access, or switch to the administration interface[1].

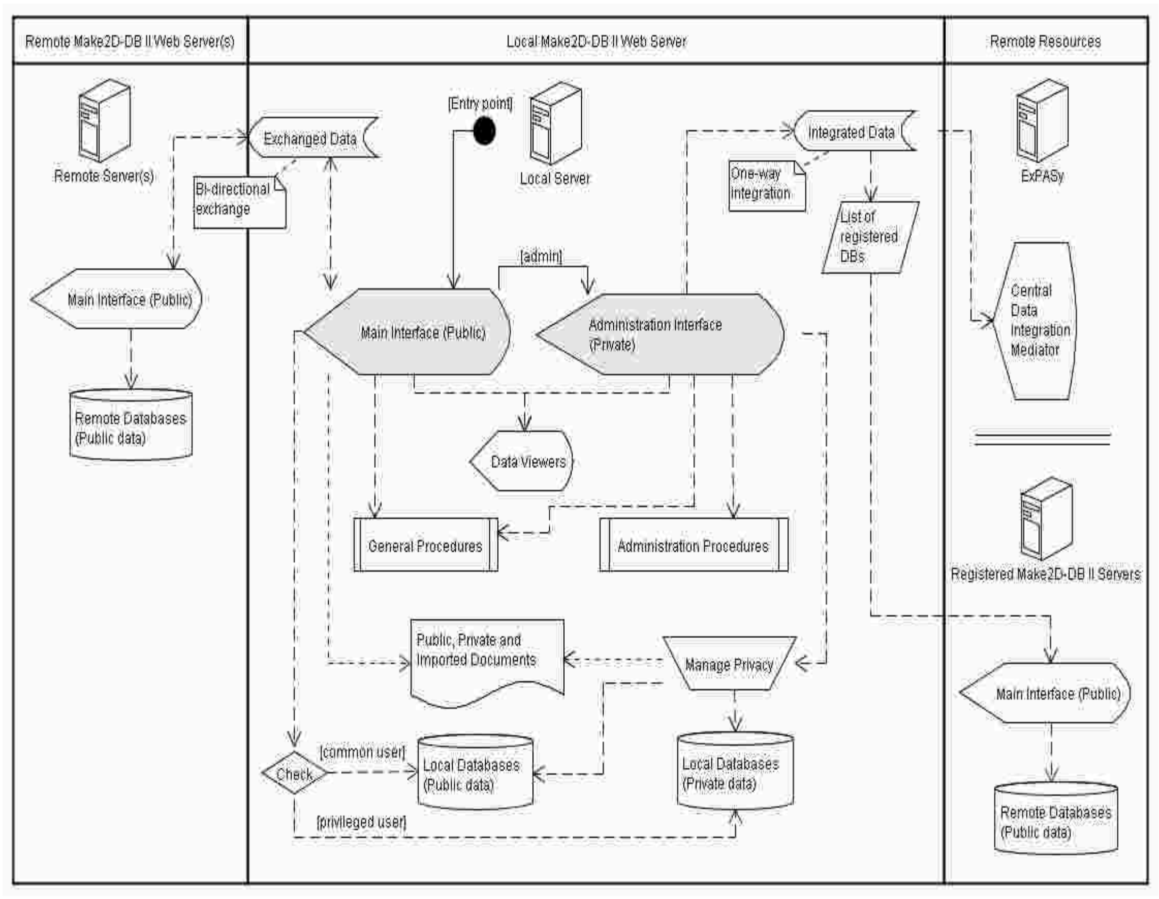

Remote Make2D-DB II Web Server(s)
Local Make2D-DB II Web Server
Remote Resources
Remote Server(s)
Exchanged Data
Bi-directional exchange
[Entry point]
Local Server
Integrated Data
One-way integration
List of registered DBs
ExPASy
Central Data Integration Mediator
Main Interface (Public)
Remote Databases (Public data)
[admin]
Main Interface (Public)
Administration Interface (Private)
Data Viewers
General Procedures
Administration Procedures
Registered Make2D-DB II Servers
Public, Private and Imported Documents
Manage Privacy
Main Interface (Public)
[common user]
Check
Local Databases (Public data)
Local Databases (Private data)
[privileged user]
Remote Databases (Public data)

Figure G.II-1: Schematic overview of Web servers' components and interactions.

The main interface

The main interface gives direct access to all local databases that are associated with the Web server. Only public data can be accessed, except for privileged users who must provide an appropriate password in order to access private (hidden) data[2]. Access to the main interface can be performed interactively or non-interactively, in order to display, extract or integrate local data.

[1] Components of the Web server are accessible at http://mordor.isb-sib.ch/make2ddb/http_server/

[2] Privileged users' passwords are encrypted and saved in a cookie that expires after 12 hours, or when users log out.

Access to remote databases is also performed from the main interface. Data exchange is then carried out between the local main interface and the distant Web servers in control of the remote databases. The exchange is bi-directional since remote Web servers that want to interrogate local databases can send their queries in the same manner to the main interface of the local Web server.

A Web portal is limited to a main interface and its associated procedures, with no direct access to any local data, or to any administration interface. Queries can therefore be only performed by exchanging data with remote Make2D-DB II Web servers.

The administration interface

The administration interface is a clone of the main interface that is supplemented with procedures for data administration. These procedures are in charge of managing all local databases associated with the Web server, as well as handling databases' metadata and performing database export operations.

Mainly, the administration procedures are responsible for updating the database content with regard to data integrated from various external resources, and by means of the ExPASy-resident central data integration mediator (F.III). Moreover, the list of currently available remote Make2D-DB II databases - provided by the central mediator - allows the administration interface to contact these remote databases and to generate up-to-date cross-references with them. Cross-references are based on common entities, such as common protein entries, maps of same species and/or tissue, etc. All data integration processes are unidirectional: the remote installations act only as data providers without being able to interrogate an administration interface.

The administration interface offers the same search features as those presented by the main interface, but the search operations are expanded to include all private data and are limited to local databases. Only interactive access is allowed. Access to the interface is reserved for administrators thanks to a highly secured and password protected mechanism.

Data viewers

Data viewers are components that display data graphically in an interactive manner. Several data viewers are associated with the main and the administration interfaces. The viewers consist of a map navigator, a mass spectrometry peak list displayer, and a mass spectrometry browser. The map navigator and the mass spectrometry browser can be launched as standalone viewers, independently from the other interfaces.

Accessing the Web server's main entry

When users set up a Web server using the <*server*> option (F.II.6), they define a reference name to be associated with their 2D-PAGE server. This name is not necessarily the name of any of the local Make2dB-DB II databases managed by the server[1], but rather a reference that, when added to the domain name[1] of the running

[1] Usually, users that install a Web server to manage a single local database/project fairly tend to assign the same name to both their Web server and their database.

HTTP Apache server, forms the URL address to access the Make2D-DB II server, such as in:

- *http://domain/2d_server/*

  Example:

  - Test 2D-PAGE: http://mordor.isb-sib.ch/2d_test/
  - SWISS-2DPAGE: http://www.expasy.org/swiss-2dpage/

  This URL redirects to the script of the main interface, which in most standard installations (depending on users' configuration parameters) is physically located at:

- *http://domain/cgi-bin/2d_server/2d.cgi*

  Example:

  - Test 2D-PAGE: http://mordor.isb-sib.ch/cgi-bin/2d_test/2d.cgi[2]
  - SWISS-2DPAGE: http://www.expasy.org/cgi-bin/swiss-2dpage/2d.cgi

The domain name is either a public Web hostname, or just a restricted domain accessible only by one or several local machines.

G.III. The main interface

G.III.1 The interactive mode

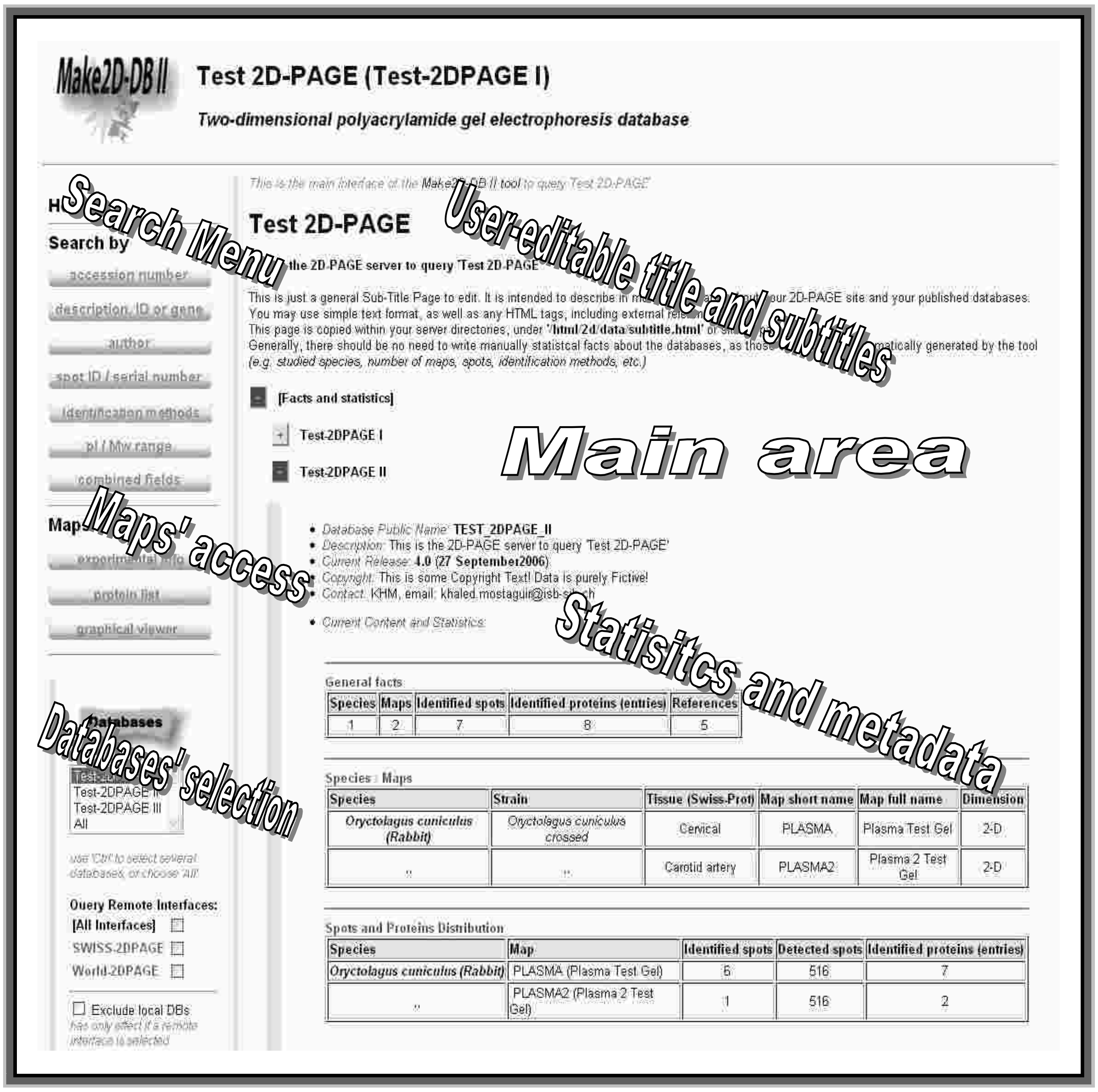
In the interactive mode, the main interface presents facts and metadata about the Web server and offers end-users numerous search and parsing features that satisfy their most frequent queries (Figure G.III-1). The interface is simple and intuitive. Administrators have a large control over the look and feel of the interface through the server configuration file (F.II.3), and, for further refinement, through the modification of its associated Cascading Style Sheets (CSS)[3]. All displayed or generated Web pages are fully HTML 4.01 compliant[4].

[1] *cf.* http://en.wikipedia.org/wiki/Domain_name

[2] To be replaced by "http://world-2dpage.expasy.org/test-2dpage/cgi-bin/2d.cgi" starting from February 2008.

[3] The *shared_definitions* subroutine in *2d_util*.pl.

[4] Following the W3C recommendations at http://www.w3.org/TR/REC-html40/



Test 2D-PAGE (Test-2DPAGE I)

Two-dimensional polyacrylamide gel electrophoresis database

This is the main interface of the Make2D-DB II tool *to query Test 2D-PAGE*

Test 2D-PAGE

the 2D-PAGE server to query 'Test 2D-PAGE'

This is just a general Sub-Title Page to edit. It is intended to describe in m[illegible] your 2D-PAGE site and your published databases.
You may use simple text format, as well as any HTML tags, including external refe[illegible]
This page is copied within your server directories, under **'/html/2d/data/subtitle.html'** or [illegible]
Generally, there should be no need to write manually statistcal facts about the databases, as those [illegible]matically generated by the tool
(e.g. studied species, number of maps, spots, identification methods, etc.)

[Facts and statistics]

Test-2DPAGE I

Test-2DPAGE II

- *Database Public Name:* **TEST_2DPAGE_II**
- *Description:* This is the 2D-PAGE server to query 'Test 2D-PAGE'
- *Current Release:* **4.0 (27 September2006)**
- *Copyright:* This is some Copyright Text! Data is purely Fictive!
- *Contact:* KHM, email: khaled.mostaguir@isb-s[illegible]ch

- *Current Content and Statistics:*

General facts

Species	Maps	Identified spots	Identified proteins (entries)	References
1	2	7	8	5

Species : Maps

Species	Strain	Tissue (Swiss-Prot)	Map short name	Map full name	Dimension
***Oryctolagus cuniculus* (Rabbit)**	*Oryctolagus cuniculus crossed*	Cervical	PLASMA	Plasma Test Gel	2-D
"	"	Carotid artery	PLASMA2	Plasma 2 Test Gel	2-D

Spots and Proteins Distribution

Species	Map	Identified spots	Detected spots	Identified proteins (entries)
***Oryctolagus cuniculus* (Rabbit)**	PLASMA (Plasma Test Gel)	6	516	7
"	PLASMA2 (Plasma 2 Test Gel)	1	516	2

use 'Ctrl' to select several databases, or choose 'All'

has only effect if a remote interface is selected

Figure G.III-1: Home page of the main interface (Test-2DPAGE).

The main area is the place where information is presented, queries are executed, and results viewed. The home page displays user-defined headings and automatically generated statistics covering the content of each local database, in addition to contact information and related resources. Access to the administration interface or logging in for privileged users[1] is also performed from here. For query results, clicking on the name or the image of any listed object within the main area fetches the object's default representation. For a map, experimental information and the list of identified spots are

[1] In order to access hidden data on the Test-2DPAGE Web server, readers of this manuscript should login as privileged users using the password '*private*'.

displayed. For a map's thumbnail, graphical navigation is initiated. For a protein, the full-annotated entry is displayed. For a spot, its map's location is graphically shown with experimental annotation at display. Clicking on an identification method of a spot displays all related experimental data and analysis documents.

Search menu

A list of search options is available through the search menu area. End-users can perform specific queries against any local or remote database by selecting one of the pre-defined search options.

Search by accession number

The accession number search option retrieves protein entries that correspond to a given primary or secondary accession number (*AC*) or a protein identifier (*ID*). A list of all proteins from an entire database or from some particular maps can be displayed beforehand. Entries' output can be requested in different formats.

The most comprehensive view is the '*Nice View*' output format. It displays protein annotations and its history, bibliographic references, corresponding maps, spot physical properties and identification evidence in an exhaustive HTML display enriched with clickable objects and many local and external links. An automatically incorporated list of cross-references to other remote Make2D-DB II databases is also in display. If the protein is cross-referenced to an entry from the protein main index (UniProtKB), theoretical computation of the expected pI/Mw values of the protein and its known fragments is proposed[1]. A list of remote maps over which the protein location can be estimated is also displayed (for the time being, only SWISS-2DPAGE maps are accessible). An ending section gives additional related annotations extracted from UniProtKB. In particular, annotated keywords[2] and a comprehensive list of cross-references to many other remote resources are available from this section. Figure G.III-2 shows parts of the sections of this comprehensive view[3].

Search by description, entry name, gene name or UniProtKB keyword

End-users can also retrieve protein entries that contain a specific keyword within their description, their entry name, their gene name (including aliases, ORF and OLN), or their associated keywords. The search can be extended to include external UniProtKB annotations with regard to the local proteins.

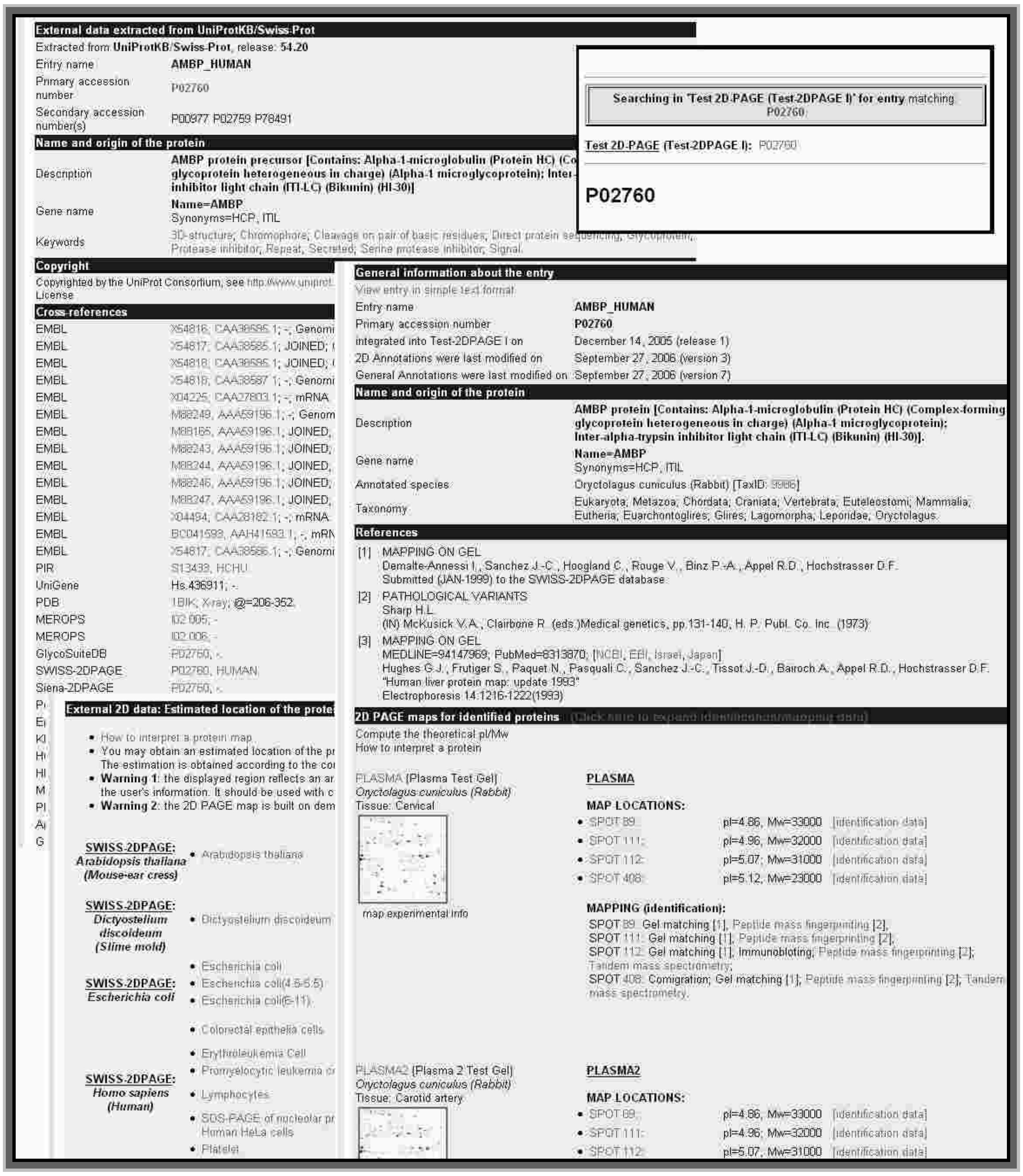
Search by author

The search by author option retrieves all authors who worked on one or several entries. Inversely, a list of all entries corresponding to a particular author may also be retrieved.

[1] By directing to the "Compute pI/Mw tool " on the ExPASy server: http://www.expasy.org/tools/pi_tool.html

[2] http://www.expasy.org/sprot/userman.html#KW_line

[3] Example given for <P02760> at http://mordor.isb-sib.ch/2d_test/ac=P02760&database=test-2dpage_I



External data extracted from UniProtKB/Swiss-Prot

Extracted from **UniProtKB/Swiss-Prot**, release: **54.20**

Entry name	**AMBP_HUMAN**
Primary accession number	P02760
Secondary accession number(s)	P00977 P02759 P78491

Name and origin of the protein

Description	**AMBP protein precursor [Contains: Alpha-1-microglobulin (Protein HC) (Co glycoprotein heterogeneous in charge) (Alpha-1 microglycoprotein); Inter- inhibitor light chain (ITI-LC) (Bikunin) (HI-30)]**
Gene name	**Name=AMBP** Synonyms=HCP, ITIL
Keywords	3D-structure; Chromophore; Cleavage on pair of basic residues; Direct protein sequencing; Glycoprotein; Protease inhibitor; Repeat; Secreted; Serine protease inhibitor; Signal.

Copyright

Copyrighted by the UniProt Consortium, see http://www.uniprot. License

Cross-references

EMBL	X54816; CAA38585.1; -; Genomi
EMBL	X54817; CAA38585.1; JOINED;
EMBL	X54818; CAA38585.1; JOINED;
EMBL	X54816; CAA38587.1; -; Genomi
EMBL	X04225; CAA27803.1; -; mRNA
EMBL	M88249; AAA59196.1; -; Genom
EMBL	M88165; AAA59196.1; JOINED,
EMBL	M88243; AAA59196.1; JOINED,
EMBL	M88244; AAA59196.1; JOINED,
EMBL	M88246; AAA59196.1; JOINED,
EMBL	M88247; AAA59196.1; JOINED,
EMBL	X04494; CAA28182.1; -; mRNA
EMBL	BC041593; AAH41593.1; -; mRN
EMBL	X54817; CAA38586.1; -; Genomi
PIR	S13433; HCHU
UniGene	Hs.436911; -
PDB	1BIK; X-ray; @=206-352
MEROPS	I02.005; -
MEROPS	I02.006; -
GlycoSuiteDB	P02760; -.
SWISS-2DPAGE	P02760; HUMAN
Siena-2DPAGE	P02760; -.
P	
E	
KI	
H	
HI	
M	
PI	
A	
G	

External 2D data: Estimated location of the prote

- How to interpret a protein map
- You may obtain an estimated location of the pr The estimation is obtained according to the co
- **Warning 1**: the displayed region reflects an ar the user's information. It should be used with c
- **Warning 2**: the 2D PAGE map is built on dem

SWISS-2DPAGE: ***Arabidopsis thaliana (Mouse-ear cress)***	• Arabidopsis thaliana
SWISS-2DPAGE: ***Dictyostelium discoideum (Slime mold)***	• Dictyostelium discoideum
SWISS-2DPAGE: ***Escherichia coli***	• Escherichia coli • Escherichia coli(4.5-5.5) • Escherichia coli(6-11)
SWISS-2DPAGE: ***Homo sapiens (Human)***	• Colorectal epithelia cells • Erythroleukemia Cell • Promyelocytic leukemia c • Lymphocytes • SDS-PAGE of nucleolar pr Human HeLa cells • Platelet

Searching in 'Test 2D-PAGE (Test-2DPAGE I)' for entry matching P02760

Test 2D-PAGE (Test-2DPAGE I): P02760

P02760

General information about the entry

View entry in simple text format

Entry name	**AMBP_HUMAN**
Primary accession number	**P02760**
integrated into Test-2DPAGE I on	December 14, 2005 (release 1)
2D Annotations were last modified on	September 27, 2006 (version 3)
General Annotations were last modified on	September 27, 2006 (version 7)

Name and origin of the protein

Description	**AMBP protein [Contains: Alpha-1-microglobulin (Protein HC) (Complex-forming glycoprotein heterogeneous in charge) (Alpha-1 microglycoprotein); Inter-alpha-trypsin inhibitor light chain (ITI-LC) (Bikunin) (HI-30)].**
Gene name	**Name=AMBP** Synonyms=HCP, ITIL
Annotated species	Oryctolagus cuniculus (Rabbit) [TaxID: 9986]
Taxonomy	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Glires; Lagomorpha; Leporidae; Oryctolagus.

References

[1] MAPPING ON GEL
Demalte-Annessi I., Sanchez J.-C., Hoogland C., Rouge V., Binz P.-A., Appel R.D., Hochstrasser D.F.
Submitted (JAN-1999) to the SWISS-2DPAGE database

[2] PATHOLOGICAL VARIANTS
Sharp H.L.
(IN) McKusick V.A., Clairbone R. (eds.)Medical genetics, pp.131-140, H. P. Publ. Co. Inc. (1973)

[3] MAPPING ON GEL
MEDLINE=94147969; PubMed=8313870; [NCBI, EBI, Israel, Japan]
Hughes G.J., Frutiger S., Paquet N., Pasquali C., Sanchez J.-C., Tissot J.-D., Bairoch A., Appel R.D., Hochstrasser D.F.
"Human liver protein map: update 1993"
Electrophoresis 14:1216-1222(1993)

2D PAGE maps for identified proteins (Click here to expand identification/mapping data)

Compute the theoretical pI/Mw
How to interpret a protein

PLASMA {Plasma Test Gel}
Oryctolagus cuniculus (Rabbit)
Tissue: Cervical

map experimental info

PLASMA

MAP LOCATIONS:

- SPOT 89: pI=4.86; Mw=33000 [identification data]
- SPOT 111: pI=4.96; Mw=32000 [identification data]
- SPOT 112: pI=5.07; Mw=31000 [identification data]
- SPOT 408: pI=5.12; Mw=23000 [identification data]

MAPPING (identification):

SPOT 89: Gel matching [1]; Peptide mass fingerprinting [2];
SPOT 111: Gel matching [1]; Peptide mass fingerprinting [2];
SPOT 112: Gel matching [1]; Immunobloting; Peptide mass fingerprinting [2]; Tandem mass spectrometry;
SPOT 408: Comigration; Gel matching [1]; Peptide mass fingerprinting [2]; Tandem mass spectrometry.

PLASMA2 {Plasma 2 Test Gel}
Oryctolagus cuniculus (Rabbit)
Tissue: Carotid artery

PLASMA2

MAP LOCATIONS:

- SPOT 89: pI=4.86; Mw=33000 [identification data]
- SPOT 111: pI=4.96; Mw=32000 [identification data]
- SPOT 112: pI=5.07; Mw=31000 [identification data]

Figure G.III-2: A protein entry in the '*Nice view*' format – displaying various sections.

Search by spot ID / serial number

With this option, it is possible to search for a particular spot, or to list all spots for a particular map.

Like many other search options, the search by spot ID displays a list of maps grouped by species from which the user can choose. The list includes all maps from local databases, as well as those from remote databases. Information covering the latter is dynamically extracted from the remote Web servers upon end-users' request and is stored inside the local Web server for a limited lap of time[1]. Extracting information dynamically from remote Web servers is a time-consuming task. Storing temporarily this information within the local Web server is a compromise between providing up-to-date data and guaranteeing a rapid execution of queries[2].

Search by identification method

Spots identified by a particular identification method can be listed using this search option. The search can combine several selected identification methods, and the result displays clickable hits that redirect to the corresponding experimental data.

Identification methods are defined in the *basic_include.pl* configuration file (F.II.3). The default list comprises many methods, ranging from PMF, Tandem MS, Amino Acid Composition, to Micro-Sequencing, Gel Matching, etc. All added user-defined identification methods are also part of this list. By default, the tool proposes this same list to end-users to select from. However, administrators can configure the tool to show a shorter list limited to the methods actually available for the currently selected local database. Restricting the identification methods to those from one local database is appropriate when querying only local databases, but it limits the search possibilities in case remote databases are queried concurrently.

Search by pI / Mw range

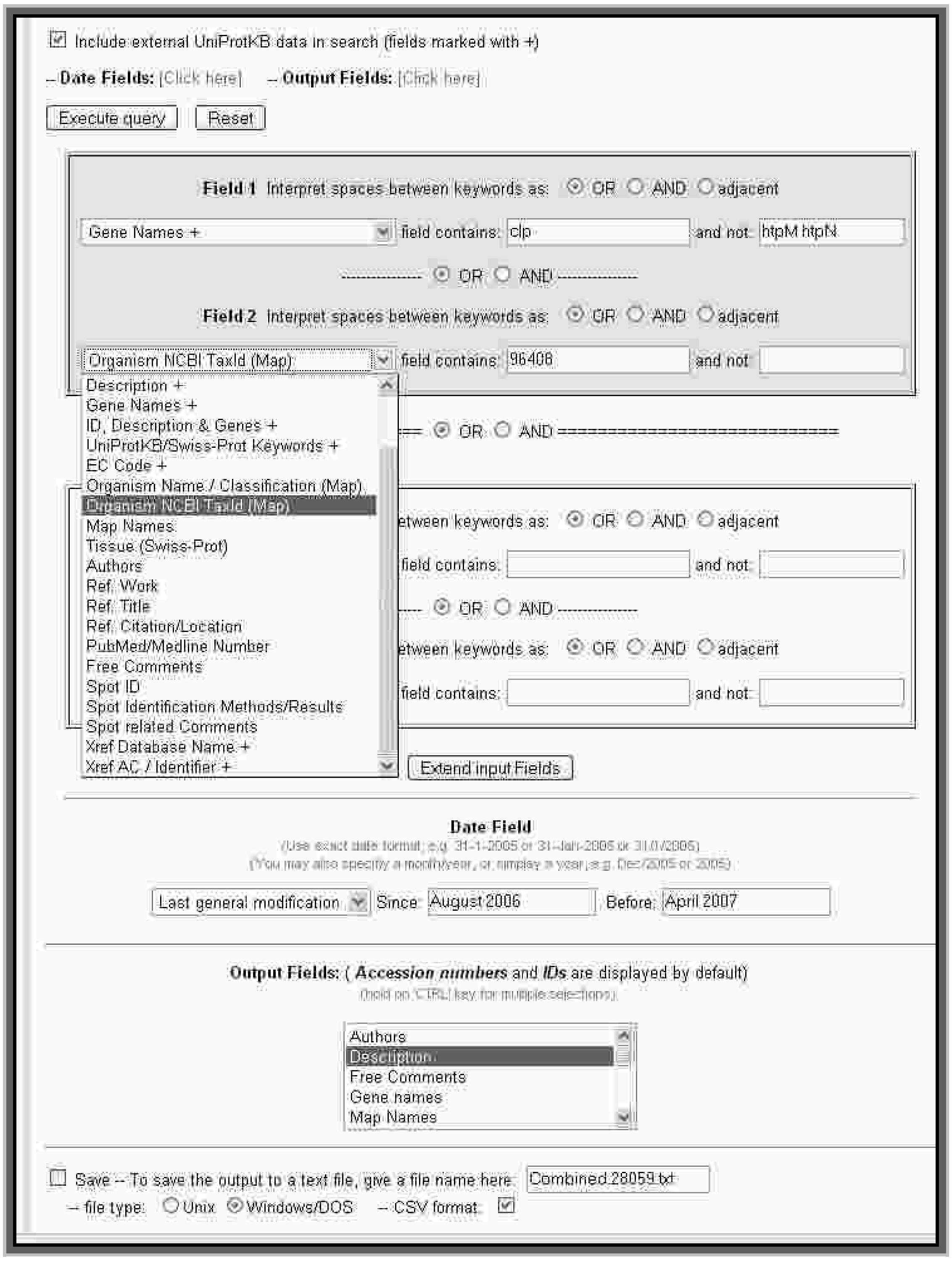
Spots that are located within a range of pI and Mw values can be retrieved using this search option. The pI range is ignored for SDS-PAGE gels.

Search by combined fields

The most exhaustive form of search engines is the search by combined fields interface. End-users can build their most complex queries by means of a SRS-like query interface (Figure G.III-3). Search by keywords can be performed against any type of data and can be combined in any order using logical operators. Inclusive and exclusive statements can be expressed. Input fields are grouped in blocks – whose number can be extended - to simulate parenthesis segments, like in a logical expression. Hits may be filtered to only retain those prior and/or subsequent to defined dates, regarding entries creation and annotation modifications. Moreover, end-users can extend the parsed data types to include related UniProtKB data. They may choose the output data types and may export results into plain or tab-delimited text files.

[1] By default, information related to the remote maps expires within 72 hours of local storing.

[2] Extraction is limited to remote Web servers, and is deactivated for remote Web portals that have no local databases.



[✓] Include external UniProtKB data in search (fields marked with +)

-- **Date Fields:** [Click here] -- **Output Fields:** [Click here]

Execute query | Reset

Field 1 Interpret spaces between keywords as: [✓ OR] AND adjacent

Gene Names + | field contains: clp | and not: htpM htpN

--------------- [✓ OR] AND ---------------

Field 2 Interpret spaces between keywords as: [✓ OR] AND adjacent

Organism NCBI TaxId (Map) | field contains: 96408 | and not:

- Description +
- Gene Names +
- ID, Description & Genes +
- UniProtKB/Swiss-Prot Keywords +
- EC Code +
- Organism Name / Classification (Map)
- Organism NCBI TaxId (Map)
- Map Names
- Tissue (Swiss-Prot)
- Authors
- Ref. Work
- Ref. Title
- Ref. Citation/Location
- PubMed/Medline Number
- Free Comments
- Spot ID
- Spot Identification Methods/Results
- Spot related Comments
- Xref Database Name +
- Xref AC / Identifier +

=== [✓ OR] AND ==============================

etween keywords as: [✓ OR] AND adjacent

field contains: | and not:

---- [✓ OR] AND ---------------

etween keywords as: [✓ OR] AND adjacent

field contains: | and not:

Extend input Fields

Date Field

(Use exact date format, e.g. 31-1-2005 or 31-Jan-2005 or 31/01/2005)

(You may also specify a month/year, or simply a year, e.g. Dec/2005 or 2005)

Last general modification | Since: August 2006 | Before: April 2007

Output Fields: (***Accession numbers*** and ***IDs*** are displayed by default)

(hold on 'CTRL' key for multiple selections)

- Authors
- Description
- Free Comments
- Gene names
- Map Names

☐ Save -- To save the output to a text file, give a file name here: Combined.28059.txt

-- file type: Unix [✓ Windows/DOS] -- CSV format: [✓]

Figure G.III-3: The 'search by combined fields' interface – A SRS-like search engine.

Thanks to the use of this interface, particularly complex queries can be expressed, like for example:

"List all *Homo sapiens* identified proteins in liver that are involved in ATP-binding activity, and for which the gene names contain 'clp' but not 'htpM' or 'htpN'. The work must have been published by author 'Smith' in journal 'Proteomics'. Spots must have been identified by mass spectrometry and the identified proteins should have a cross-reference to UniProtKB. Display only those that have been modified since last year".

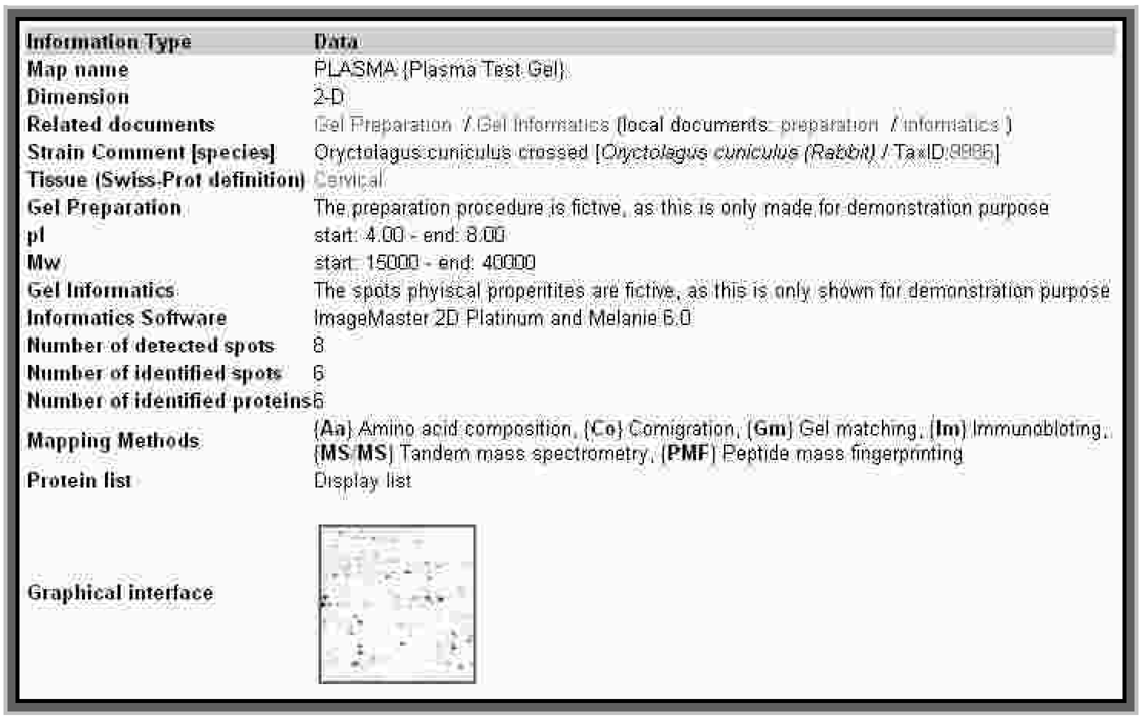
We may argue that it might have been better to group all the search options in a unique search interface especially as, in fact, the search by combined fields engine can effectively include all the mentioned search options. However, we have estimated that for a simple access, and in order to refine some specific queries, proposing separated simple search options makes the use of the search interface easier and more intuitive. In addition, extending the currently available list of search options by adding new options is a straightforward task for developers. The prospective extension depends only on users' needs and feedback.

Map access

This part is specific to data related to available maps of local and remote databases.

Experimental info

This option displays experimental information, origin, links to gel preparation protocols and gel informatics, applied identification methods, and map's metadata for a selected map (Figure G.III-4).



Information Type	Data
Map name	PLASMA {Plasma Test Gel}
Dimension	2-D
Related documents	Gel Preparation / Gel Informatics (local documents: preparation / informatics)
Strain Comment [species]	Oryctolagus cuniculus crossed [*Oryctolagus cuniculus (Rabbit)* / TaxID:9986]
Tissue (Swiss-Prot definition)	Cervical
Gel Preparation	The preparation procedure is fictive, as this is only made for demonstration purpose
pI	start: 4.00 - end: 8.00
Mw	start: 15000 - end: 40000
Gel Informatics	The spots phyiscal properitites are fictive, as this is only shown for demonstration purpose
Informatics Software	ImageMaster 2D Platinum and Melanie 6.0
Number of detected spots	8
Number of identified spots	6
Number of identified proteins	6
Mapping Methods	{**Aa**} Amino acid composition, {**Co**} Comigration, {**Gm**} Gel matching, {**Im**} Immunobloting, {**MS/MS**} Tandem mass spectrometry, {**PMF**} Peptide mass fingerprinting
Protein list	Display list
Graphical interface	

Figure G.III-4: Map's experimental information.

Protein list

The protein list option lists in a table all the protein entries identified for a given map, with related 2-DE information: spot identifiers, experimental pI, Mw and relative volume and density (read from gel). Identification procedures and their corresponding annotations are also listed, as well as all bibliographic references. The output table can be exported as a plain or a tab-delimited text file.

Graphical viewer

The graphical viewer button directs to the map navigator interface, which is a standalone viewer to navigate through the various local maps.

Container for unidentified spots

For all the previously mentioned search engines, database administrators may allow privileged end-users to access unidentified spots or spots to which no protein has been yet definitively assigned. The access also includes the spots' analysis data hidden from public end-users. Make2D-DB II presents unidentified spots to privileged end-users within a unique container, similar to a protein entry, which has the identifier '*UNIDENTIFIED_SPOTS*'[1].

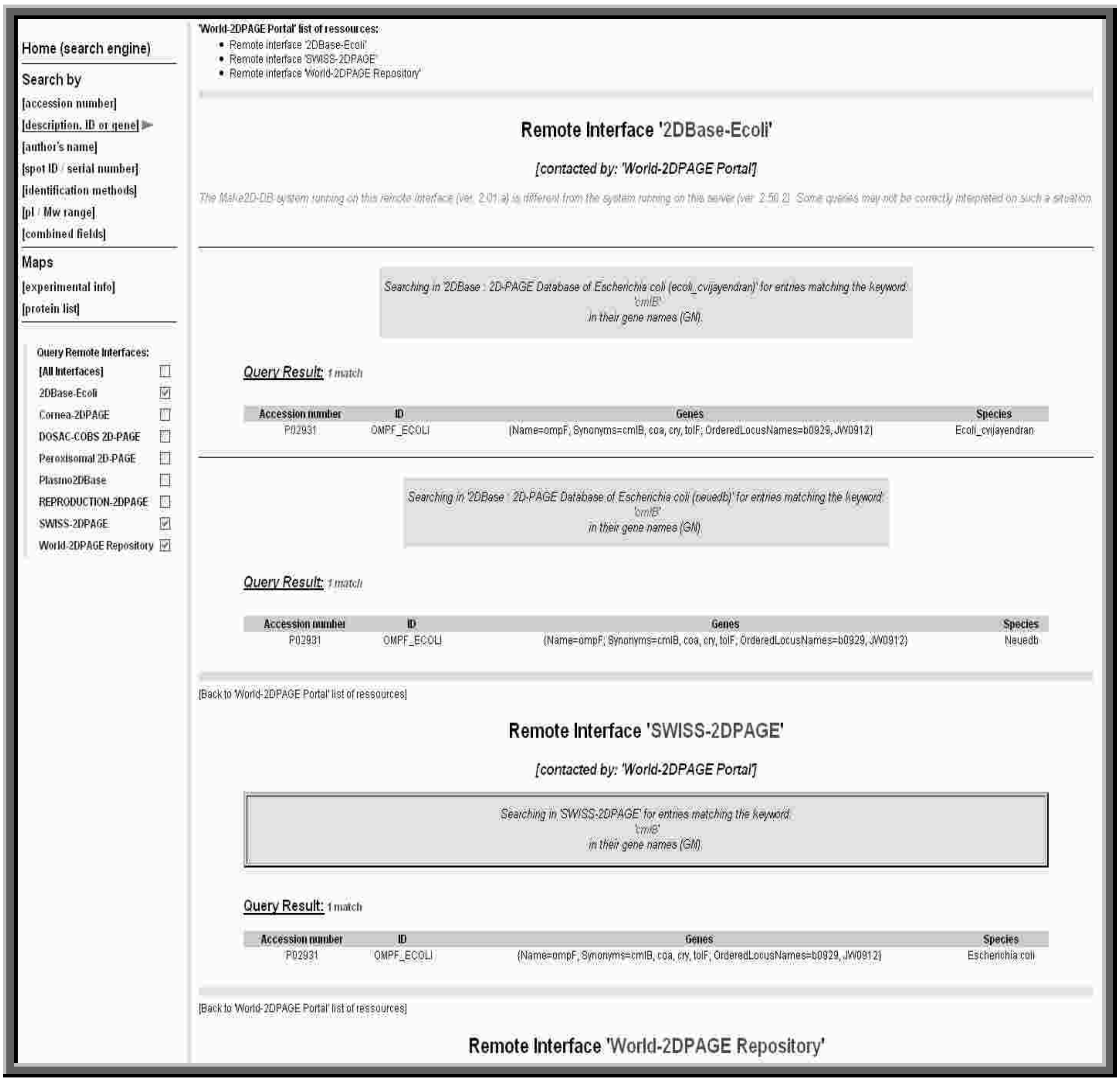
Databases' selection area

Web servers that access several local and remote databases present a list of available resources in the databases' selection area. Users can select one or several local databases and/or remote Web servers (interfaces) to query at once. The output result clearly states the origin of the resulting hits and permits further navigation through related information (Figure G.III-5). Depending on the Make2D-DB II version running on the remote Web server, the calling Web server is able to estimate when a specific query can be fully, partially, or not at all interpreted by the remote server and it tries therefore to adapt the query. A message addressed to end-users indicates potential incompatibilities each time the version of a contacted Web server is prior to that of the calling server.

Accessing simultaneously too many remote databases is an issue that should be considered. The time needed to establish HTTP connections depends on several factors (Web traffic, number of remote Web servers, accessibility and size of remote databases, the amount of exchanged data, etc.). Besides, selecting remote Web portals implies an additional amount of time for these portals to contacts their associated remote Web servers. We have experimented many different situations. For example, we simultaneously contacted a dozen of local databases and remote Web servers, as well as a Web portal that contacts, in its turn, eight remote Web servers. The result was rather satisfactory, as the waiting time for a moderate query (searching '*atp*' in protein annotations) did not exceed 60 seconds. Nonetheless, the issue must be taken into account, especially to anticipate future situations where a much larger number of

[1] By connecting to the Test-2DPAGE database and providing the privileged users password '*private*', readers can access this unidentified spots' container. The container becomes visible, for example, by listing all the proteins using the search by accession number option.

remote Make2D-DB II Web servers and portals will become available all over the Web. Some technical solutions are proposed in the chapter covering the tool's perspectives.



'World-2DPAGE Portal' list of ressources:

- Remote interface '2DBase-Ecoli'
- Remote interface 'SWISS-2DPAGE'
- Remote interface 'World-2DPAGE Repository'

Remote Interface '2DBase-Ecoli'

[contacted by: 'World-2DPAGE Portal']

The Make2D-DB system running on this remote interface (ver. 2.01.a) is different from the system running on this server (ver. 2.50.2). Some queries may not be correctly interpreted on such a situation.

Searching in '2DBase : 2D-PAGE Database of Escherichia coli (ecoli_cvijayendran)' for entries matching the keyword 'cmlB' in their gene names (GN).

Query Result: 1 match

Accession number	ID	Genes	Species
P02931	OMPF_ECOLI	{Name=ompF; Synonyms=cmlB, coa, cry, tolF; OrderedLocusNames=b0929, JW0912}	Ecoli_cvijayendran

Searching in '2DBase : 2D-PAGE Database of Escherichia coli (neuedb)' for entries matching the keyword 'cmlB' in their gene names (GN).

Query Result: 1 match

Accession number	ID	Genes	Species
P02931	OMPF_ECOLI	{Name=ompF; Synonyms=cmlB, coa, cry, tolF; OrderedLocusNames=b0929, JW0912}	Neuedb

[Back to 'World-2DPAGE Portal' list of ressources]

Remote Interface 'SWISS-2DPAGE'

[contacted by: 'World-2DPAGE Portal']

Searching in 'SWISS-2DPAGE' for entries matching the keyword 'cmlB' in their gene names (GN).

Query Result: 1 match

Accession number	ID	Genes	Species
P02931	OMPF_ECOLI	{Name=ompF; Synonyms=cmlB, coa, cry, tolF; OrderedLocusNames=b0929, JW0912}	Escherichia coli

[Back to 'World-2DPAGE Portal' list of ressources]

Remote Interface 'World-2DPAGE Repository'

Figure G.III-5: Querying several remote Make2D-DB II Web servers at once.

G.III.2 Referencing or extracting data in the non-interactive mode

The non-interactive mode is intended to reference objects, as well as to perform data integration operations.

Logical and physical URLs

The Apache mode rewrite mode[1,2] offers a powerful URL manipulation mechanism through the rewriting of the requested URL on the fly based on configuration directives and rules. This feature lets a HTTP server understand **logical URLs** and translate them into physical URLs that reflect the architecture of the server directory tree[3]. Logical URLs have the advantage of being descriptive, self-explanatory and intuitive, as opposed to physical URLs, which point to a physical Web page or to a Web script with a set of optional arguments.

In Make2D-DB II, logical URLs are theoretically formed by a URI[4] (Uniform Resource Identifier), which locates a specific object over the Web, as well as a number of optional and definite directives to apply to that object in a REST[5] manner. An object can be a map, a spot, an identification experiment or a protein[6], while the associated directives define the desired format, the purpose and the actions to perform. Formats are either plain text, HTML, or XML formats. The purpose represents the aim: a machine extraction, a text-based display or a graphical representation, while the action to perform is applied to limit the extracted data to a subpart of the data and/or to extend it with additional data. In Make2D-DB II, logical URLs follow a very simple and intuitive formulation, in which objects, formats, purposes and actions are pre-defined terms that could be extended should the need arise.

Physical URLs point simply to the script of the main Web interface (*2d.cgi*), followed by a set of arguments that defines the object and the associated directives in a manner close to the logical URLs' logic. Logical and physical URLs coexist, which means that the tool is able to interpret both syntaxes.

Derived objects are additional "objects" other than regular maps, spots, identification experiments and protein objects. Either they are derived entities that are extracted from a Make2D-DB II Web server, for data integration purpose, or they are reserved keywords used to formulate search queries. In both cases, they are considered as an answer to a question and are treated using the same logic as with logical objects. Therefore, in order to extract a derived object, a question mark "*?*" in the logical URL is needed to formulate the appropriate query.

The knowledge of the formalism used in the non-interactive mode is essential in data extraction and data integration operations, and the use of logical URLs rather than physical URLs ensures much more stability. Table G.III-1 lists some representative examples of logical and physical URLs' formulation. By convention, variable parameters are in italic and all URLs are lowercased. Many URLs can be expressed in

[1] http://mordor.isb-sib.ch/make2ddb/readme/4.Readme_interface.html#modeRewrite

[2] http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html

[3] *cf.* glossary: URL.

[4] *cf.* glossary: URI

[5] *cf.* glossary.

[6] Objects may be extended to other entities, like projects, bibliographic references, etc.

different manners: a formal manner and convenient shortcuts (informal). Arguments are separated by "&" characters and may have aliases (*e.g.*, *database* and *db*). Several values can be given to an argument using "+" signs (*e.g.*, *database*=project1+project2) or using the keyword 'all'. For the formal format, the order of arguments does not have any consequence on the interpretation of the URL.

Table G.III-1: Representative examples of logical URLs' formulation.

Purpose	Access directly a specific protein entry object from a Web server that manages a single local database[1]. The protein is designated using its accession number. By default, format is HTML and purpose is text display. No special action is performed.
Physical URL	http://domain/cgi-bin/*2d-server*/2d.cgi?ac=*accession*°
Logical URL	*Formally*: http://domain/*2d-server*/protein/*accession*° *Shortcut*: http://domain/*2d-server*/entry/*accession*° *Shortcut*: http://domain/*2d-server*/*accession*°
Example	http://www.expasy.org/swiss-2dpage/P18335 (informal)
Purpose	Extract a specific entry from a local database (project) that is part of a Web interface managing several local databases[2]. Format is raw (unformatted text) and the purpose is an extraction for machine processing (*cf.* Web page source).
Physical URL	http://domain/cgi-bin/*2d-server*/2d.cgi?database=*project*&ac=*accession_number* &format=raw&extract
Logical URL	*Formally*: http://domain/*2d-server*/*accession*°&database=*project*&format=raw&extract *A shortcut*: http://domain/*project*/*accession*°&format=raw&extract
Example	http://mordor.isb-sib.ch/test-2dpage_ii/P58771&format=raw&extract (informal)
Purpose	View a specific map (graphical representation). Scale the image to *n* and include some (or "all") identified proteins in the display.
Physical URL	http://domain/cgi-bin/*2d-server*/2d_view_map.cgi?map=*map*&ac=*accession*°&scale=*n*
Logical URL	http://domain/*2d-server*/map/*map*&ac=*accession*°&view=image&scale=*n*
Example	http://www.expasy.org/swiss-2dpage/map/ecoli&ac=all&view=image&scale=2

[1] In single database installations, the Web server and the local database generally share the same name (2d-server / database); *e.g.,* "swiss-2dpage" refers to both the SWISS-2DPAGE database and its Web server's reference name.

[2] For Web servers that manage several local databases, in case the database argument is omitted, the first local database (the default database) is then automatically selected..

Purpose	For a defined spot, display all data related to a specific identification method. A spot identifier is defined in Make2D-DB II using the identifier of the map containing the spot, followed by a semicolon and the local identifier of the spot within its map (*cf.* E.V.4).
Physical URL	http://domain/cgi-bin/*2d-server*/2d.cgi?db=*project*&spot=*spotID*&data=*identification*
Logical URL	*Formally*: http://domain2d-server/spot/spot=*spotID*&data=*identification*&db=*project* *A shortcut*: http:// domain/*project*/spot/*spotID*&data=*identification*
Example	http://mordor.isb-sib.ch/2d_test/spot/spot=plasma2:89&data=msms&db=test-2dpage_i
Purpose	Extracting a derived object.
Physical URL	http:// domain/cgi-bin/*2d-server*/2d.cgi?*DerivedObject*[&*argument*[=*value*]]
Logical URL	http://domain/*2d-server*/?*DerivedObject*[&*argument*[=*value*]]
Examples	http://www.expasy.org/swiss-2dpage/?maplist&format=text (display a list of all maps) http://www.expasy.org/swiss-2dpage/?author=appel&extract (extract all entries for author "appel")

While installing a Web server, Make2D-DB II automatically generates the complete set of rewriting rules and functions needed by the HTTP server in order to interpret logical URLs and to suitably redirect them to the corresponding physical URLs. The rules depend on the user's configuration parameters, as well as on the number of managed local databases. Activating the rules is a user's choice and it only requires him/her to include them within the HTTP server configuration file. An example of generated rules and functions set for a Web server *2d* that manages two databases *test_2dpage_i* and *test_2dpage_ii* can be found at:

- http://mordor.isb-sib.ch/make2ddb/readme/examples/mod_rewrite.txt
- http://mordor.isb-sib.ch/make2ddb/readme/examples/make2db_map.txt

Adopting a logical syntax to access or extract objects enhances and simplifies the interconnection of remote resources in a federated system. This is the reason why we are trying to reinforce federation rules - as described in C.IV.2 - by proposing to extract and to refer to objects using simple but expressive URIs. Another obvious advantage, besides the intuitive way to extract objects and to formulate queries, is the ability to unambiguously refer to each object in any type of reports or publications.

Referring to the search engines

The search engines described in G.III.1 can each be directly triggered using logical URLs of the form:

- http://domain/database/?*option* (*e.g.,* http://www.expasy.org/swiss-2dpage/?ac)

Where *option* is one of the self-explanatory keywords: *ac*, *id*, *de*, *ge*, *author*, *spot*, *ident*, *pi_mw*, *combined*, *map* and *mapExpInfo*, which correspond to the various search options.

Data exchange

The distributed Web servers interact and exchange data using HTTP GET and POST queries in a REST manner. The interaction is performed by means of the formalism presented in the previous section, whether the logical URLs' mode is set active or is disabled.

Currently, the data exchange process relies on a defined set of statements that are specific to the federated Make2D-DB II systems. Data is retrieved mostly in a semi-structured text format that the tool knows how to interpret. Besides being able to remotely perform all standard search options on any remote interface, a local Web server can also extract additional data (or metadata) regarding the remote installations by using the *extract* purpose. This additional data is considered derived objects and includes the version of the remote Make2D-DB II installation (using the keyword *make2d*), a list of the remote databases (*databaseList*), a list of all maps and their related species and annotations (*mapList*), database statistics (stats), etc. An important feature is the ability to retrieve all identified proteins that have a cross-reference to the UniProtKB main index[1] or to the SWISS-2DPAGE secondary index[2] (*index*[=*indexNumber*°], *cf.* also E.V.7). The importance resides in the fact that federated or third-party systems and resources, such as the distributed Make2D-DB II systems and UniProtKB, can easily establish cross-references between their protein entries, their studied species or their maps, and those from the contacted Make2D-DB Web server.

The current list of extractible data and the output formats cover the vital requirements to interconnect distributed Make2D-DB II systems and to link to any object within these systems. Further developments of the tool should expand the interaction, not only among Make2D-DB II systems, but also between Make2D-DB II and third-party systems, such as the Proteome Database System (C.IV.5) and PROTICdb (C.IV.6). This will certainly imply more derived objects, such as the GO terms (E.V.6: Gene ontology classification), more directives, and a more generic format for data output (preferably a generic XML format). Extending the extraction capabilities is a straightforward task for developers. Each object, or derived object, needs to be first recognised by the interface. A set of SQL queries associated with this

[1] *e.g.*, page source of http://mordor.isb-sib.ch/cgi-bin/2d_test/2d.cgi?index&database=test-2dpage_i&extract

[2] *e.g.*, page source of http://mordor.isb-sib.ch/cgi-bin/2d_test/2d.cgi?index=3&database=test-2dpage_ii&extract

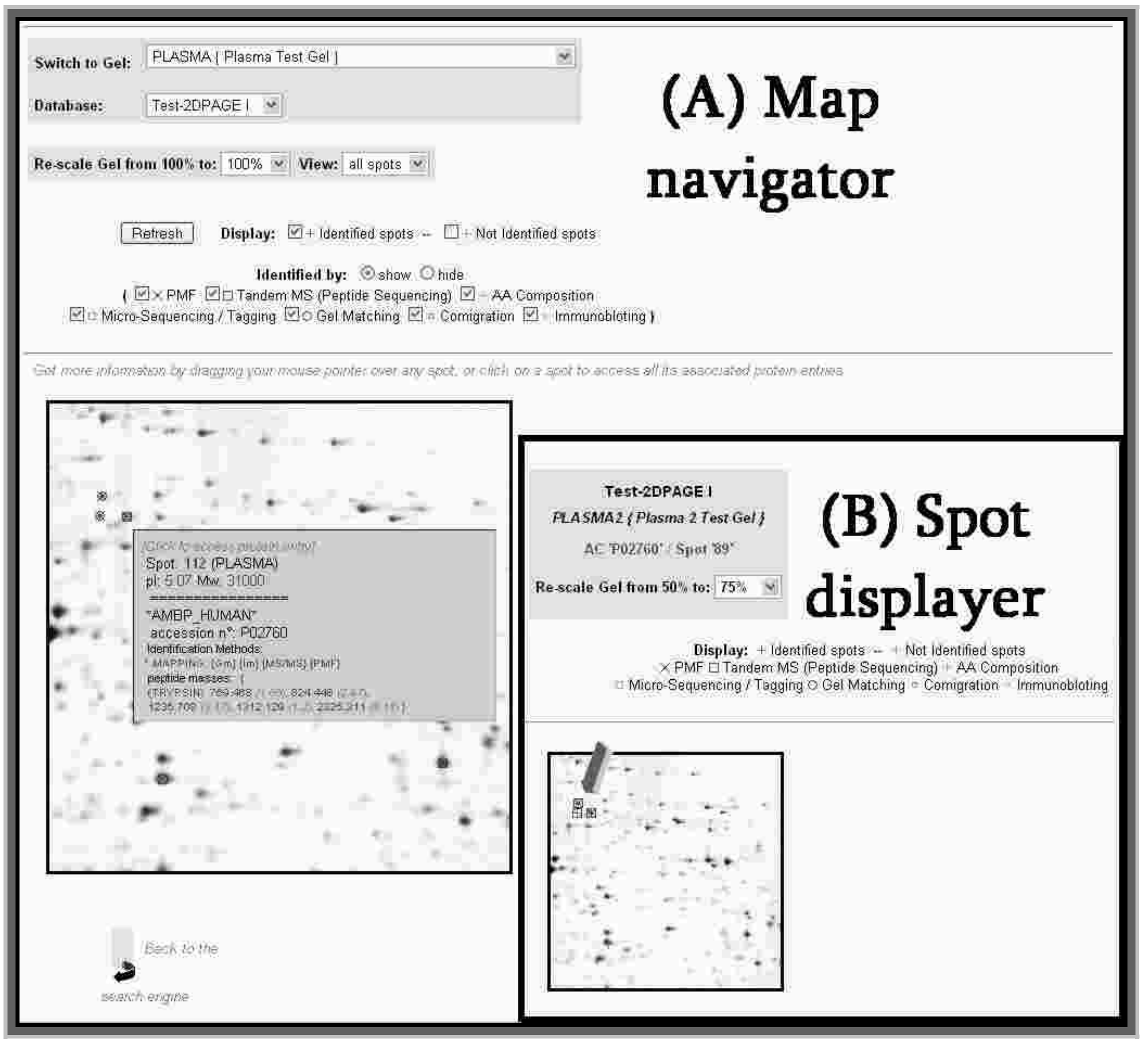
object is generated, based on the input directives, and is then executed. The result is handed to the appropriate output routine, depending on the required format.

G.IV. Data viewers

There are three viewers, generally accessed from the main interface. Viewers are associated with different objects and can be interactively triggered by clicking on their related objects.

The map navigator

In a 2D-PAGE environment, it is essential to visualise the maps and the physical location of the spots that have been identified. The map navigator is a standalone graphical viewer that is intended to visualise and navigate through the maps of all local databases (Figure G.IV-1-A). The viewer displays an image of the selected map, which can be rescaled. Identified spots - and upon request unidentified spots - are displayed. Identified spots are marked thanks to a range of different symbols that visually indicate their identification methods. Depending on the user's choice, the view can be limited to one particular protein, or to spots that have been identified by one or several identification methods.



Switch to Gel: PLASMA { Plasma Test Gel }
Database: Test-2DPAGE I
Re-scale Gel from 100% to: 100% View: all spots
Display: ☑ + Identified spots -- ☐ + Not Identified spots
Identified by: ◉ show ○ hide
(☑ × PMF ☑ □ Tandem MS (Peptide Sequencing) ☑ AA Composition
☑ Micro-Sequencing / Tagging ☑ ○ Gel Matching ☑ Comigration ☑ Immunobloting)

Get more information by dragging your mouse pointer over any spot, or click on a spot to access all its associated protein entries

Spot: 112 (PLASMA)
pI: 5.07 Mw: 31000
"AMBP_HUMAN"
accession n°: P02760
Identification Methods:
MAPPING: {Gm} {Im} {MS/MS} {PMF}
peptide masses: {
(TRYPSIN) 769.468 [illegible] 824.448 [illegible]
1235.709 [illegible] 1312.129 [illegible] 2325.311 [illegible] }

Test-2DPAGE I
PLASMA2 { Plasma 2 Test Gel }
AC 'P02760' / Spot '89'
Re-scale Gel from 50% to: 75%
Display: + Identified spots -- + Not Identified spots
× PMF □ Tandem MS (Peptide Sequencing) AA Composition
Micro-Sequencing / Tagging ○ Gel Matching Comigration Immunobloting

Figure G.IV-1: The map navigator (A) and the spot displayer (B).

Dragging the mouse pointer over any spot displays a small box that shows the spot physical properties and all its assigned proteins. This box also shows a list of all identification methods that have been applied to the spot, as well as a summary of available experimental data. To access the full annotations related to a spot and its assigned proteins, one can simply apply a mouse click over the spot. The viewer is also accessible from the main interface and can be triggered by clicking on a spot, which displays the location of the clicked spot over its map and, optionally, all potential isoforms of the protein(s) assigned to that spot (Figure G.IV-1-B).

Referencing objects in the graphical representation can be done by a formalism similar to the one used in the non-interactive mode of the main interface (to visualise an entire map, a single spot, all spots belonging to a specific protein, etc.). The map navigator can be directly accessed via the address:

- http://domain/2d-server/2d_view_map.cgi

or, with logical URLs activated:

- http://domain/2d-server/viewer (*e.g.*, http://mordor.isb-sib.ch/2d_test/viewer)

The mass spectrometry peak list viewer

This viewer is a small utility that display a quick view of the mass spectrometry peak lists when the latter are listed within the main interface. It is a java based program, and is therefore present in most standard installations.

The mass spectrometry browser (GDchart)

Like the map navigator, the mass spectrometry browser is an interactive standalone viewer that is also attached to the main interface (Figure G.IV-2). The browser is accessible whenever the main interface displays mass spectrometry identification results. It displays information regarding the MS identification and the local identifiers assigned to it. It also distinctively displays retained and non-retained peaks if the distinction is given by data providers. Being built over some graphical C libraries, the viewer is the only Make2D-DB II component that requires compiling non-standard components. Therefore, it might be not present in some Make2D-DB II installations.

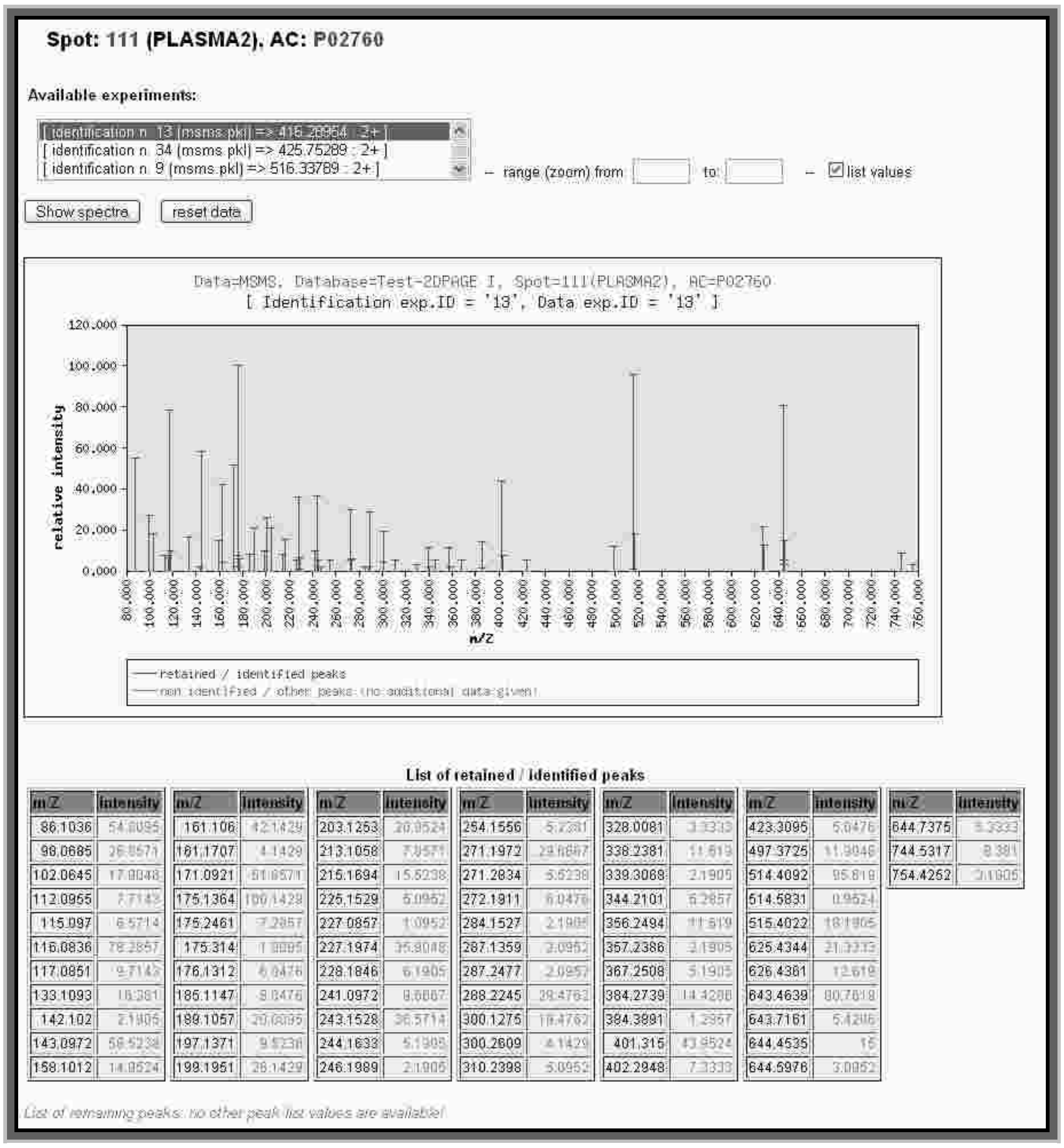
In the standalone mode[1], the browser lets users surf between all PMF and tandem MS experimental data and identification analysis regarding any spot. For a selected spot, all its related MS identifications are listed with their identifiers, and each spectrum can be individually displayed. Like many other objects within the Make2D-DB II environment, the spectrum of a mass spectrometry analysis can be directly referenced by a unique URL. For instance, in the following URL:

- http://mordor.isb-sib.ch/cgi-bin/2d_test/GDchart.cgi?*spectra* &*database*=test_2page_i&*spot*=111&*map*=plasma2&*ac*=P02760 &*data*=msms&expid=13&*identid*=13&*range*=180-430

We designate a spectrum object and we select an identified spot (by its local database, map identifier, local spot identifier and identified protein). We name the type of data (*msms* for tandem MS), the experiment / spectrum identifier and the identification analysis identifier (as several identification analyses may be associated to one MS spectrum, *cf.* E.V.5). We can also define a range of m/Z values to limit the display to a subset of peaks[2].

[1] *cf.* http://mordor.isb-sib.ch/cgi-bin/2d_test/GDchart.cgi

[2] In future developments, we may also add a threshold value or a range for the intensities to filter the peaks.



Spot: 111 (PLASMA2), AC: P02760

Available experiments:

[identification n. 13 (msms.pkl) => 416.28954 : 2+]
[identification n. 34 (msms.pkl) => 425.75289 : 2+]
[identification n. 9 (msms.pkl) => 516.33789 : 2+]

– range (zoom) from [ ] to: [ ] – ☑ list values

Data=MSMS, Database=Test-2DPAGE I, Spot=111(PLASMA2), AC=P02760
[Identification exp.ID = '13', Data exp.ID = '13']

——retained / identified peaks
——non identified / other peaks (no additional data given)

List of retained / identified peaks

m/Z	Intensity	m/Z	Intensity	m/Z	Intensity	m/Z	Intensity	m/Z	Intensity	m/Z	Intensity	m/Z	Intensity
86.1036	54.8095	161.106	42.1429	203.1253	20.9524	254.1556	5.7381	328.0081	3.3333	423.3095	5.0476	644.7375	5.3333
98.0685	26.8571	161.1707	4.1429	213.1058	7.8571	271.1972	24.6667	338.2381	11.619	497.3725	11.9048	744.5317	8.381
102.0645	17.9048	171.0921	51.8571	215.1694	15.5238	271.2834	5.5238	339.3068	2.1905	514.4092	95.619	754.4262	2.1905
112.0955	7.7143	175.1364	100.1429	225.1529	5.0952	272.1911	6.0476	344.2101	6.2857	514.5831	0.9524		
115.097	6.5714	175.2461	7.2857	227.0857	1.0952	284.1527	2.1905	356.2494	11.619	515.4022	18.1905		
116.0836	78.2857	175.314	1.8095	227.1974	35.9048	287.1359	2.0952	357.2386	3.1905	625.4344	21.3333		
117.0851	9.7143	176.1312	6.0476	228.1846	6.1905	287.2477	2.0952	367.2508	5.1905	626.4381	12.619		
133.1093	18.381	186.1147	9.0476	241.0972	8.6667	288.2245	39.4762	384.2739	14.4286	643.4639	80.7619		
142.102	2.1905	189.1057	20.8095	243.1528	36.5714	300.1275	18.4762	384.3891	1.2857	643.7161	5.4286		
143.0972	58.5238	197.1371	9.5238	244.1633	5.1905	300.2609	4.1429	401.315	43.9524	644.4535	15		
158.1012	14.9524	198.1951	26.1429	246.1989	2.1905	310.2398	5.0952	402.2948	7.3333	644.5976	3.0952		

List of remaining peaks: no other peak list values are available!

Figure G.IV-2: The mass spectrometry browser.

G.V. The administration interface

Local databases are managed and updated - with regard to external resources - by means of the administration interface, which can be accessed from the main interface, or directly at the URL:

- *http://domain/2d_server/admin/*

Database administrators enjoy full permissions to perform all administration tasks by providing the required PostgreSQL connection parameters to login. Two parameters are required: the database owner identifier and his/her password. When several local databases are managed by the same Web server, the PostgreSQL database name must also be given. The login procedure is a time-dependant secured process that encrypts the login parameters and makes any connection available during only one hour. Connections also expire after 30 minutes of inactivity[1].

In order to offer readers of this manuscript an access to the functionalities of the administration interface, we have reduced the security requirements to connect to one of the local databases of the Test-2DPAGE Web server at:

- http://mordor.isb-sib.ch/2d_test/admin/

Readers are required to provide the following connection parameters to login: *ID* = '*select2d*', *pass* = '*select2d*' and *database* = '*test_2dpage_iii*'. This login ensures an access to the administration interface and to all private data of Test-2DPAGE III, but without the ability to modify or to update the database content[2].

G.V.1 The search options

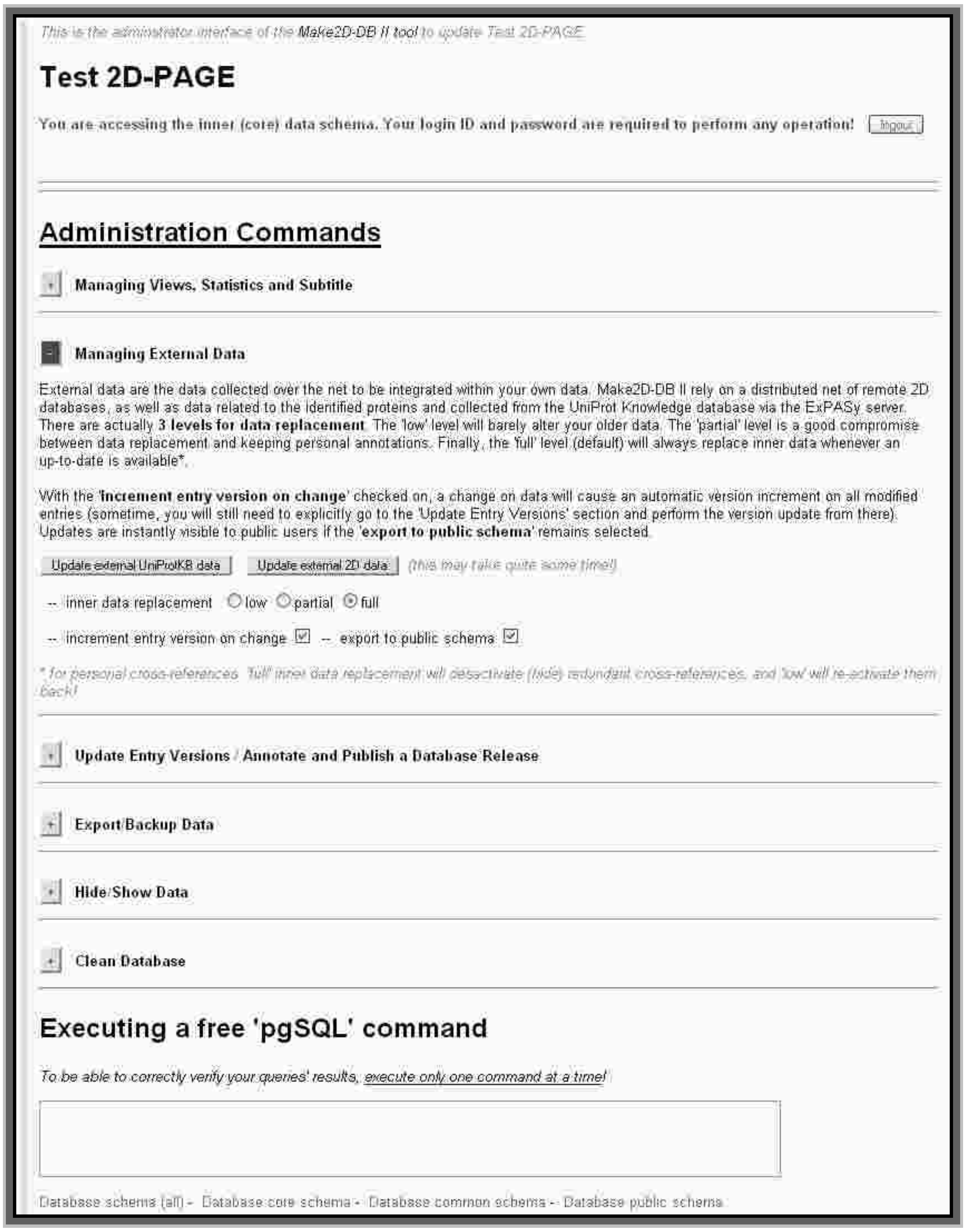
Since the administration interface is an extended clone of the main interface, all the search options of the main interface are available, except that only the local database that is being updated is active. Data is retrieved from the relational *core* schema and not from the *public* schema (*cf.* E.IV), which means that all private data and annotations of unidentified spots are accessible. Data viewers launched from the administration interface also give full access to the *core* schema and to all its private data.

G.V.2 The administration commands

Different update commands are available and can be performed independently. Highlighting any of these commands expands an associated area with related options (Figure G.V-1).

[1] Depending on how security parameters for TCP/IP connections are configured for the running PostgreSQL server, some installations may give access to the main page of the administration interface without verifying PostgreSQL connection parameters. Nevertheless, no data access or data modification will be allowed unless the correct parameters are provided.

[2] Readers who are interested in having full permissions, in order to test the effects of performing updates and data modifications, may contact us to obtain the appropriate login parameters.



This is the administrator interface of the Make2D-DB II tool to update Test 2D-PAGE

Test 2D-PAGE

You are accessing the inner (core) data schema. Your login ID and password are required to perform any operation! [logout]

Administration Commands

[+] **Managing Views, Statistics and Subtitle**

[−] **Managing External Data**

External data are the data collected over the net to be integrated within your own data. Make2D-DB II rely on a distributed net of remote 2D databases, as well as data related to the identified proteins and collected from the UniProt Knowledge database via the ExPASy server. There are actually **3 levels for data replacement**. The 'low' level will barely alter your older data. The 'partial' level is a good compromise between data replacement and keeping personal annotations. Finally, the 'full' level (default) will always replace inner data whenever an up-to-date is available*.

With the '**increment entry version on change**' checked on, a change on data will cause an automatic version increment on all modified entries (sometime, you will still need to explicitly go to the 'Update Entry Versions' section and perform the version update from there). Updates are instantly visible to public users if the '**export to public schema**' remains selected.

[Update external UniProtKB data] [Update external 2D data] *(this may take quite some time!)*

-- inner data replacement ○ low ○ partial [✓ full]

-- increment entry version on change ✓ -- export to public schema ✓

** for personal cross-references. 'full' inner data replacement will desactivate (hide) redundant cross-references, and 'low' will re-activate them back!*

[+] **Update Entry Versions / Annotate and Publish a Database Release**

[+] **Export/Backup Data**

[+] **Hide/Show Data**

[+] **Clean Database**

Executing a free 'pgSQL' command

To be able to correctly verify your queries' results, execute only one command at a time!

Database schema (all) - Database core schema - Database common schema - Database public schema

Figure G.V-1: The administration interface main page (the administration commands).

Managing views, statistics and subtitles

Views and Web server location

As described in E.V.10, views are collections of diverse data grouped together to form a particular data representation that is displayed to end-users (*e.g.*, a protein entry). After modifying or updating the content of the *core* schema, administrators choose the moment when to reflect their changes on the different views. Updating the views can be restricted to the *core* schema, which means that changes will become visible only to administrators and privileged users. At the chosen moment, the entire *core* schema content, including the views, can be exported to the *public* schema, thus making changes also visible to public end-users.

The update functions also detect any change of the Web server location (URL), and consequently update the *DynamicRemoteMake2DDBInterface* table. Updating this table is valuable for future development plans regarding an increased independence of distributed Make2D-DB II systems, as described in E.V.8: Connecting to remote 2-DE databases.

Statistics and Web server subtitles

From this area, administrators can maintain up-to-date statistics and edit their Web server subtitles. Statistics and subtitles are visible to end-users when they access the Web server home page. They serve to describe and to display facts about the local databases.

Managing external data

Integrating data from external resources into the database (*cf.* F.III) is performed from this area. Two distinct categories are available and can be executed separately:

UniProtKB / General annotations for the proteins

This prompts the system to contact the central data integration mediator on the ExPASy server to extract up-to-date protein annotations from UniProtKB and related non 2D-PAGE resources. Updated documents listing remote databases' locations, tissue names, etc., are imported and integrated as well.

2D data

This prompts the system to contact each registered Make2D-DB II Web server in order to establish and/or to update cross-references between the local database and the remote databases of the registered Web servers.

Data integration can be performed at different depths. Administrators choose the level with which local data is replaced by imported data (*low*, *partial* or *full* level). The lowest level replaces inner data with external one only if the former is not defined, while the highest level replaces any inner annotation if a new external annotation of the same type is available (*e.g.*, replacing protein identifiers by those extracted from UniProtKB or redefining organism classifications).

Increment entry version

Increment of the versions of protein entries depends on the *annotationChanged* flags as describe in E.V.6: Entry versions. This mechanism can be activated or deactivated from this area. Activating this option turns on the *annotationChanged* flags for any entry that has been modified due to an annotation update.

Update entry versions / Annotate and publish a database release

This area controls entry versions, database publication and metadata.

Update entry versions

At any moment, administrators can perform a version increment on all protein entries that have an *annotationChanged* flag turned on. The increment of the general and the 2D-PAGE entry versions is performed independently.

Annotate and publish a database

As described in E.V.13, database releases are intended for end-users as a milestone to indicate significant changes in a database's content. At any moment, in particular following a major database update, administrators can consolidate all recent changes, export them into the *public* schema, and publish a new database release. Depending on the importance of recent changes, a new full release or a new sub-release can be selected. The editable release details, regarding the database in its new version, are different types of metadata that are visible to the public (*e.g.*, release data and notes, contact person, etc.).

Export and backup data

Currently, databases can be exported in two formats: A flat file format, readable by humans (protein perspective) and containing the required data to rebuild the database anywhere else using the Make2D-DB II tool, and a SQL dump format, which is the common format to export or backup a relational database content. The flat file export is a protein-centred perspective based on the protein entry views (E.V.10) followed by a section listing the physical properties of the spots and their locations. In future developments, and as a means to export data to third-party systems, a generic spot perspective in XML format[1] will be proposed.

Hide/show data

These commands control which maps, which protein entries and which spots should be hidden from public. Identification analysis and results regarding a specific spot, a specific protein or a whole map can also be made invisible. All these commands access the figurative *ShowOrHideObject* interface introduced in E.V: The Gel Class.

Clean database

Performing many deletion and update operations on a relational database produces a decrease in the database performance due to records' fragmentation. Administrators can defragment their databases when query execution becomes slow. A more important fact to consider is that since any modification applied to the database content is reported into the *log* schema (E.V.13: Operation dates and history of data modifications), performing many updates inflates the size of the database in the long run. Emptying the backup clears history records and stores it in an external SQL dump file that can be restored if the need arises.

[1] The structure of the spot perspective format should be consistent with ongoing PSI recommendations.

The administration interface also gives advanced users a SQL direct access to the database content to perform SQL queries and to execute functions and commands.

G.V.3 Performance of the administration interface

To simplify management operations, we have grouped many processes in a minimal set of batch commands and functions that can be easily executed by administrators with very few interactive clicks. Collecting external sets of data, integrating them into the core system, updating the views and exporting all new modifications to the *public* schema requires an amount of time that significantly depends on many factors. Like with the main search interface, the Web traffic and the number of external resources that are contacted have a great influence on the execution time. In addition, performing the integration of collected data, generating the views and exporting all changes to public access is carried out by functions based on SQL commands, which can be a long process, depending on the size of the local dataset. The duration of an update should not be a major concern, except that many batch operations are technically performed one after another during a single instance of HTTP connection (Web based connection). Therefore, exposure to timed-out[1] connections is likely; causing an interruption of the processes that are launched on the relational server and resulting in some unachieved operations[2].

There are two possible alternatives in future developments to prevent this kind of situations if they persist. We can decompose the grouped batch operations into their components, and request the administrators to execute each component independently from the others, thus reducing the set of commands to run during one instance of HTTP connection. Unfortunately, this is done at the cost of reducing the simplicity and the intuitiveness of the update management. Currently, a decomposition of batch operations is already partially offered via the administration interface, since exporting data to the *public* schema can always be deactivated during any batch process and can be performed independently. The second alternative is to develop a non Web-based interface that creates limited and independent instances of HTTP connections to collect data, and then executes batch operations on the local relational server via the more stable TCP/IP connections. The limitation is that the non Web-based interface needs to be installed and accessed from the same computer where the Make2D-DB II Web server is running. Feedback from users will give more insight on the necessity to offer these management alternatives with the distributed Make2D-DB II package.

G.VI. Extending the Web interfaces

Developing and extending the Web server functionalities is made in concert with the implemented relational model (E.IV). Therefore, any development must remain synchronised with the model's evolution. At the same time, a high degree of abstraction regarding the data model has been reinforced in order to ensure long-term stability of

[1] A network parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time.

[2] We have experienced this situation in some occasions while interactively updating the SWISS-2DPAGE database.

data exchange operations between the distributed Make2D-DB II systems, since these systems may differ by their data models (*i.e.*, their package's versions). For example, fetching a particular object in a specific format from a Make2D-DB II system is always performed with the same logic. The result always produces the same type of output, regardless of how data is modelled and stored in the queried system.

Currently, Web interfaces do not make full use of the possibilities offered by the implemented data model. Further developments of Web interfaces should gradually integrate more objects and annotations that are already part of the data model. This includes the dealings with distinct projects within the same local database, the description of biosource materials and preparation protocols, the generation of other forms of materialised views, a more efficient use of GO terms in annotations, in data integration and in cross-referencing, etc. Simultaneously, the data analyser and converter (F.II.4) should be prepared - when necessary - to populate the relational database with additional data types extracted from text files that follow a predefined structure (plain text, XML or CSV formats) and/or from standard PSI documents, such as the PSI-MIAPE (C.IV.8) and GelML documents (Jones, Gibson 2007).

An important subproject for future development is the conception of an additional annotation interface to interactively add, remove or modify annotations in an existing database, instead of using the non-interactive <*update*> option (*cf.* F.II.5). Developing such an interface requires a significant effort because of the evolution of the relational schema and the many indirect associations between the relations (*e.g.*, associations controlled by triggers). Some basic prototypes have already been developed but they lack enough generalization to easily adapt to the constant changes of the data model. Efficient solutions should mainly centre on object-relational mapping techniques that replace direct persistence-related database accesses with high-level object handling (*e.g.*, Java Hibernate[1]) or on the Model-View-Controller (MVC)[2] abstraction approach offered by some advanced development frameworks (*e.g.*, some Smalltalk frameworks[3] or Ruby on Rails[4]). These solutions are generic enough to properly handle the complexity of the Make2D-DB II data model but require a substantial time to be implemented.

[1] http://www.hibernate.org/

[2] *cf.* Glossary. The MVC approach is already used to some extent in the main and the administration Web interfaces of Make2D-Db II.

[3] http://www.smalltalk.org/main/

[4] http://www.rubyonrails.org/

Chapter H

CHAPTER H. ACHIEVEMENTS AND TECHNICAL PERSPECTIVES

Was Make2D-DB II able to provide scientists with a functional integrative solution to manage and publish gel-based data? What are the implementations needed to make the environment more robust?

H.I. Characteristics of the Make2D-DB II environment

We have previously listed some of the main characteristics of Make2D-DB II (Table D.VI-2), and compared them with a list of representative management and integration systems. The environment is a federated 2-DE environment with a mediator support for remote data integration. Data is distributed over a network of remote independent nodes that communicate together, using simple REST protocols. The local schemas of the different versions of the tool are not necessarily identical and a federated transaction sharing approach (D.II.3) is therefore used to guarantee the federation of all nodes, the mediator being theoretically based on a local-as-view approach (LAV, *cf.* D.IV.1). Part of the non 2-DE data is warehoused, which makes the environment not purely specific to one single integration approach. Most of the distributed 2-DE data is dynamically accessed on the fly, while non 2-DE data and a small part of 2-DE data need periodic data updates to be warehoused by each node. The tool relies on a set of data exchange operations that primarily aim to ensure intercommunication between the nodes. These data exchange operations can be easily extended in order to communicate with other systems by using agreed-on data exchange structures.

At the management level, the tool ensures a high level of data consistency thanks to an object-relational structure, and to an evolving and powerful data model. Efficiency is guaranteed by an extensive use of materialised views. Data preparation, input and update are straightforward. The tool also fulfils its objective of making very simple for biologists the publication of their data on the Web.

H.II. History of the package releases

Since the beginning, the project has always followed the main defined objectives (E.II). Make2D-DB II was developed using a bottom-up approach. The core of the central model has been built originally to handle the existing SWISS-2DPAGE (and similar databases) data representation. It then expanded and became more generic with subsequent versions.

The initial versions of Make2D-DB II were made available for EBP partners (E.I.1) in 2003. At that time, the tool was limited in functionality and was only able to deal with flat files. Nevertheless, it was already possible to perform queries against distant databases. Part of the data belonging to the EBP project was collected by our group and was made available to the project partners through a dedicated server[1]. The interaction with the EBP partners significantly helped in further developments of the tool, in particular in making the tool more accessible for people with little computer science expertise.

[1] http://mordor.isb-sib.ch/ebp/

Concurrently, SWISS-2DPAGE was also converted into the new relational format and was hosted during three years in an alternative Web site. In 2006, the new installation definitively substituted the former official SWISS-2DPAGE installation running on the ExPASy server.

Public releases

Since the beginning of the public distribution of the package, we have maintained contacts with many users to assist them to install and to use the tool. The same contacts also significantly helped us to fix many problems and to improve the functionality of the tool, which was continuously extended.

The first public release was made available in October 2003. This version was a beta release that required many fixes. This version was improved twice in 2004 by two new releases. In April 2005, the first official stable version was made available and was followed, in December 2005, by a second major version with extended features. The last public release to be published was made available in September 2006. We expect to replace this release in spring 2008 by the one that has been developed during 2007 and the beginning of 2008.

Technical details regarding the evolution of the tool are given at the address:

- http://world-2dpage.expasy.org/make2ddb/changes.txt

Due to the adopted federated approach, we always had to ensure that each new version was fully compatible with all preceding versions of the tool. This was important since most users do not necessarily update their installations when a new version is available. We also had to systematically check the compatibility of the tool with all new versions of third-party software required to run the tool (PostgreSQL, Apache, Perl, InSilicoSpectro, etc.).

In order to download the tool, users are required to register by providing some personal information[1]. The registration form also asks them whether they would like to receive updates regarding new releases and fixed bugs. Table H.II-1 displays the number of public downloads from the ExPASy server since June 2003. There have been 253 downloads in total by 208 different users, using 203 distinct e-mail addresses. Downloads originated from 208 different organisations from all over the world. These organisations include a large number of universities, institutes, research centres, as well as a number of business companies.

[1] http://www.expasy.org/ch2d/make2ddb.html

Table H.II-1: Make2D-DB II public releases.

Version	Availability Date	Downloads	Remarks
0.25	June 2003	N/A	Reserved for EBP partners upon request
0.40	October 2003	23	First public release (test version)
0.89	April 2004	30	Many improvements
0.95	November 2004	30	Many improvements
1.00	April 2005	37	First official major release
2.00	December 2005	56	Second official major release
2.50.1	September 2006	77	Many improvements
2.50.2	Spring 2007	N/A	Internal version applied to SIB databases, portals and repositories
2.60.1	Scheduled for spring 2008	N/A	In development

We estimate that approximately half of the time needed to develop Make2D-DB II was spent in building up Web interactive interfaces. Since we had to find an equilibrium between the conceptual development and the implementation of interfaces, the current Web interfaces do not entirely cover all the features expressed by the implemented data model. For example, project and protocol elements are not fully presented by the web interfaces, and data integration based on Gene Ontology is not operational yet, despite the fact that these concepts are fully integrated within the physically implemented data model (E.IV). A significant amount of time was also needed to test the tool with different sets of input data, on different systems, with different versions of third-party software, and using different combinations of parameters and configurations. We tried as much as we could to reach a balance between the available resources, the time at hand, the conceptual ideas and the building up of a pragmatic and working integration system. To optimise prospective developments, many features that are not operational yet have been suitably implemented in a way that makes their future activation straightforward.

H.III. Available Make2D-DB II resources

H.III.1 Remote Make2D-DB II databases

There are currently several public and private remote databases built using our tool. Since we have been contacted by many users for technical assistance during the setting up of their databases, we are aware of the existence of a number of databases that are not part of the public domain. All databases developed in business companies belong to this category.

More importantly for the proteomics community, many 2D-PAGE resources have already joined the Make2D-DB II public environment. Most of these resources are new databases. However, part of them is formerly existent databases that have been converted into the new environment.

Almost all public databases are installed using one of the official versions of Make2D-DB II. However, one public database, 2Dbase-Ecoli (Table H.III-1) includes a feature added by the database developers, a comparison procedure between maps based on protein functional categories[1] (Vijayendran et al. 2007). This illustrates the possibility for third-party developers to easily include additional functionalities to the open source code of Make2D-DB II.

Table H.III-1: Some public 2-DE databases built with Make2D-DB II.

Database	Institution	Species (tissue)	Number of maps[2]	Identified spots / proteins	Make2D-DB II version
SWISS-2DPAGE	Swiss Institute of Bioinformatics, Geneva, Switzerland	Various (various)	36	3976 1265	2.50.2
http://www.expasy.org/swiss-2dpage/, http://www.expasy.org/ch2d/					
Rreproduction-2DPAGE	Lab of Reproductive Medicine, Nanjing Medical University, P. R. China	*Human and Mouse* (testis / ovary)	6	2605 1172	2.50.1
http://reprod.njmu.edu.cn/2d/					
CompluYeast 2D-PAGE DB	Department of Microbiology, Faculty of Pharmacy, Complutense University, Madrid, Spain	*Yeast*	16	546 169	2.50.1
http://compluyeast2dpage.dacya.ucm.es/cgi-bin/2d/2d.cgi					
CIPRO 2D-PAGE	Institute for Bioinformatics Research and Development, Japan Science and Technology Agency, Saitama, Japan	*Ciona intestinalis*	4	492 276	2.50.1
http://cipro.ibio.jp/~ueno/2d_page/cgi-bin/2d/2d.cgi					
2Dbase Ecoli	Fermentation Engineering Group, University of Bielfield, Germany	*Escherichia coli*	14	1185 723	2.01.a Modified version
http://2dbase.techfak.uni-bielefeld.de/					

[1] http://2dbase.techfak.uni-bielefeld.de/cgi-bin/2d/2d_compare_gels.cgi

[2] For February 2008

DOSAC-COBS 2D PAGE	DOSAC – COBS Proteomics and Genomics Study Group, University of Palermo, Italy	*Human* (various)	9	909 160	2.00.1
http://www.dosac.unipa.it/2d/					
Peroxisomal 2D-PAGE	Department of Cell and Molecular Biology, Upsala University, Sweden	*Mus musculus* (liver)	2	135 66	2.00.1
http://www.sbc.su.se/~jia/2D/					
Cornea 2DPAGE	Department of Molecular Biology, University of Aarhus, Denmark	*Human* (cornea)	5	268 67	1.00.a
http://www.cornea-proteomics.com/					
Plasmo 2DBase	Indian Institute of Science, Bangalore, India	*Plasmodium falciparum*	15	51 16	1.00.a
http://utlab3.biochem.iisc.ernet.in/Plasmo2Dbase/					
KAIKO 2D DataBase	The Silkworm Genome Research Program, National Institute of Agrobiological Sciences, Ibaraki, Japan	*Silkworm*	116	N/A N/A	1.00.a
http://kaiko2ddb.dna.affrc.go.jp/					

Additional resources are in a process of adhering to the public Make2D-DB II environment in 2008. In particular, the Siena-2DPAGE database from the Department of Molecular Biology (University of Siena, Italy) that has been previously published on the Web using the former Make2ddb tool (Table C.IV-4). 2-DE resources from the Proteome Research Centre at UCD Conway Institute of Biomolecular and Biomedical Research[1] (University College Dublin, Ireland) are also expected to adhere to the environment shortly.

H.III.2 World-2DPAGE Portal

In 2006, we have launched World-2DPAGE Portal, the first dynamic 2D-PAGE portal to query simultaneously worldwide gel-based proteomics databases:

- http://world-2dpage.expasy.org/portal/ [2]

This portal is simply a Web interface accessing on the fly other remote Make2D-DB II Web servers from all over the world. Therefore, it can be seen as a virtual unique database. Our group has chosen a set of available remote resources with a particular interest for the proteomics community. However, the list of resources offered is

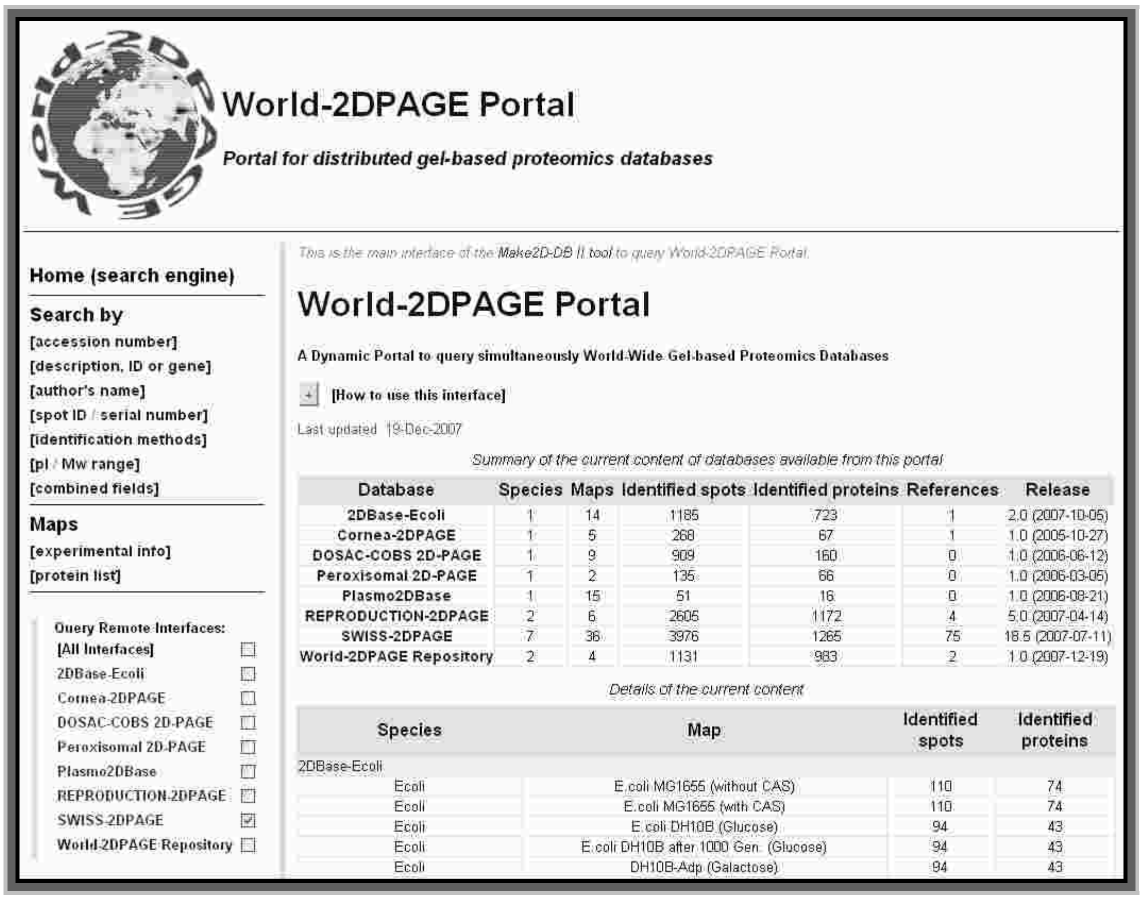
[1] http://www.ucd.ie/conway/Integrative_proteome.html

[2] Redirecting to http://www.expasy.org/world-2dpage/

intentionally limited to databases using the UniProtKB protein index or including cross-references to UniProtKB, since these databases benefit from a higher degree of data integration. In December 2007, the portal was already linked to 8 remote Web servers / 10 remote databases, totalising 91 reference maps and nearly 10300 identified spots from 10 different organisms, which makes it the largest gel-based proteomics database accessible from a single entry point. Additional 2-DE resources will soon be included within the portal.

Similarly, any organisation can easily set up analogous portals using the Make2D-DB II package. Portals are easily configurable and can include any number of remote Make2D-DB II Web servers and/or portals (F.II.6 - Web portals).

We have chosen to integrate an access to World-2DPAGE Portal within the distributed package of Make2D-DB II. Hence, end-users are able, from any remote Web server, to effortlessly select the World-2DPAGE portal for their queries.



This is the main interface of the Make2D-DB II tool to query World-2DPAGE Portal.

World-2DPAGE Portal

A Dynamic Portal to query simultaneously World-Wide Gel-based Proteomics Databases

Last updated 19-Dec-2007

Summary of the current content of databases available from this portal

Database	Species	Maps	Identified spots	Identified proteins	References	Release
2DBase-Ecoli	1	14	1185	723	1	2.0 (2007-10-05)
Cornea-2DPAGE	1	5	268	67	1	1.0 (2005-10-27)
DOSAC-COBS 2D-PAGE	1	9	909	160	0	1.0 (2006-06-12)
Peroxisomal 2D-PAGE	1	2	135	66	0	1.0 (2006-03-05)
Plasmo2DBase	1	15	51	16	0	1.0 (2006-08-21)
REPRODUCTION-2DPAGE	2	6	2605	1172	4	5.0 (2007-04-14)
SWISS-2DPAGE	7	36	3976	1265	75	18.5 (2007-07-11)
World-2DPAGE Repository	2	4	1131	983	2	1.0 (2007-12-19)

Details of the current content

Species	Map	Identified spots	Identified proteins
2DBase-Ecoli			
Ecoli	E.coli MG1655 (without CAS)	110	74
Ecoli	E.coli MG1655 (with CAS)	110	74
Ecoli	E.coli DH10B (Glucose)	94	43
Ecoli	E.coli DH10B after 1000 Gen. (Glucose)	94	43
Ecoli	DH10B-Adp (Galactose)	94	43

Figure H.III-1: World-2DPAGE Portal.

H.III.3 World-2DPAGE Repository

World-2DPAGE Repository (Hoogland et al. 2008) is the recently created supplement to World-2DPAGE Portal. Since Make2D-DB II has the ability to build and access any number of local databases, setting up a repository of 2-DE databases was an unproblematic task. World-2DPAGE Repository is a public standards-compliant repository that aims to host gel-based proteomics data with protein identifications published in the literature. It aims to support laboratories that do not have the means of hosting a database containing their data. The repository is accessible at:

- http://world-2dpage.expasy.org/repository/

Data from two publications (Plikat et al. 2007; Li et al. 2007) are already accessible. They include four 2-DE image maps with nearly 1200 identified spots. In addition, all is in place to add more datasets. Authors can easily submit their published gel-based proteomics data through a form at:

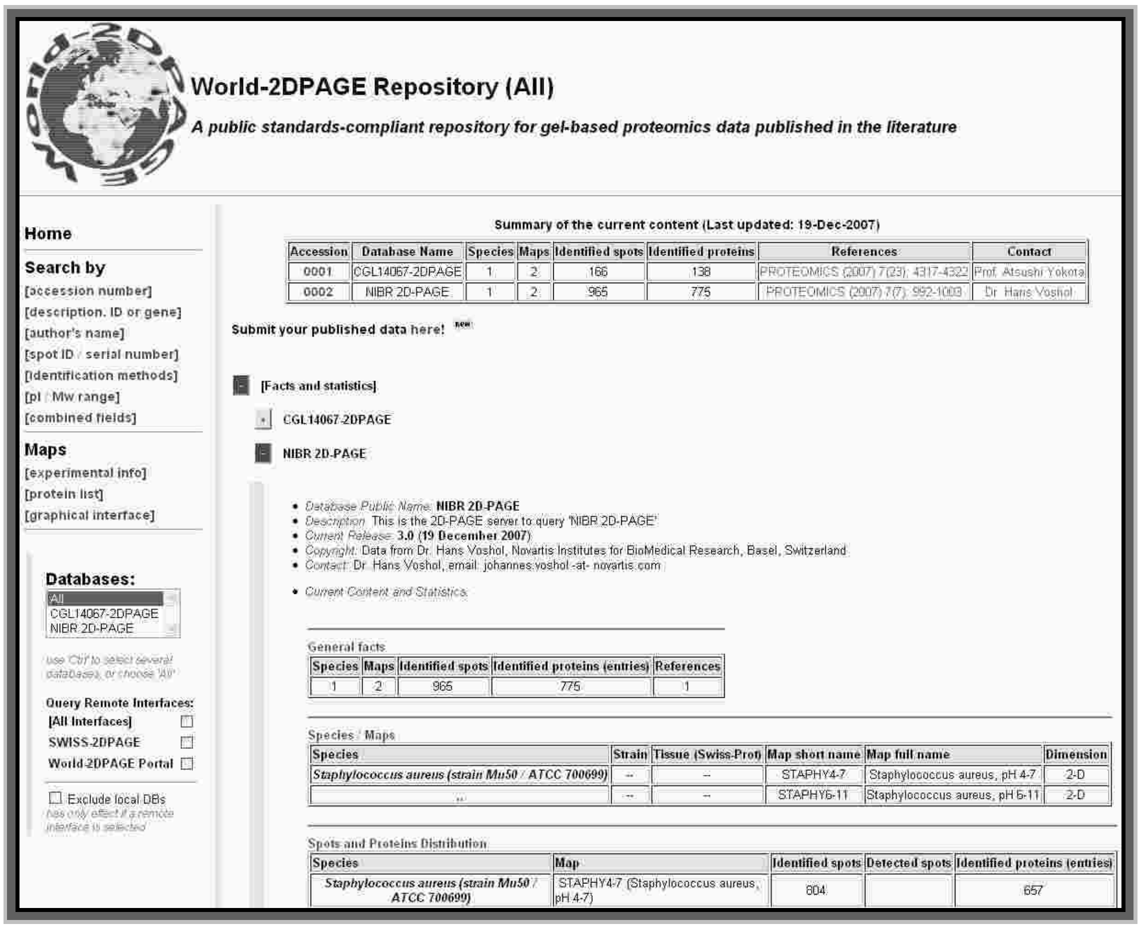
- http://world-2dpage.expasy.org/submission/

Submitters are asked to upload gel image(s), spot identification list(s), annotations, and MS data (if any). They are also asked to give relevant information on publications and experimental protocols, such as PSI-MIAPE documents (*cf.* C.IV.8 and E.IV.2 - References to external data documents). To help submitters to create PSI-MIAPE documents, our group has developed MIAPEGelDB[1], a tool that interactively generates and stores PSI-MIAPE documents through a self-documenting web interface. The more information is given, the better the annotation of the datasets will be.

In the event that a dataset was already submitted as MS identification data to the PRIDE repository (C.V.2), the same files can be reused for submission to the World-2DPAGE Repository without any additional work. Bi-directional cross-references between World-2DPAGE and PRIDE have been set up to offer a smooth navigation between both repositories. Therefore, upon manuscript submission to proteomics journals, we encourage authors to submit their MS data to PRIDE, and their gel-based data (as well as their MS data) to World-2DPAGE. Our repository supports data privacy, allowing temporary data restriction to registered users only (the submitters, their collaborators, the journal editor and the article reviewers). Data is straightforwardly promoted to public access upon authors' decision or typically upon acceptance of the corresponding article. Public datasets from the World-2DPAGE Repository become automatically part of the World-2DPAGE Portal, thanks to the Web server interconnection described in F.II.6 / Figure F.II-6.

Because of the amount of local datasets that we expect to gradually append to the new repository, we plan an imminent reorganisation of the repository Web interface (Figure H.III-2). A list of submitted databases, clustered by organisms, by tissues, by pI/Mw range, or by other characteristics, should then be presented to end-users. Thus, users can target their queries to a limited set of data resources related to their specific interests.

[1] By Xavier Robin and Christine Hoogland (Proteome Informatics Group, SIB), http://miapegeldb.expasy.org/


World-2DPAGE Repository (All)

A public standards-compliant repository for gel-based proteomics data published in the literature

Home

Search by
[accession number]
[description, ID or gene]
[author's name]
[spot ID / serial number]
[identification methods]
[pI / Mw range]
[combined fields]

Maps
[experimental info]
[protein list]
[graphical interface]

Databases:
All
CGL14067-2DPAGE
NIBR 2D-PAGE

use 'Ctrl' to select several databases, or choose 'All'

Query Remote Interfaces:
[All Interfaces] ☐
SWISS-2DPAGE ☐
World-2DPAGE Portal ☐

☐ Exclude local DBs
has only effect if a remote interface is selected

Summary of the current content (Last updated: 19-Dec-2007)

Accession	Database Name	Species	Maps	Identified spots	Identified proteins	References	Contact
0001	CGL14067-2DPAGE	1	2	166	138	PROTEOMICS (2007) 7(23): 4317-4322	Prof. Atsushi Yokota
0002	NIBR 2D-PAGE	1	2	965	775	PROTEOMICS (2007) 7(7): 992-1003	Dr. Hans Voshol

Submit your published data here! new

[Facts and statistics]

CGL14067-2DPAGE

NIBR 2D-PAGE

- *Database Public Name:* NIBR 2D-PAGE
- *Description:* This is the 2D-PAGE server to query 'NIBR 2D-PAGE'
- *Current Release:* 3.0 (19 December 2007)
- *Copyright:* Data from Dr. Hans Voshol, Novartis Institutes for BioMedical Research, Basel, Switzerland
- *Contact:* Dr. Hans Voshol, email: johannes.voshol -at- novartis.com

- *Current Content and Statistics:*

General facts

Species	Maps	Identified spots	Identified proteins (entries)	References
1	2	965	775	1

Species / Maps

Species	Strain	Tissue (Swiss-Prot)	Map short name	Map full name	Dimension
Staphylococcus aureus (strain Mu50 / ATCC 700699)	--	--	STAPHY4-7	Staphylococcus aureus, pH 4-7	2-D
..	--	--	STAPHY6-11	Staphylococcus aureus, pH 6-11	2-D

Spots and Proteins Distribution

Species	Map	Identified spots	Detected spots	Identified proteins (entries)
Staphylococcus aureus (strain Mu50 / ATCC 700699)	STAPHY4-7 (Staphylococcus aureus, pH 4-7)	804		657

Figure H.III-2: World-2DPAGE Repository.

End-users should then be presented with a list of submitted databases clustered by organisms, by tissues, or by other characteristics, so that queries can be targeted to a limited set of data resources with relation to specific interests.

H.III.4 Grouping 2-DE resources: The World-2DPAGE Constellation

The WORLD-2DPAGE Constellation has been set up to group together the many gel-based resources proposed by SIB. It is represents the home page of the recently created ExPASy 2D-PAGE domain name:

- http://world-2dpage.expasy.org/

World-2DPAGE Constellation offers a direct access to the Make2D-DB II based resources: World-2DPAGE Portal, World-2DPAGE Repository, SWISS-2DPAGE and Make2D-DB II Web server, as well as to WORLD-2DPAGE List (C.IV.4) and World-2DPAGE Repository submission form. Information and news regarding these many resources, or any other future resources, can therefore be rapidly consulted (and the resources accessed) from this insightful Web address.

H.IV. Perspectives

Make2D-DB II is still an ongoing project that can and should be extended in many different aspects. At the data model level, we have already discussed in Chapter E. the many implemented but not fully activated concepts that will need additional developments to make the environment more global and integrative (inclusion of additional external resources, use of GO terms and tissue classification to compare related data, cross-linking of similar maps between distant databases, etc). We have also presented in Chapter F. some of the limitations regarding data input, which are the unavoidable consequences of the friendly but low structured data input formats currently in use (spreadsheets / flat files). The Web interfaces and the search engines, which were presented in Chapter G. , do not presently take full advantage of the entire potential of the implemented data model. Further development of these interfaces is needed in order to handle a wider range of important features and queries that may be required by researchers (search by projects and studies, comparison of protein expression, inclusion of third-party analysis software, etc). Data updates should also be made simpler by means of an interactive interface rather than by providing text files and performing shell commands.

In the immediate future, some newly added features have to be completed and tested, and some minor bugs should be fixed, before the version currently in development (2.60) is released. The new version should be available in spring 2008. Most notably, this version will include a queriable archive system of modified protein entries (E.V.6 - Archiving modified entries) and a more automatic process to update already running installations into the most recent version of the tool (F.II.5 - The <update> option). Making the update process easier is important, since many users are not instinctively willing to reinstall their already running databases. By encouraging users to update their personal public installations, we ensure that the whole Make2D-DB II community will efficiently benefit from the most up-to-date functionalities of the tool, regardless of the origin of the distributed data.

H.IV.1 Short-term perspectives

The short-term perspectives are additional functionalities that represent important add-ons and that should not require large development efforts and resources.

Extending data integration capabilities

An imminent feature that should immediately generate a significant gain in data integration capabilities is the automatic assimilation of Gene ontology classification terms within the protein annotations (E.V.6 - Gene ontology classification). The process would mainly rely on the extracted UniProtKB cross-references to GO, as well as on the integrated UniProtKB keywords, since the mapping of the latter with GO terms is becoming increasingly reliable. Protein classification and clustering should consequently become possible over several remote databases, which is valuable in comparative studies.

Another significant development would be to extend the support for project, biosource and study annotations (E.V.2) - and for comparative studies, to extend analyte annotations (E.V.3). Such a development would require to broaden the structure of the data input files and/or to extract annotations from documents that follow the ongoing PSI recommendations covering these subjects. We may also reinforce tissue annotations by promoting a more sophisticated tissue classification to share amongst users (as described in E.V.2 - The Tissue classes).

Although the current listing of related objects (*e.g.*, related maps, studied organisms or tissues) is partially achieved using the combined search option over several remote Web servers (*e.g.*, via a Make2D-DB II portal), such developments would expand the prospect of permanently linking all related objects between remote 2-DE databases. For instance, this could be achieved by locally storing multi-directional cross-references, which implies to implement and to periodically update *ObjectDynamic* classes (*e.g.*, *OrganismDynamic* and *TissueDynmaic* classes) like the currently implemented *GelDynamic* class (E.V.8 - Remote gels).

Importing annotations

Importing directly data from PSI and MIAPE documents into the relational system should be promoted as soon as these documents become stable. Documents describing gel protocols, gel informatics and identification evidence will certainly be provided not only as supporting documents, but will also provide the relational implementation with relevant data. The gel protocol classes (E.V.3 - The Gel protocols) and the identification subsystems (E.V.5 - The predefined identification subsystems) are likely to be the first classes concerned with any upcoming document-based data extraction.

Extending data exchange formalism and formats

For the moment, the logical URL formalism and the resulting output formats used by the environment are primarily intended to exchange data between the federated remote Make2D-DB II nodes, and to extract or refer to objects in a specific format (G.III.2). The current formalism needs to be more expressive and the formats to be more generic, in order to reinforce data exchange between the environment and other integration systems. Since a generic XML format seems a practical choice, the tool must be provided with a generic procedure to export data in this structure. For better efficiency, much of the work needed to nest data should be performed at the relational database level. Object views could be generated in the same way we already pre-process the plain materialised views (E.V.10). Providing XML-based views for data exchange will not necessitate the use of Web service SOAP protocols for interoperability, since we believe that REST, combined with logical URLs, can achieve the same objective in a much simpler way.

By pre-processing data this way, we will also be able to export directly database content in XML format, along with the existing exports in extended flat file and dump formats (G.V.2), which is convenient for the distribution of databases. The XML structure reflecting the protein perspective should be intuitive to define. However, defining at this stage the XML structure from the more "natural" gel/spot perspective may lead users to confusion and undermine the PSI recommendations in 2-DE data representations, which are still in progress.

Enhancing the data update procedure

To add or modify local data within an already installed database, users are required to provide all their previous non-modified data and use the batch <*update*> option (G.VI). This approach is not user-friendly and can be improved by requiring users to provide only their new data. Modified data and data to be deleted will then need a new mechanism to be erased from the system. This mechanism should offer a way to express data to be deleted. However, expressing in a simple way which data has been modified is rather a complex task. In case not enough resources can be allocated to the development of an interactive annotation interface, dedicating some resources to set up this mechanism would be valuable for many users.

H.IV.2 Long-term perspectives

Long-term perspectives are developments that require more available resources and time. They also reflect concerns about long-term behaviour and stability of the system, and potential solutions to overcome them.

The Web-based annotation interface

We have already commented on the development of an annotation environment to interactively add, remove or modify annotations in an existing database, instead of using the non-interactive <*update*> option. We estimate that the best way is to adopt an object-relational mapping solution, or a Model-View-Controller approach, which are generic enough to cope with the complexity, the evolvement and the decentralisation of the Make2D-DB II environment (G.VI). We estimate the development time to be at least of 4 to 6 months of dedicated work by a single person. The development of such an environment will therefore depend on the resources attributed to the project in the future.

Accessing an increasing number of distributed databases

Up to now, we have not experienced any particular problem while querying simultaneously many remote Web servers. However, we cannot entirely predict the system behaviour when it will have to deal with a considerably large number of distributed resources at once, which is likely to happen, in particular with the World-2DPAGE Portal and Repository. A main concern is the common problem of timeout. We previously mentioned the possibility of clustering databases in order to reduce the queried ones to those of interest (H.III.3), which can be done by analysing and locally storing the databases' exportable statistics. Other alternatives may include the use of AJAX technology (Asynchronous JavaScript and XML)[1] to increase responsiveness and interactivity and which results in pushing data asynchronously. Similar results can be achieved by splitting the Web-based interactive process into many parallel non-interactive and independent processes that each contacts a single node and delivers the received answers in a common container that is continuously accessed by the interactive Web interface using pushovers.

[1] *cf.* glossary.

Currently, when end-users perform queries on several remote databases at once, data is presented as a list of consolidated objects, which clearly states the origin of each object. In the long run, and with a rising number of remote resources, we may make abstraction of the origin of data and concentrate on the merging of objects (Join operations), thus avoiding long and unreadable lists of objects that may contain redundancy.

Encouraging the top-down approach in annotations

Make2D-DB II has been developed using a bottom-up approach, which was essentially due to practical considerations. We wanted to deliver a working system to manage 2-DE datasets that follow a protein-based perspective, and in a relatively short period. However, we progressively managed to extend the data model to cover the wider aspects of proteomics experiments. In the present circumstances, it would be preferable to provide data following a top-down path: project – sample – separation (2-DE) – isolated entities (spots) – analysis (MS…) – identification (protein) - annotations. Make2D-DB II is not a LIMS and does not aim to reflect a proteomics experiment workflow, even if providing data this way is much more natural to deal with. This top-down path will require a different manner for data input, which should be nested data (*e.g.*, XML files). In theory, the data model can handle both approaches. However, some significant work will be needed to implement the appropriate data converters that will have to intensely check and transform the nested semi-structured content into the fully structured database relational core.

Integrating the available experimental data in SWISS-2DPAGE

Many databases published with Make2D-DB II do not provide a large set of annotations, identification evidences and preparation protocols within their contents.

Currently, SWISS-2DPAGE does not provide all of its identification evidences either. The database was converted into Make2D-DB II using the originally distributed database flat file, which cannot represent such data. To take full advantage of the possibilities offered by the new environment, we look forward to integrating all the available SWISS-2DPAGE preparation and analysis protocols, as well as its many identification evidences, within the database central implementation. Integrating all experimental identification results within SWISS-2DPAGE is important for two reasons. Given the importance of SWISS-2DPAGE as a reference resource in proteomics, this will encourage many data providers to publish their data in a similar and richly annotated manner. The second reason is that the ExPASy server might become more centred on proteomics workflows and experimental analyses, in particular after the imminent migration of UniProtKB to a self-dedicated server.

Promoting the most up-to-date version of the tool

When users install or update a database, they may be using a version of the tool that is outdated. It is possible to endorse users on-line with the most-up-to date functionalities of the tool when they are installing or updating their databases. Hence, we can think of replacing the distributed package by a live-installer that would extract the most recent components of the tool. Promoting up-to-date installations would profit to every end-user of the Make2D-DB II environment.

Chapter I

CHAPTER I. CONCLUSION

As bioinformaticians become familiar with the challenges facing data management and data integration, they realise there is no plain path that leads to a unifying solution. However, the diversity of systems, their evolvement and their cooperation will certainly contribute and converge towards the same objective: a better understanding of the complexity of the interrelated life science domains.

I.I. Discussion

Developing the Make2D-DB II environment was a very interesting task that constantly needed many considerations both at technical social levels. We were involved in many of the schemas developed in Appendix III. (A survey on the development of a proteomics data integration system).

The main constraints we encountered is that we did not start from scratch, since we had to deal with already existing semi-structured datasets. In our case, the new data model had been adapted to data and not the other way round. This was a realistic approach, given the availability of data, as opposed to a theoretical approach, which would have required data to be formulated in the logic of a theoretical model. To progress from a data-centric model towards a more generic data model, we have gone through smooth transitional steps in the model evolvement, making sure that the model always perfectly fitted with existing data and with all former versions of the tool. We frequently tolerated data incompleteness at the conceptual level, and we adopted many former views of data to ensure that end-users would not be disturbed and that former computer parsers would not suddenly break. We were aware from the beginning that a top-down approach would have been conceptually easier to handle than the bottom-up approach. But since we needed quick results, we tried to find a balance between the two approaches.

In the last few years, Make2D-DB II has established itself as a reference in data management, in data integration, and in data publication of gel-based proteomics resources. The environment has been adopted by many academic and private research groups, and it continues to serve many researchers, providing them with an easy-to-use and reliable solution. Many 2-DE datasets have become visible to the proteomics community thanks to the virtual global database set up by Make2D-DB II. Furthermore, the recent additions of both World-2DPAGE Portal and World-2DPAGE Repository are expected to significantly contribute to the expansion of the distributed integrative environment.

Still an ongoing project

However, Make2D-DB II is still an ongoing project. The tool is tied to the relational data model, which is highly structured and consistent, but implies every piece of data that it handles to be of atomic nature, such as strings and numbers. On the contrary, most data sources in biology are not that simple and are deeply nested. We overcame this problem by adopting an object-relational approach that relies on pointers to data structures and object materialisation (materialised views) at the inner level of the implementation. In a relational implementation, it is nevertheless not simple to add new data sources. Incorporating normalised proteomics data in Make2D-DB II and defining the structure of materialised objects requires a deep reflexion. On the positive side, consistency and non-redundancy are guaranteed.

Promoting open source developments

Make2D-DB II is principally dedicated to academic researchers and has not been developed to claim fees out of it. The entire code can be made open source, so that many other contributors can participate in its improvement. The involvement of other bioinformaticians and biologists in future developments, as opposed currently to a single developer, will ensure a better longevity and evolution of the system. In order to facilitate such an implication from other developers, we may consider a change in license and to set up a dedicated server to centralise and manage ideas and contributions from researchers from all over the world.

Mass spectrometry search engine

Make2D-DB II, although not formally being a mass spectrometry repository, has the ability to store mass spectrometry data. In particular, it extracts peak list values from a large range of common MS formats. Depending on the amount of MS annotations that users may provide to the tool, the distributed nodes could comprise valuable data, which represent a potential interest in spectra comparison. Therefore, we may think of implementing a matching algorithm for mass spectra within the tool. The algorithm would be triggered in order to identify the closest spectra to one or a set of provided peak list values. The main advantage would be that, in a large distributed environment, the charge of performing matching algorithms would also be distributed amid the different computers, thus reducing the need for powerful resources to deal with large quantities of locally stored data. This latent functionality would not substitute specialised MS spectra repositories. It would mostly be a supplementary means of directly linking analysed spots (or proteins) to spots on remote 2-DE resources, based on mass spectra similarity. This may help to identify, for example, non-identified proteins, or to reveal related PTM forms over a gel. To efficiently apply such an algorithm over a distributed environment would require querying all the remote databases in parallel.

I.II. What to expect next?

Data integration systems are undoubtedly crucial to the success of molecular biology research. They are the foundation blocks for the success of our aspiration to understand the many interconnected life science domains. While serving similar objectives, integration approaches are heterogeneous. The systems differ in their architectures, their purposes, and their functionalities, providing thus disparate means for data access and analysis from different perspectives.

Data integration systems will need to collaborate with each other in order to cope with the increasingly large and complex biological data. To make this possible, there is a growing need for a federated approach to share data. This approach mainly resides in sharing similar definitions of concepts and their representations. Unfortunately, common definitions and semantics amid the different research communities are hard to define. However, for a few years we have been observing many joined efforts to bring together different communities, with the intention to define the appropriate semantics. Such efforts will gradually enhance interoperability between the various data integration systems and they will be reflected by a better collaboration between the various systems.

With the promotion of data exchange standards, we may anticipate that in the near future, many data integration systems will tend to adopt a mediator and a federated approach rather than a warehouse approach. This should be supported by technical progress in network response, and by efficient algorithms to compress semi-structured/text data (not necessarily native XML, but most likely RDF, which explicitly describes data semantics). In the meantime, the warehouse approach will still play a dominant role in systems that need to collect data to generate and store data mining and analysis results.

For the time being, Make2D-DB II has been recently involved with PROTICdb and the Proteome Database System in a prospective collaboration effort aiming to pool resources in gel-based proteomics data management and integration.

BIBLIOGRAPHY

Reference List

Achard,F., Vaysseix,G., and Barillot,E. 2001. XML, bioinformatics and data integration. *Bioinformatics* **17**:115-125.

Aebersold,R. and Mann,M. 2003. Mass spectrometry-based proteomics. *Nature* **422**:198-207.

Appel,R.D., Bairoch,A., Sanchez,J.C., Vargas,J.R., Golaz,O., Pasquali,C., and Hochstrasser,D.F. 1996. Federated two-dimensional electrophoresis database: a simple means of publishing two-dimensional electrophoresis data. *Electrophoresis* **17**:540-546.

Appel,R.D., Sanchez,J.C., Bairoch,A., Golaz,O., Miu,M., Vargas,J.R., and Hochstrasser,D.F. 1993. SWISS-2DPAGE: a database of two-dimensional gel electrophoresis images. *Electrophoresis* **14**:1232-1238.

Apweiler,R., Bairoch,A., and Wu,C.H. 2004. Protein sequence databases. *Curr. Opin. Chem Biol* **8**:76-80.

Babnigg,G. and Giometti,C.S. 2004. GELBANK: a database of annotated two-dimensional gel electrophoresis patterns of biological systems with completed genomes. *Nucleic Acids Res.* **32**:D582-D585.

Babnigg,G. and Giometti,C.S. 2006. A database of unique protein sequence identifiers for proteome studies. *Proteomics.* **6**:4514-4522.

Babnigg,G. and Giometti,C.S. 2003. ProteomeWeb: a web-based interface for the display and interrogation of proteomes. *Proteomics.* **3**:584-600.

Bader,G.D., Betel,D., and Hogue,C.W. 2003. BIND: the Biomolecular Interaction Network Database. *Nucleic Acids Res.* **31**:248-250.

Bairoch,A. 2000. The ENZYME database in 2000. *Nucleic Acids Res.* **28**:304-305.

Bairoch,A. 1997. Proteome Databases. In *Proteome Research: New Frontiers in Functional Genomics*. (eds. MR Wilkins, KL Williams, RD Appel, and DF Hochstrasser), pp 93-132. Springer: Berlin, Germany.

Bairoch,A., Apweiler,R., Wu,C.H., Barker,W.C., Boeckmann,B., Ferro,S., Gasteiger,E., Huang,H., Lopez,R., Magrane,M., Martin,M.J., Natale,D.A., O'Donovan,C., Redaschi,N., and Yeh,L.S. 2005. The Universal Protein Resource (UniProt). *Nucleic Acids Res.* **33**:D154-D159.

Bairoch,A., Boeckmann,B., Ferro,S., and Gasteiger,E. 2004. Swiss-Prot: juggling between evolution and stability. *Brief. Bioinform.* **5**:39-55.

Baker,P.G., Goble,C.A., Bechhofer,S., Paton,N.W., Stevens,R., and Brass,A. 1999. An ontology for bioinformatics applications. *Bioinformatics* **15**:510-520.

Ball,C.A., Awad,I.A., Demeter,J., Gollub,J., Hebert,J.M., Hernandez-Boussard,T., Jin,H., Matese,J.C., Nitzberg,M., Wymore,F., Zachariah,Z.K., Brown,P.O., and Sherlock,G. 2005. The Stanford Microarray Database accommodates additional microarray platforms and data formats. *Nucleic Acids Res.* **33**:D580-D582.

Ball,C.A. and Brazma,A. 2006. MGED standards: work in progress. *OMICS.* **10**:138-144.

Ball,C.A., Sherlock,G., Parkinson,H., Rocca-Sera,P., Brooksbank,C., Causton,H.C., Cavalieri,D., Gaasterland,T., Hingamp,P., Holstege,F., Ringwald,M., Spellman,P., Stoeckert,C.J., Jr., Stewart,J.E., Taylor,R., Brazma,A., and Quackenbush,J. 2002. Standards for microarray data. *Science* **298**:539.

Benson,D.A., Karsch-Mizrachi,I., Lipman,D.J., Ostell,J., and Wheeler,D.L. 2006. GenBank. *Nucleic Acids Res.* **34**:D16-D20.

Berman,H., Henrick,K., Nakamura,H., and Markley,J.L. 2007. The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data. *Nucleic Acids Res.* **35**:D301-D303.

Binz,P.A., Muller,M., Walther,D., Bienvenut,W.V., Gras,R., Hoogland,C., Bouchet,G., Gasteiger,E., Fabbretti,R., Gay,S., Palagi,P., Wilkins,M.R., Rouge,V., Tonella,L., Paesano,S., Rossellat,G., Karmime,A., Bairoch,A., Sanchez,J.C., Appel,R.D., and Hochstrasser,D.F. 1999. A molecular scanner to automate proteomic research and to display proteome images. *Anal. Chem* **71**:4981-4988.

Birkland,A. and Yona,G. 2006a. BIOZON: a hub of heterogeneous biological data. *Nucleic Acids Res.* **34**:D235-D242.

Birkland,A. and Yona,G. 2006b. BIOZON: a system for unification, management and analysis of heterogeneous biological data. *BMC. Bioinformatics* **7**:70.

Birney,E. and Clamp,M. 2004. Biological database design and implementation. *Brief. Bioinform.* **5**:31-38.

Blueggel,M., Chamrad,D., and Meyer,H.E. 2004. Bioinformatics in proteomics. *Curr. Pharm. Biotechnol.* **5**:79-88.

Boeckmann,B., Blatter,M.C., Famiglietti,L., Hinz,U., Lane,L., Roechert,B., and Bairoch,A. 2005. Protein variety and functional diversity: Swiss-Prot annotation in its biological context. *C. R. Biol* **328**:882-899.

Bradshaw,R.A., Burlingame,A.L., Carr,S., and Aebersold,R. 2006. Reporting protein identification data: the next generation of guidelines. *Mol. Cell Proteomics.* **5**:787-788.

Brooksbank,C. and Quackenbush,J. 2006. Data standards: a call to action. *OMICS.* **10**:94-99.

Buehler,L.K. and Rashidi,H.H. 2005. *Bioinformatics Basics: Applications in Biological Science and Medicine*, 2nd ed. CRC Press, Taylor & Francis Group: Boca Raton, USA.

Buneman,P., Khanna,S., and Tan,W.C. 2000. Data Provenance: Some Basic Issues. *Lecture Notes in Computer Science* **1974**:87-93.

Camon,E., Magrane,M., Barrell,D., Lee,V., Dimmer,E., Maslen,J., Binns,D., Harte,N., Lopez,R., and Apweiler,R. 2004. The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Res.* **32**:D262-D266.

Chatr-aryamontri,A., Ceol,A., Palazzi,L.M., Nardelli,G., Schneider,M.V., Castagnoli,L., and Cesareni,G. 2007. MINT: the Molecular INTeraction database. *Nucleic Acids Res.* **35**:D572-D574.

Chen,I.-M.A. and Markowitz,V.M. 1995. An overview of the object protocol model (OPM) and the OPM data management tools. *Information Systems* **20**:393-418.

Chen,P. 1976. The Entity-Relationship Model--Toward a Unified View of Data. *ACM TODS* **1**:9-36.

Cheng,D. and Boudjlida,N. An Architecture for Heterogeneous Federated Mediators. 2nd INTEROP-EMOI Open Workshop on Enterprise Models and Ontologies for Interoperability - INTEROP-EMOI'05. In Proceedings of the EMOI'05 (Enterprise Modelling and Ontologies for Interoperability), in connection with the 17th Conference on Advanced Information Systems Engineering , 263-271. 2005. CAiSE'2005.
Ref Type: Conference Proceeding

Cheung,K.H., Yip,K.Y., Smith,A., Deknikker,R., Masiar,A., and Gerstein,M. 2005. YeastHub: a semantic web use case for integrating data in the life sciences domain. *Bioinformatics.* **21 Suppl 1**:i85-i96.

Chung,S.Y. and Wong,L. 1999. Kleisli: a new tool for data integration in biology. *Trends Biotechnol.* **17**:351-355.

Chung,S.Y. and Wooley,J.C. 2003. Challenges Faced in the Integration pf Biological Information. In *Bioinformatics: Managing scientific data* pp 11-34. Morgan Kaufman: San Francisco, USA.

Cochrane,G., Aldebert,P., Althorpe,N., Andersson,M., Baker,W., Baldwin,A., Bates,K., Bhattacharyya,S., Browne,P., van den,B.A., Castro,M., Duggan,K., Eberhardt,R., Faruque,N., Gamble,J., Kanz,C., Kulikova,T., Lee,C., Leinonen,R., Lin,Q., Lombard,V., Lopez,R., McHale,M., McWilliam,H., Mukherjee,G., Nardone,F., Pastor,M.P., Sobhany,S., Stoehr,P., Tzouvara,K., Vaughan,R., Wu,D., Zhu,W., and Apweiler,R. 2006. EMBL Nucleotide Sequence Database: developments in 2005. *Nucleic Acids Res.* **34**:D10-D15.

Codd,E.F. 1970. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* **13**:377-387.

Craig,R., Cortens,J.P., and Beavis,R.C. 2004. Open source system for analyzing, validating, and storing protein identification data. *J. Proteome. Res.* **3**:1234-1242.

Davidson,S.B. and et al. 2006. K2/Kleisli and GUS: Experiments in integrated access to genomic data sources. *IBM Systems Journal* **40**.

Daviss,B. 2005. Growing pains for metabolomics. *The Scientist* **19**:25-28.

Dayhoff,M.O., Eck,R.V., Chang,M.A., and Sochard,M.R. 1965. *Atlas of Protein Sequence and Structure*. National Biomedical Foundation Research Foundation: Silver Spring, MD.

Desiere,F., Deutsch,E.W., King,N.L., Nesvizhskii,A.I., Mallick,P., Eng,J., Chen,S., Eddes,J., Loevenich,S.N., and Aebersold,R. 2006. The PeptideAtlas project. *Nucleic Acids Res.* **34**:D655-D658.

Desiere,F., Deutsch,E.W., Nesvizhskii,A.I., Mallick,P., King,N.L., Eng,J.K., Aderem,A., Boyle,R., Brunner,E., Donohoe,S., Fausto,N., Hafen,E., Hood,L., Katze,M.G., Kennedy,K.A., Kregenow,F., Lee,H., Lin,B., Martin,D., Ranish,J.A., Rawlings,D.J., Samelson,L.E., Shiio,Y., Watts,J.D., Wollscheid,B., Wright,M.E., Yan,W., Yang,L., Yi,E.C., Zhang,H., and Aebersold,R. 2005. Integration with the human genome of peptide sequences obtained by high-throughput mass spectrometry. *Genome Biol.* **6**:R9.

Dowell,R.D., Jokerst,R.M., Day,A., Eddy,S.R., and Stein,L. 2001. The distributed annotation system. *BMC. Bioinformatics.* **2**:7.

Dowsey,A.W., Dunn,M.J., and Yang,G.Z. 2003. The role of bioinformatics in two-dimensional gel electrophoresis. *Proteomics.* **3**:1567-1596.

Dowsey,A.W., Dunn,M.J., and Yang,G.Z. 2004. ProteomeGRID: towards a high-throughput proteomics pipeline through opportunistic cluster image computing for two-dimensional gel electrophoresis. *Proteomics.* **4**:3800-3812.

Dowsey,A.W., English,J., Pennington,K., Cotter,D., Stuehler,K., Marcus,K., Meyer,H.E., Dunn,M.J., and Yang,G.Z. 2006. Examination of 2-DE in the Human Proteome Organisation Brain Proteome Project pilot studies with the new RAIN gel matching technique. *Proteomics.* **6**:5030-5047.

Drews,O. and Gorg,A. 2005. DynaProt 2D: an advanced proteomic database for dynamic online access to proteomes and two-dimensional electrophoresis gels. *Nucleic Acids Res.* **33**:D583-D587.

Durinck,S., Moreau,Y., Kasprzyk,A., Davis,S., De,M.B., Brazma,A., and Huber,W. 2005. BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics.* **21**:3439-3440.

Eckman,B.A. 2003. A Practitioner's Guide to Data Management and Data Integration in Bioinformatics. In *Bioinformatics: Managing scientific data* pp 35-73. Morgan Kaufman: San Francisco, USA.

Englbrecht,C.C. and Facius,A. 2005. Bioinformatics challenges in proteomics. *Comb. Chem. High Throughput. Screen.* **8**:705-715.

Eppig,J.T., Bult,C.J., Kadin,J.A., Richardson,J.E., Blake,J.A., Anagnostopoulos,A., Baldarelli,R.M., Baya,M., Beal,J.S., Bello,S.M., Boddy,W.J., Bradt,D.W., Burkart,D.L., Butler,N.E., Campbell,J., Cassell,M.A., Corbani,L.E., Cousins,S.L., Dahmen,D.J., Dene,H., Diehl,A.D., Drabkin,H.J., Frazer,K.S., Frost,P., Glass,L.H., Goldsmith,C.W., Grant,P.L., Lennon-Pierce,M., Lewis,J., Lu,I., Maltais,L.J., ndrews-Hill,M., McClellan,L., Miers,D.B., Miller,L.A., Ni,L., Ormsby,J.E., Qi,D., Reddy,T.B., Reed,D.J., Richards-Smith,B., Shaw,D.R., Sinclair,R., Smith,C.L., Szauter,P., Walker,M.B., Walton,D.O., Washburn,L.L., Witham,I.T., and Zhu,Y. 2005. The Mouse Genome Database (MGD): from genes to mice--a community resource for mouse biology. *Nucleic Acids Res.* **33**:D471-D475.

Etzold,T. and Argos,P. 1993. SRS--an indexing and retrieval tool for flat file data libraries. *Comput. Appl. Biosci.* **9**:49-57.

Etzold,T., Howard,H., and Beaulah,S. 2003. SRS: An Integration Platform for Databanks and Analysis Tools in Bioinformaitcs. In *Bioinformatics: Managing scientific data* pp 109-145. Morgan Kaufman: San Francisco, USA.

Ferry-Dumazet,H., Houel,G., Montalent,P., Moreau,L., Langella,O., Negroni,L., Vincent,D., Lalanne,C., de,D.A., Plomion,C., Zivy,M., and Joets,J. 2005. PROTICdb: a web-based application to store, track, query, and compare plant proteome data. *Proteomics.* **5**:2069-2081.

Galperin,M.Y. 2007. The Molecular Biology Database Collection: 2007 update. *Nucleic Acids Res.* **35**:D3-D4.

Garcia,C.A., Chen,Y.P., and Ragan,M.A. 2005. Information integration in molecular bioscience. *Appl. Bioinformatics.* **4**:157-173.

Garvey,T.D., Lincoln,P., Pedersen,C.J., Martin,D., and Johnson,M. 2003. BioSPICE: access to the most current computational tools for biologists. *OMICS.* **7**:411-420.

Gasteiger,E., Gattiker,A., Hoogland,C., Ivanyi,I., Appel,R.D., and Bairoch,A. 2003. ExPASy: The proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Res.* **31**:3784-3788.

Geer,R.C. and Sayers,E.W. 2003. Entrez: making use of its power. *Brief. Bioinform.* **4**:179-184.

Gianazza,E., Dossi,G., Celentano,F., and Righetti,P.G. 1983. Isoelectric focusing in immobilized pH gradients: generation and optimization of wide pH intervals with two-chamber mixers. *J. Biochem. Biophys. Methods* **8**:109-133.

Goesmann,A., Linke,B., Bartels,D., Dondrup,M., Krause,L., Neuweger,H., Oehm,S., Paczian,T., Wilke,A., and Meyer,F. 2005. BRIGEP--the BRIDGE-based genome-transcriptome-proteome browser. *Nucleic Acids Res.* **33**:W710-W716.

Goesmann,A., Linke,B., Rupp,O., Krause,L., Bartels,D., Dondrup,M., McHardy,A.C., Wilke,A., Puhler,A., and Meyer,F. 2003. Building a BRIDGE for the integration of heterogeneous data from functional genomics into a platform for systems biology. *J. Biotechnol.* **106**:157-167.

Gorg,A., Obermaier,C., Boguth,G., Harder,A., Scheibe,B., Wildgruber,R., and Weiss,W. 2000. The current state of two-dimensional electrophoresis with immobilized pH gradients. *Electrophoresis* **21**:1037-1053.

Gorg,A., Weiss,W., and Dunn,M.J. 2004. Current two-dimensional electrophoresis technology for proteomics. *Proteomics.* **4**:3665-3685.

Govorun,V.M. and Archakov,A.I. 2002. Proteomic technologies in modern biomedical science. *Biochemistry (Mosc.)* **67**:1109-1123.

Gruber,T. 1994. Toward Principles for the Design of Ontologieres Used for Knowledge Sharing. *International Journal on Human Computer Systems* **43**:907-928.

Guerrera,I.C. and Kleiner,O. 2005. Application of mass spectrometry in proteomics. *Biosci. Rep.* **25**:71-93.

Gupta,P. and Lin,E.T. Datajoiner: A Prarcitcal Approach to Multi-Database Access. In Proceedings of the International IEEE Conference on Parallel and Distributed Information Systems , 264. 1994. Los Alamitos, CA, IEEE Computer Society.
Ref Type: Conference Proceeding

Gygi,S.P., Rist,B., Gerber,S.A., Turecek,F., Gelb,M.H., and Aebersold,R. 1999. Quantitative analysis of complex protein mixtures using isotope-coded affinity tags. *Nat. Biotechnol.* **17**:994-999.

Haas,L.M. and et al. 2001. DiscoveryLink: A system for integrated access to life sciences data sources. *IBM Systems Journal* **40**.

Haas,L.M., Kossmann,D., Wimmers,E.L., and et al. Optimizing Queries Across Diverse Data Sources. In proceddings of the Conference on Very Large Databases (VLDB) , 276-285. 1997. San Francisco, Morgan Kaufmann.
Ref Type: Conference Proceeding

Halevy,A.Y. 2001. Answering queries using views: A survey. *Very Large Database J.* **10**:270-294.

Hanash,S. and Celis,J.E. 2002. The Human Proteome Organization: a mission to advance proteome knowledge. *Mol. Cell Proteomics.* **1**:413-414.

Hancock,W.S., Wu,S.L., Stanley,R.R., and Gombocz,E.A. 2002. Publishing large proteome datasets: scientific policy meets emerging technologies. *Trends Biotechnol.* **20**:S39-S44.

Herbert,B.R., Sanchez,J.C., and Bini,L. 1997. Two-Dimensional Electrophoresis: The State of the Art and Future Directions. In *Proteome Research: New Frontiers in Functional Genomics*. (eds. MR Wilkins, KL Williams, RD Appel, and DF Hochstrasser), pp 13-33. Springer: Berlin, Germany.

Hermjakob,H. 2006. The HUPO Proteomics Standards Initiative - Overcoming the Fragmentation of Proteomics Data. *Proteomics.* **6 Suppl 2**:34-38.

Hermjakob,H., Montecchi-Palazzi,L., Bader,G., Wojcik,J., Salwinski,L., Ceol,A., Moore,S., Orchard,S., Sarkans,U., von,M.C., Roechert,B., Poux,S., Jung,E., Mersch,H., Kersey,P., Lappe,M., Li,Y., Zeng,R., Rana,D., Nikolski,M., Husi,H., Brun,C., Shanker,K., Grant,S.G., Sander,C., Bork,P., Zhu,W., Pandey,A., Brazma,A., Jacq,B., Vidal,M., Sherman,D., Legrain,P., Cesareni,G., Xenarios,I., Eisenberg,D., Steipe,B., Hogue,C., and Apweiler,R. 2004. The HUPO PSI's molecular interaction format--a community standard for the representation of protein interaction data. *Nat. Biotechnol.* **22**:177-183.

Hernandez,P., Muller,M., and Appel,R.D. 2006. Automated protein identification by tandem mass spectrometry: issues and strategies. *Mass Spectrom. Rev.* **25**:235-254.

Hernandez,T. and Kambhampati,S. 2004. Integration of biological sources: current systems and challenges ahead. *ACM SIGMOD Record* **33**:51-60.

Hill,A. and Kim,H. 2003. The UAB Proteomics Database. *Bioinformatics.* **19**:2149-2151.

Hoogland,C., Baujard,V., Sanchez,J.C., Hochstrasser,D.F., and Appel,R.D. 1997. Make2ddb: a simple package to set up a two-dimensional electrophoresis database for the World Wide Web. *Electrophoresis* **18**:2755-2758.

Hoogland,C., Mostaguir,K., Appel,R.D., and Lisacek,F. 2008. The World-2DPAGE Constellation to promote and publish gel-based proteomics data through the ExPASy server. *Journal of Proteomics* **in Press**.

Hoogland,C., Mostaguir,K., Sanchez,J.C., Hochstrasser,D.F., and Appel,R.D. 2004. SWISS-2DPAGE, ten years later. *Proteomics.* **4**:2352-2356.

Hoogland,C., Sanchez,J.C., Tonella,L., Bairoch,A., Hochstrasser,D.F., and Appel,R.D. 1999. The SWISS-2DPAGE database: what has changed during the last year. *Nucleic Acids Res.* **27**:289-291.

Hubbard,T.J., Aken,B.L., Beal,K., Ballester,B., Caccamo,M., Chen,Y., Clarke,L., Coates,G., Cunningham,F., Cutts,T., Down,T., Dyer,S.C., Fitzgerald,S., Fernandez-Banet,J., Graf,S., Haider,S., Hammond,M., Herrero,J., Holland,R., Howe,K., Howe,K., Johnson,N., Kahari,A., Keefe,D., Kokocinski,F., Kulesha,E., Lawson,D., Longden,I., Melsopp,C., Megy,K., Meidl,P., Ouverdin,B., Parker,A., Prlic,A., Rice,S., Rios,D., Schuster,M., Sealy,I., Severin,J., Slater,G., Smedley,D., Spudich,G., Trevanion,S., Vilella,A., Vogel,J., White,S., Wood,M., Cox,T., Curwen,V., Durbin,R., Fernandez-Suarez,X.M., Flicek,P., Kasprzyk,A., Proctor,G., Searle,S., Smith,J., Ureta-Vidal,A., and Birney,E. 2007. Ensembl 2007. *Nucleic Acids Res.* **35**:D610-D617.

Hull,R. Managing semantic heterogeneity in databases: A theoretical perspective. Symposium on Principles of Database Systems. Proc.of the 16th ACM SIGACT SIGMOD SIGART Symp.on Principles of Database Systems , 51-61. 1997. New York, USA, ACM Press.
Ref Type: Conference Proceeding

Jones,A. and Gibson,F. 2007. An Update on Data Standards for Gel Electrophoresis. *Practical Proteomics* **7**:35-40.

Jones,A.R., Pizarro,A., Spellman,P., and Miller,M. 2006a. FuGE: Functional Genomics Experiment Object Model. *OMICS.* **10**:179-184.

Jones,P., Cote,R.G., Martens,L., Quinn,A.F., Taylor,C.F., Derache,W., Hermjakob,H., and Apweiler,R. 2006b. PRIDE: a public repository of protein and peptide identifications for the proteomics community. *Nucleic Acids Res.* **34**:D659-D663.

Kanehisa,M., Goto,S., Hattori,M., oki-Kinoshita,K.F., Itoh,M., Kawashima,S., Katayama,T., Araki,M., and Hirakawa,M. 2006. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.* **34**:D354-D357.

Karas,M. and Hillenkamp,F. 1988. Laser desorption ionization of proteins with molecular masses exceeding 10,000 daltons. *Anal. Chem* **60**:2299-2301.

Kelso,J., Visagie,J., Theiler,G., Christoffels,A., Bardien,S., Smedley,D., Otgaar,D., Greyling,G., Jongeneel,C.V., McCarthy,M.I., Hide,T., and Hide,W. 2003. eVOC: a controlled vocabulary for unifying gene expression data. *Genome Res.* **13**:1222-1230.

Kennedy,J., Hyam,R., Kukla,R., and Paterson,T. 2006. Standard data model representation for taxonomic information. *OMICS.* **10**:220-230.

Kerrien,S., am-Faruque,Y., Aranda,B., Bancarz,I., Bridge,A., Derow,C., Dimmer,E., Feuermann,M., Friedrichsen,A., Huntley,R., Kohler,C., Khadake,J., Leroy,C., Liban,A., Lieftink,C., Montecchi-Palazzi,L., Orchard,S., Risse,J., Robbe,K., Roechert,B., Thorneycroft,D., Zhang,Y., Apweiler,R., and Hermjakob,H. 2007. IntAct--open source resource for molecular interaction data. *Nucleic Acids Res.* **35**:D561-D565.

Kersey,P., Bower,L., Morris,L., Horne,A., Petryszak,R., Kanz,C., Kanapin,A., Das,U., Michoud,K., Phan,I., Gattiker,A., Kulikova,T., Faruque,N., Duggan,K., Mclaren,P., Reimholz,B., Duret,L., Penel,S., Reuter,I., and Apweiler,R. 2005. Integr8 and Genome Reviews: integrated views of complete genomes and proteomes. *Nucleic Acids Res.* **33**:D297-D302.

Kulikova,T., Akhtar,R., Aldebert,P., Althorpe,N., Andersson,M., Baldwin,A., Bates,K., Bhattacharyya,S., Bower,L., Browne,P., Castro,M., Cochrane,G., Duggan,K., Eberhardt,R., Faruque,N., Hoad,G., Kanz,C., Lee,C., Leinonen,R., Lin,Q., Lombard,V., Lopez,R., Lorenc,D., McWilliam,H., Mukherjee,G., Nardone,F., Pastor,M.P., Plaister,S., Sobhany,S., Stoehr,P., Vaughan,R., Wu,D., Zhu,W., and Apweiler,R. 2007. EMBL Nucleotide Sequence Database in 2006. *Nucleic Acids Res.* **35**:D16-D20.

Lacroix,Z. 2002. Biological data integration: wrapping data and tools. *IEEE Trans. Inf. Technol. Biomed.* **6**:123-128.

Lacroix,Z. and Critchlow,T. 2003b. Compared Evaluation of Scientific Data Management Systems. In *Bioinformatics: Managing scientific data* pp 371-391. Morgan Kaufman: San Francisco, USA.

Lacroix,Z. and Critchlow,T. 2003a. *Bioinformatics: Managing scientific data*. Morgan Kaufman: San Francisco, USA.

Lam,H.Y., Marenco,L., Clark,T., Gao,Y., Kinoshita,J., Shepherd,G., Miller,P., Wu,E., Wong,G.T., Liu,N., Crasto,C., Morse,T., Stephens,S., and Cheung,K.H. 2007. AlzPharm: integration of neurodegeneration data using RDF. *BMC. Bioinformatics.* **8 Suppl 3**:S4.

Lambert,J.P., Ethier,M., Smith,J.C., and Figeys,D. 2005. Proteomics: from gel based to gel free. *Anal. Chem.* **77**:3771-3787.

Lee,V., Camon,E., Dimmer,E., Barrell,D., and Apweiler,R. 2005. Who tangos with GOA?-Use of Gene Ontology Annotation (GOA) for biological interpretation of '-omics' data and for validation of automatic annotation tools. *In Silico. Biol.* **5**:5-8.

Lemkin,P.F. 1997. The 2DWG meta-database of two-dimensional electrophoretic gel images on the Internet. *Electrophoresis* **18**:2759-2773.

Lemkin,P.F. and Thornwall,G. 1999. Flicker image comparison of 2-D gel images for putative protein identification using the 2DWG meta-database. *Mol. Biotechnol.* **12**:159-172.

Lenzerini,M. 2001. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* pp 233-246. ACM Press: New York, USA.

Lenzerini,M., Batini,C., Giunchiglia,F., Giorgini,P., and Mecella,M. Data integration is harder than you thought (Illustration). CoopIS 2001 : cooperative information systems (Trento, 5-7 September 2001). Cooperative information systems.International conference No9, Trento , ITALIE . 2001. Berlin, Germany, Springer. 5-9-2001.
Ref Type: Conference Proceeding

Levander,F., Krogh,M., Warell,K., Garden,P., James,P., and Hakkinen,J. 2007. Automated reporting from gel-based proteomics experiments using the open source Proteios database application. *Proteomics.* **7**:668-674.

Li,F., Li,M., Xiao,Z., Zhang,P., Li,J., and Chen,Z. 2006. Construction of a nasopharyngeal carcinoma 2D/MS repository with Open Source XML database--Xindice. *BMC. Bioinformatics* **7**:13.

Li,L., Wada,M., and Yokota,A. 2007. Cytoplasmic proteome reference map for a glutamic acid-producing Corynebacterium glutamicum ATCC 14067. *Proteomics* **7**:4317-4322.

Lin,S.M., Zhu,L., Winter,A.Q., Sasinowski,M., and Kibbe,W.A. 2005. What is mzXML good for? *Expert. Rev. Proteomics.* **2**:839-845.

Lindon,J.C., Nicholson,J.K., Holmes,E., Keun,H.C., Craig,A., Pearce,J.T., Bruce,S.J., Hardy,N., Sansone,S.A., Antti,H., Jonsson,P., Daykin,C., Navarange,M., Beger,R.D., Verheij,E.R., Amberg,A., Baunsgaard,D., Cantor,G.H., Lehman-McKeeman,L., Earll,M., Wold,S., Johansson,E., Haselden,J.N., Kramer,K., Thomas,C., Lindberg,J., Schuppe-Koistinen,I., Wilson,I.D., Reily,M.D., Robertson,D.G., Senn,H., Krotzky,A., Kochhar,S., Powell,J., van der,O.F., Plumb,R., Schaefer,H., and Spraul,M. 2005. Summary recommendations for standardization and reporting of metabolic analyses. *Nat. Biotechnol.* **23**:833-838.

Lisacek,F. 2006. Web-based MS/MS Data Analysis. *Proteomics.* **6 Suppl 2**:22-32.

Lisacek,F. and Appel,R.D. 2007. Systems Biology. *Proteomics.* **7**:825-827.

Lisacek,F., Cohen-Boulakia,S., and Appel,R.D. 2006a. Proteome informatics II: bioinformatics for comparative proteomics. *Proteomics.* **6**:5445-5466.

Lisacek,F., Hoogland,C., and Bougueleret,L. 2006b. Data Integration. In *Proteome Research: Concepts, Technology and Application*. (eds. KL Williams, RD Appel, KL Williams, and DF Hochstrasser), Springer Verlag (In press): Berlin Heidelberg.

Lopez,A.J. 1998. Alternative splicing of pre-mRNA: developmental consequences and mechanisms of regulation. *Annu. Rev. Genet.* **32**:279-305.

MacBeath,G. and Schreiber,S.L. 2000. Printing proteins as microarrays for high-throughput function determination. *Science* **289**:1760-1763.

Maglott,D., Ostell,J., Pruitt,K.D., and Tatusova,T. 2005. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res.* **33**:D54-D58.

Mann,M. and Wilm,M. 1995. Electrospray mass spectrometry for protein characterization. *Trends Biochem. Sci.* **20**:219-224.

Medjahed,D., Smythers,G.W., Powell,D.A., Stephens,R.M., Lemkin,P.F., and Munroe,D.J. 2003. VIRTUAL2D: A web-accessible predictive database for proteomics analysis. *Proteomics.* **3**:129-138.

Morisawa,H., Hirota,M., and Toda,T. 2006. Development of an open source laboratory information management system for 2-D gel electrophoresis-based proteomics workflow. *BMC. Bioinformatics.* **7**:430.

Mork,P., Halevy,A., and Tarczy-Hornoch,P. 2001. A model for data integration systems of biomedical data applied to online genetic databases. *Proc. AMIA. Symp.*473-477.

Mostaguir,K., Hoogland,C., Binz,P.A., and Appel,R.D. 2003. The Make 2D-DB II package: conversion of federated two-dimensional gel electrophoresis databases into a relational format and interconnection of distributed databases. *Proteomics.* **3**:1441-1444.

Mukherjea,S. 2005. Information retrieval and knowledge discovery utilising a biomedical Semantic Web. *Brief. Bioinform.* **6**:252-262.

Mulder,N.J., Apweiler,R., Attwood,T.K., Bairoch,A., Bateman,A., Binns,D., Bork,P., Buillard,V., Cerutti,L., Copley,R., Courcelle,E., Das,U., Daugherty,L., Dibley,M., Finn,R., Fleischmann,W., Gough,J., Haft,D., Hulo,N., Hunter,S., Kahn,D., Kanapin,A., Kejariwal,A., Labarga,A., Langendijk-Genevaux,P.S., Lonsdale,D., Lopez,R., Letunic,I., Madera,M., Maslen,J., McAnulla,C., McDowall,J., Mistry,J., Mitchell,A., Nikolskaya,A.N., Orchard,S., Orengo,C., Petryszak,R., Selengut,J.D., Sigrist,C.J., Thomas,P.D., Valentin,F., Wilson,D., Wu,C.H., and Yeats,C. 2007. New developments in the InterPro database. *Nucleic Acids Res.* **35**:D224-D228.

Nilsson,C.L. and Davidsson,P. 2000. New separation tools for comprehensive studies of protein expression by mass spectrometry. *Mass Spectrom. Rev.* **19**:390-397.

O'Farrel,P.H. 1975. High Resolution Two-dimensional electrophoresis of proteins. *J Biol Chem* **250**:4007-4021.

Okubo,K., Sugawara,H., Gojobori,T., and Tateno,Y. 2006. DDBJ in preparation for overview of research activities behind data submissions. *Nucleic Acids Res.* **34**:D6-D9.

Orchard,S., Hermjakob,H., and Apweiler,R. 2003. The proteomics standards initiative. *Proteomics.* **3**:1374-1376.

Palagi,P.M., Walther,D., Quadroni,M., Catherinet,S., Burgess,J., Zimmermann-Ivol,C.G., Sanchez,J.C., Binz,P.A., Hochstrasser,D.F., and Appel,R.D. 2005. MSight: an image analysis software for liquid chromatography-mass spectrometry. *Proteomics.* **5**:2381-2384.

Paton,N.W., Goble,C.A., and Bechhofer,S. 2000. Knowledge based information integration systems. *Information and Software Technology* **42**:299-312.

Payne,W.E. and Garrels,J.I. 1997. Yeast Protein database (YPD): a database for the complete proteome of Saccharomyces cerevisiae. *Nucleic Acids Res.* **25**:57-62.

Pedrioli,P.G., Eng,J.K., Hubley,R., Vogelzang,M., Deutsch,E.W., Raught,B., Pratt,B., Nilsson,E., Angeletti,R.H., Apweiler,R., Cheung,K., Costello,C.E., Hermjakob,H., Huang,S., Julian,R.K., Kapp,E., McComb,M.E., Oliver,S.G., Omenn,G., Paton,N.W., Simpson,R., Smith,R., Taylor,C.F., Zhu,W., and Aebersold,R. 2004. A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.* **22**:1459-1466.

Perco,P., Rapberger,R., Siehs,C., Lukas,A., Oberbauer,R., Mayer,G., and Mayer,B. 2006. Transforming omics data into context: bioinformatics on genomics and proteomics raw data. *Electrophoresis* **27**:2659-2675.

Perrot,M., Guieysse-Peugeot,A.L., Massoni,A., Espagne,C., Claverol,S., Silva,R.M., Jeno,P., Santos,M., Bonneu,M., and Boucherie,H. 2007. Yeast proteome map (update 2006). *Proteomics.* **7**:1117-1120.

Pleissner,K.P., Schmelzer,P., Wehrl,W., and Jungblut,P.R. 2004. Presentation of differentially regulated proteins within a web-accessible proteome database system of microorganisms. *Proteomics.* **4**:2987-2990.

Plikat,U., Voshol,H., Dangendorf,Y., Wiedmann,B., Devay,P., Muller,D., Wirth,U., Szustakowski,J., Chirn,G.W., Inverardi,B., Puyang,X., Brown,K., Kamp,H., Hoving,S., Ruchti,A., Brendlen,N., Peterson,R., Buco,J., Oostrum,J., and Peitsch,M.C. 2007. From proteomics to systems biology of bacterial pathogens: approaches, tools, and applications. *Proteomics* **7**:992-1003.

Prince,J.T., Carlson,M.W., Wang,R., Lu,P., and Marcotte,E.M. 2004. The need for a public proteomics repository. *Nat. Biotechnol.* **22**:471-472.

Prompramote,S. and Chen,Y. ANNODA: Tool for integrating Molecular-biological Annotation Data. 21st International Conference on Data Engineering Workshops (ICDEW'05) , 1166. 5-4-2005. Washington, DC, USA, IEEE Computer Society.
Ref Type: Conference Proceeding

Pruess,M., Kersey,P., and Apweiler,R. 2005. The Integr8 project--a resource for genomic and proteomic data. *In Silico. Biol.* **5**:179-185.

Ravichandran,V., Lubell,J., Vasquez,G.B., Lemkin,P., Sriram,R.D., and Gilliland,G.L. 2004. Ongoing development of two-dimensional polyacrylamide gel electrophoresis data standards. *Electrophoresis* **25**:297-308.

Saier,M.H., Jr., Tran,C.V., and Barabote,R.D. 2006. TCDB: the Transporter Classification Database for membrane transport protein analyses and information. *Nucleic Acids Res.* **34**:D181-D186.

Sanger,F., Air,G.M., Barrell,B.G., Brown,N.L., Coulson,A.R., Fiddes,C.A., Hutchison,C.A., Slocombe,P.M., and Smith,M. 1977. Nucleotide sequence of bacteriophage phi X174 DNA. *Nature* **265**:687-695.

Schena,M., Heller,R.A., Theriault,T.P., Konrad,K., Lachenmeier,E., and Davis,R.W. 1998. Microarrays: biotechnology's discovery platform for functional genomics. *Trends Biotechnol.* **16**:301-306.

Schmuller,J. 2004. *Teach Yourself UML*. SAMS Publishing.

Schomburg,I., Chang,A., Ebeling,C., Gremse,M., Heldt,C., Huhn,G., and Schomburg,D. 2004. BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Res.* **32**:D431-D433.

Shah,S.P., Huang,Y., Xu,T., Yuen,M.M., Ling,J., and Ouellette,B.F. 2005. Atlas - a data warehouse for integrative bioinformatics. *BMC. Bioinformatics.* **6**:34.

Soldatova,L.N. and King,R.D. 2005. Are the current ontologies in biology good ontologies? *Nat. Biotechnol.* **23**:1095-1098.

Spellman,P.T., Miller,M., Stewart,J., Troup,C., Sarkans,U., Chervitz,S., Bernhart,D., Sherlock,G., Ball,C., Lepage,M., Swiatek,M., Marks,W.L., Goncalves,J., Markel,S., Iordan,D., Shojatalab,M., Pizarro,A., White,J., Hubley,R., Deutsch,E., Senger,M., Aronow,B.J., Robinson,A., Bassett,D., Stoeckert,C.J., Jr., and Brazma,A. 2002. Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biol.* **3**:RESEARCH0046.

Stanislaus,R., Chen,C., Franklin,J., Arthur,J., and Almeida,J.S. 2005. AGML Central: web based gel proteomic infrastructure. *Bioinformatics.* **21**:1754-1757.

Stevens,R., Baker,P., Bechhofer,S., Ng,G., Jacoby,A., Paton,N.W., Goble,C.A., and Brass,A. 2000. TAMBIS: transparent access to multiple bioinformatics information sources. *Bioinformatics* **16**:184-185.

Sujansky,W. 2002. Heterogeneous Database Integration in Bioinformatics. *J. Biomed. Inform.* **34**:285-298.

Suresh,S., Sujatha,M.S., Mishra,G., Hanumanthu,G.R., Suresh,M., Reddy,R., and Pandey,A. 2005. Proteomic resources: Integrating biomedical information in humans. *Gene* **364**:13-18.

Tatbul,N., Karpenko,O., and Convey,C. 2001. *Data Integration Services*. http://www.cs.brown.edu/people/koa/227papers/chapter.pdf.

Taylor,C.F., Hermjakob,H., Julian,R.K., Jr., Garavelli,J.S., Aebersold,R., and Apweiler,R. 2006. The work of the Human Proteome Organisation's Proteomics Standards Initiative (HUPO PSI). *OMICS.* **10**:145-151.

Taylor,C.F., Paton,N.W., Garwood,K.L., Kirby,P.D., Stead,D.A., Yin,Z., Deutsch,E.W., Selway,L., Walker,J., Riba-Garcia,I., Mohammed,S., Deery,M.J., Howard,J.A., Dunkley,T., Aebersold,R., Kell,D.B., Lilley,K.S., Roepstorff,P., Yates,J.R., III, Brass,A., Brown,A.J., Cash,P., Gaskell,S.J., Hubbard,S.J., and Oliver,S.G. 2003. A systematic approach to modeling, capturing, and disseminating proteomics experimental data. *Nat. Biotechnol.* **21**:247-254.

Taylor,C.F., Paton,N.W., Lilley,K.S., Binz,P.A., Julian,R.K., Jr., Jones,A.R., Zhu,W., Apweiler,R., Aebersold,R., Deutsch,E.W., Dunn,M.J., Heck,A.J., Leitner,A., Macht,M., Mann,M., Martens,L., Neubert,T.A., Patterson,S.D., Ping,P., Seymour,S.L., Souda,P., Tsugita,A., Vandekerckhove,J., Vondriska,T.M., Whitelegge,J.P., Wilkins,M.R., Xenarios,I., Yates,J.R., III, and Hermjakob,H. 2007. The minimum information about a proteomics experiment (MIAPE). *Nat. Biotechnol.* **25**:887-893.

The FlyBase Consortium 2003. The FlyBase database of the Drosophila genome projects and community literature. *Nucleic Acids Res.* **31**:172-175.

Uhlen,M., Bjorling,E., Agaton,C., Szigyarto,C.A., Amini,B., Andersen,E., Andersson,A.C., Angelidou,P., Asplund,A., Asplund,C., Berglund,L., Bergstrom,K., Brumer,H., Cerjan,D., Ekstrom,M., Elobeid,A., Eriksson,C., Fagerberg,L., Falk,R., Fall,J., Forsberg,M., Bjorklund,M.G., Gumbel,K., Halimi,A., Hallin,I., Hamsten,C., Hansson,M., Hedhammar,M., Hercules,G., Kampf,C., Larsson,K., Lindskog,M., Lodewyckx,W., Lund,J., Lundeberg,J., Magnusson,K., Malm,E., Nilsson,P., Odling,J., Oksvold,P., Olsson,I., Oster,E., Ottosson,J., Paavilainen,L., Persson,A., Rimini,R., Rockberg,J., Runeson,M., Sivertsson,A., Skollermo,A., Steen,J., Stenvall,M., Sterky,F., Stromberg,S., Sundberg,M., Tegel,H., Tourle,S., Wahlund,E., Walden,A., Wan,J., Wernerus,H., Westberg,J., Wester,K., Wrethagen,U., Xu,L.L., Hober,S., and Ponten,F. 2005. A Human Protein Atlas for Normal and Cancer Tissues Based on Antibody Proteomics. *Mol. Cell Proteomics.* **4**:1920-1932.

Ullman,J.D. Information Integration Using Logical Views. ICDT '97: 6th International Conference on Database Theory , 19-40. 1997. Heidelberg, Germany, Springer-Verlag.
Ref Type: Conference Proceeding

Ullman,J.D. and Widom,J.D. 2001. *A First Course in Database Systems*, 2nd ed. Prentice-Hall: Upper Saddle River, USA.

Unlu,M., Morgan,M.E., and Minden,J.S. 1997. Difference gel electrophoresis: a single gel method for detecting changes in protein extracts. *Electrophoresis* **18**:2071-2077.

Velculescu,V.E., Zhang,L., Vogelstein,B., and Kinzler,K.W. 1995. Serial analysis of gene expression. *Science* **270**:484-487.

Vijayendran,C., Burgemeister,S., Friehs,K., Niehaus,K., and Flaschel,E. 2007. 2DBase: 2D-PAGE database of Escherichia coli. *Biochem. Biophys. Res. Commun.* **363**:822-827.

Wang,J., Caron,C., He,A., Carpentier,A., Mistou,M.-Y., Gitton,C., Henry,C., and Guillot,A. 2005a. A system for integrative and post-planned analysis of 2-DE/MS centered proteomics data. *Journal of Integrative Bioinformatics - JIB* **0012, 2005**.

Wang,X., Gorlitsky,R., and Almeida,J.S. 2005b. From XML to RDF: how semantic web technologies will change the design of 'omic' standards. *Nat. Biotechnol.* **23**:1099-1103.

Wilke,A., Ruckert,C., Bartels,D., Dondrup,M., Goesmann,A., Huser,A.T., Kespohl,S., Linke,B., Mahne,M., McHardy,A., Puhler,A., and Meyer,F. 2003. Bioinformatics support for high-throughput proteomics. *J. Biotechnol.* **106**:147-156.

Wilkins,M.R., Appel,R.D., Van Eyk,J.E., Chung,M.C., Gorg,A., Hecker,M., Huber,L.A., Langen,H., Link,A.J., Paik,Y.K., Patterson,S.D., Pennington,S.R., Rabilloud,T., Simpson,R.J., Weiss,W., and Dunn,M.J. 2006. Guidelines for the next 10 years of proteomics. *Proteomics.* **6**:4-8.

Wilkins,M.R. and Gooley,A.A. 1997. Protein Identification in Proteome Projects. In *Proteome Research: New Frontiers in Functional Genomics*. (eds. MR Wilkins, KL Williams, RD Appel, and DF Hochstrasser), pp 35-64. Springer: Berlin, Germany.

Wilkins,M.R., Sanchez,J.C., Gooley,A.A., Appel,R.D., Humphery-Smith,I., Hochstrasser,D.F., and Williams,K.L. 1996. Progress with proteome projects: why all proteins expressed by a genome should be identified and how to do it. *Biotechnol. Genet. Eng Rev.* **13**:19-50.

Wojcik,J. and Schachter,V. 2000. Proteomic databases and software on the web. *Brief. Bioinform.* **1**:250-259.

Wong,L. 2002. Technologies for integrating biological data. *Brief. Bioinform.* **3**:389-404.

Wu,C.H., Yeh,L.S., Huang,H., Arminski,L., Castro-Alvear,J., Chen,Y., Hu,Z., Kourtesis,P., Ledley,R.S., Suzek,B.E., Vinayaka,C.R., Zhang,J., and Barker,W.C. 2003. The Protein Information Resource. *Nucleic Acids Res.* **31**:345-347.

Xirasagar,S., Gustafson,S., Merrick,B.A., Tomer,K.B., Stasiewicz,S., Chan,D.D., Yost,K.J., III, Yates,J.R., III, Sumner,S., Xiao,N., and Waters,M.D. 2004. CEBS object model for systems biology data, SysBio-OM. *Bioinformatics.* **20**:2004-2015.

Zaluzec,E.J., Gage,D.A., and Watson,J.T. 1995. Matrix-assisted laser desorption ionization mass spectrometry: applications in peptide and protein characterization. *Protein Expr. Purif.* **6**:109-123.

[No authors listed] 1999. Nomenclature committee of the international union of biochemistry and molecular biology (NC-IUBMB), Enzyme Supplement 5 (1999). *Eur. J Biochem.* **264**:610-650.

[No authors listed] 2005. Proteomics' new order. *Nature* **437**:169-170.

APPENDIX I. THE GENETIC MATERIAL

The genetic material

All information necessary to maintain and propagate life is contained within a linear array of four simple bases.

DNA and RNA

Genetic information, which is present in all organic life forms, is contained within the genetic material and is transmitted by all living organisms to their subsequent generations. This material is called the *genome* and is exclusively formed by *Deoxyribonucleic Acid*, the **DNA**; though, some viruses employ *Ribonucleic Acid*, the **RNA**, as their genetic material. DNA and RNA are macromolecules (polymers) formed by small units (monomers) called the *nucleotides*. Those are chemical structural units consisting of a heterocyclic base (a derivative of purine or pyrimidine), a pentose sugar (a deoxyribose for DNA and a ribose for RNA), and a phosphate group. There are four distinct bases in DNA, known as Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). In RNA, a Uracil (U) replaces the Thymine.

Genes are the fundamental building blocks of genetic material. They consist of a specific sequence of nucleotides encoded within the DNA - except for some viruses where genes are encoded within the RNA. In the case of DNA, two linear strands are maintained together by hydrogen bonds between opposite and complementary G-C on the one hand, and A-T on the other hand. In consequence, DNA adopts a helix conformation and forms one or several *chromosomes*. As a result, a chromosome groups the genes sequentially and ensures the good functioning of the replication process and the gene activity procedures. In opposition, RNA is only single stranded.

There are three main processes governing the maintenance and the expression of genomic information:

- The replication: Maintenance of the identity of the genetic information through a process of duplicating the DNA code. The genetic information is then maintained each time a cell divides into two child cells.

- The transcription: Copying of the information into RNA. The process involves complementarity between the bases and results in the synthesis of single stranded RNA (a sequence of A, U, G and C). Messenger RNA (mRNA) may contain coding portions that are intended to be translated into proteins.

- The translation: Generation of an "active" *amino acid* sequence (protein) based on the base sequence contained within the mRNA coding portions.

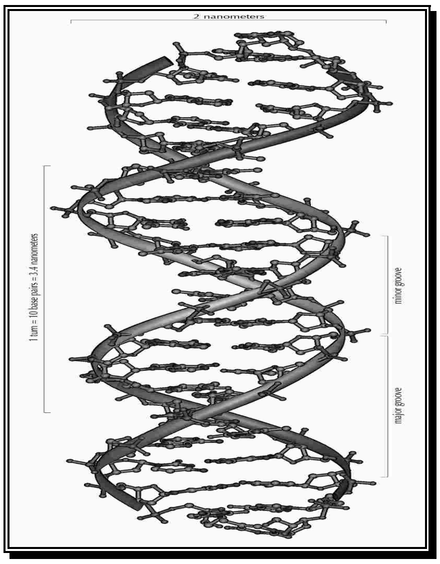


Figure Appendix - 1: Structure of part of a DNA double helix.[1]

Proteins

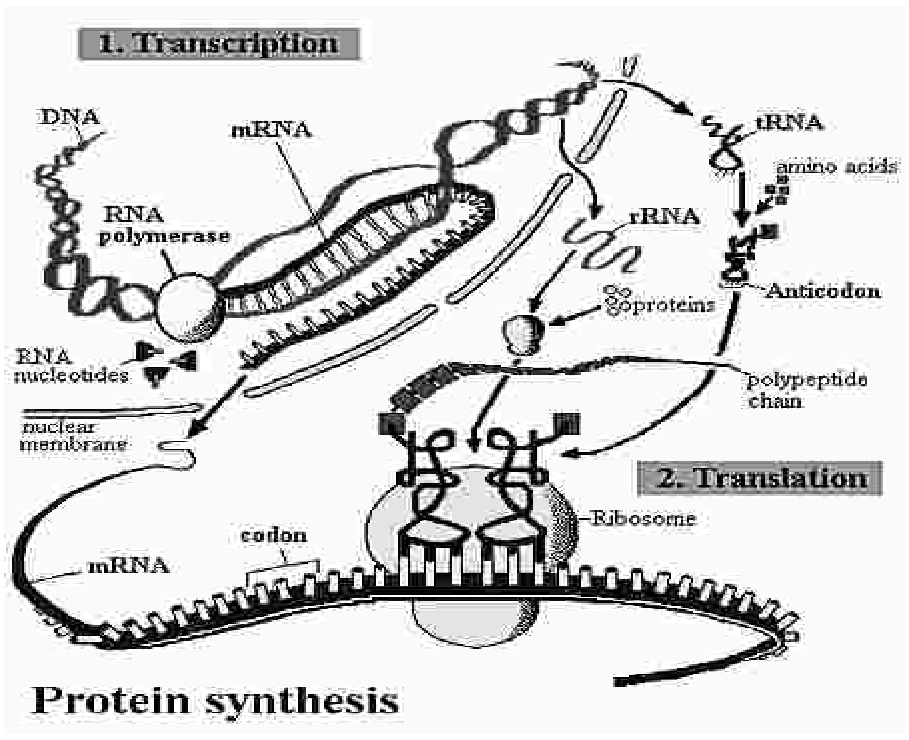
Proteins are the chemical agents responsible for almost each process occurring throughout a cell life. They play a major role in the regulation of cells' metabolism, in the interaction between cells, and are essential for the generation of specific structures. Proteins are large macromolecules made of amino acids arranged in a linear chain and adopting specific 3D structures. As already stated, two major steps separate a protein-coding gene from its protein product: first, the DNA in which the gene resides is transcribed from DNA to messenger RNA (mRNA), and then this mRNA is translated into a protein (Figure Appendix - 2).

Amino acids are organic components that are optically active compounds (L-groups) each carrying four different groups: an amino group, a carboxyl group, a proton, and a side chain (Figure Appendix - 3). There are 20 distinct amino acids in all living systems[2], differing only by their side chain composition. This implies differences in their physical and chemical properties, such as their size, their charge density distribution, their hydrophobicity, their proton affinity or electronegativity, etc (Table Appendix - 1). Joined together by peptide bonds – a bond between the carboxyl of one amino acid and the amine nitrogen of another (Figure Appendix - 4) – amino acids form the main chain of the protein. Chemical alterations of the amino acids residues within a protein, called *post-translational modifications* or **PTM** are regularly observed; they result in specific behaviour and conformation of the modified protein. More than a hundred distinct PTM are already known, *e.g.*, removing of the methionine

[1] Reproduced under the terms of the "GNU Free Documentation License".

[2] http://en.wikipedia.org/wiki/List_of_standard_amino_acids

starting signal, cleavage of other signal sequences, propeptide excision, attachment of any of a number of biochemical functional groups, such as acetate, phosphate, various lipids and carbohydrates. In addition, proteins have the ability to associate amid themselves to form stable and active complexes. Mature proteins travel to their destination and fulfil their specific function(s) until their degradation. At this point all their amino acids constituents are recycled over again in the process of new protein synthesis.



A protein is composed of amino acids. Three successive bases in the DNA strand, called a *codon*, can act as an instruction for the cell to select a specific type of amino acid. A series of codons instruct the cell to piece together a string of amino acids to form a protein. The cell knows when to begin and stop coding for proteins by recognizing special start and stop codons that are within the DNA strand. The mechanism of protein synthesis is carried out by various molecules, including transfer RNA (tRNA), messenger RNA (mRNA), and ribosomes, which are complex machineries involving dozens of different proteins associated with structural RNA (rRNA).

Figure Appendix - 2: Transcription and translation.[1]

[1] The graphical part is "Copyright 1999 Access Excellence © the National Health Museum", http://www.accessexcellence.org/RC/VL/GG/protein_synthesis.html.

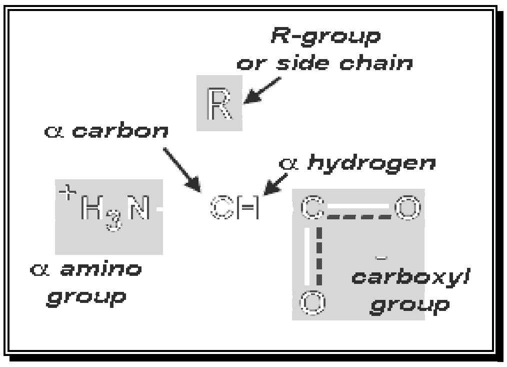


Figure Appendix - 3: Amino acid basic structure.

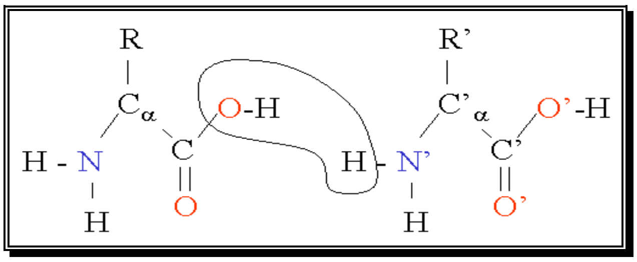


Figure Appendix - 4: Formation of a peptide bond between 2 amino acids.

Understanding the function of a protein requires knowing how the polypeptide chain folds up. The chemical reactivity of the amino acid side chains causes a specific three-dimensional structure (3D), called the tertiary (Figure Appendix - 5). While the primary structure depends exclusively on covalent bonds (peptide bonds), it plays a minor part in the formation of the tertiary structure. One important bond that is especially important for the tertiary structure is the disulfide bond (disulfide bridge), which occurs between the side chains of two cysteine residues. Ionic interactions, hydrogen bonds and Van-der-Waals attractions are also present and accountable for 3D arrangements.

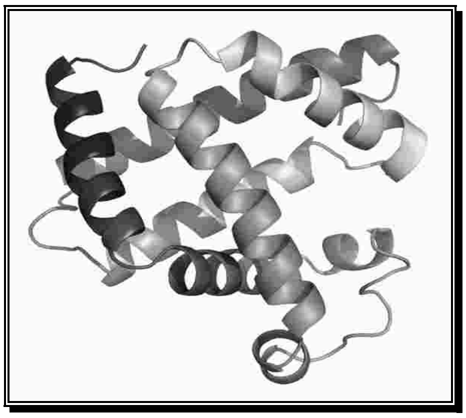


Figure Appendix - 5: 3D structure of a protein (myoglobine).

A protein varies hugely in length (from 50 up to 30000 amino acids) and may adopt different configurations, which determine its functions. It can be involved in many processes, including enzymatic activities, informative messaging, metabolic regulation, immune system (antibodies), structural material, etc. One protein may have several independent and specific functional domains. Several proteins may also interact among themselves, thus forming together a network, called a *pathway*.

Table Appendix - 1: Amino acids codes and their physical properties.

Amino Acid	3-Letter	1-Letter	Polarity	Acidity/Basicity	Hydrophob.	pI	mass (D°)
Alanine	Ala	A	nonpolar	neutral	1.8	6.01	71.09
Arginine	Arg	R	polar	strongly basic	-4.5	10.76	156.19
Asparagine	Asn	N	polar	neutral	-3.5	5.41	114.11
Aspartic acid	Asp	D	polar	acidic	-3.5	2.85	115.09
Cysteine	Cys	C	polar	neutral	2.5	5.05	103.15
Glutamic acid	Glu	E	polar	acidic	-3.5	3.15	129.12
Glutamine	Gln	Q	polar	neutral	-3.5	5.65	128.14
Glycine	Gly	G	nonpolar	neutral	-0.4	6.06	57.05
Histidine	His	H	polar	weakly basic	-3.2	7.60	137.14
Isoleucine	Ile	I	nonpolar	neutral	4.5	6.05	113.16
Leucine	Leu	L	nonpolar	neutral	3.8	6.01	113.16
Lysine	Lys	K	polar	basic	-3.9	9.60	128.17
Methionine	Met	M	nonpolar	neutral	1.9	5.74	131.19
Phenylalanine	Phe	F	nonpolar	neutral	2.8	5.49	147.18
Proline	Pro	P	nonpolar	neutral	1.6	6.30	97.12
Serine	Ser	S	polar	neutral	-0.8	5.68	87.08
Threonine	Thr	T	polar	neutral	-0.7	5.60	101.11
Tryptophan	Trp	W	nonpolar	neutral	-0.9	5.89	186.12
Tyrosine	Tyr	Y	polar	neutral	-1.3	5.64	163.18

Table Appendix - 2: The universal genetic code of translation from DNA to amino acids.

Codon	*A.Acid*	*Codon*	*A.Acid*	*Codon*	*A.Acid*	*Codon*	*A.Acid*
TTT	Phe	TCT	Ser	TAT	Tyr	TGT	Cys
TTC	Phe	TCC	Ser	TAC	Tyr	TGC	Cys
TTA	Leu	TCA	Ser	TAA	STOP	TGA	STOP
TTG	Leu	TCG	Ser	TAG	STOP	TGG	Trp
CTT	Leu	CCT	Pro	CAT	His	CGT	Arg
CTC	Leu	CCC	Pro	CAC	His	CGC	Arg
CTA	Leu	CCA	Pro	CAA	Gln	CGA	Arg
CTG	Leu	CCG	Pro	CAG	Gln	CGG	Arg
ATT	Ile	ACT	Thr	AAT	Asn	AGT	Ser
ATC	Ile	ACC	Thr	AAC	Asn	AGC	Ser
ATA	Ile	ACA	Thr	AAA	Lys	AGA	Arg
ATG	Met*	ACG	Thr	AAG	Lys	AGG	Arg
GTT	Val	GCT	Ala	GAT	Asp	GGT	Gly
GTC	Val	GCC	Ala	GAC	Asp	GGC	Gly
GTA	Val	GCA	Ala	GAA	Glu	GGA	Gly
GTG	Val	GCG	Ala	GAG	Glu	GGG	Gly

There are 64 possible combinations. Codons are read on the sense 5' to 3'. As mRNA is the template in the translation process, thymine (T) is in fact replaced by uracil (U). *At beginning of gene, "ATG" signals start of translation.

Alternative splicing

The same gene may code for several proteins differing by their amino acid sequence. The resulting proteins are named *alternative forms* or *isoforms*. The phenomenon is a precisely regulated post-transcriptional process that occurs before mRNA translation (Lopez 1998). It is only observed in eukaryotes where a gene generates a transcript of pre-messenger RNA containing sequential regions called *introns* and *exons*. The pre-messenger RNA undergoes a *splicing* process, also called "maturation", during which introns are excised (removed) while exons can either be concatenated in the mature message or targeted for removal in different combinations. The resulting sequence is called a **CDS** (CoDing Sequence). Sequence rearrangements depend on the cell type and state, the surrounding conditions and many other regulation factors. The reconnection of exons leads to various new mRNAs to be translated into different protein isoforms. This process cancels the old theory of "one gene one protein". It is of great importance in species evolution, as it raises dramatically the efficiency and the flexibility of the encoded information.

APPENDIX II. MASS SPECTROMETRY

Mass spectrometers can be divided into three fundamental parts, namely an ion **source**, an ion **analyser**, and an ion **detector**. ESI and MALDI techniques are soft ionisation methods that produce little fragmentation of the ionised peptides. ESI relies on the direct ionisation of the peptides from solution. It can therefore be interfaced with liquid separation methods. ESI produces a spraying of an electrically generated fine steam of ions directed into the inlet of the mass spectrometer. In the MALDI technique, the sample is mixed with a special matrix acting as a proton donor. The matrix interacts with the peptides and forms with them crystalloid structures. Those structures are then exposed at their surface to UV laser pulses, giving them enough energy for evaporation. The peptides, successively detached from the matrix, are ionised before reaching the inlet to the spectrometer. The use of laser in MALDI is responsible of the generation of packets of ions, rather than a steady stream, which shall be "trapped" on the analyser part of the spectrometer. Contrary to ESI, MALDI ionisation generates only singly charged ions. Traditionally, MALDI was used to analyse 2-DE separated proteins in combination with time-of-fly mass analysers (TOF). A small quantity of material is consumed, which is adapted to the quantity of material that may be contained within a spot, especially when repeated analyses are needed. The produced spectra are also easier to interpret, due to the exclusively singly charged ions produced at the source level. Alternatively, ESI is well adapted for liquid-phase solutions and offers a better option for high masses exceeding the range of MS sensibilities, given the fact that the mass-to-charge ratio (m/Z) is lessened when $Z > 1$.

When the ions reach the analyser part of the spectrometer, they are separated according to their mass-to-charge ratio (m/Z). There are several types of analysers in use in proteomics mass spectrometry, but the most common are:

- Ion traps (IT): An ion trap is a combination of electric or magnetic fields that captures ions in a device, a tube, under vacuum. Ions are "trapped" in a time varying electric field, and oscillate at frequencies related to their mass-to-charge ratio. By varying the field parameters, ions are excited and ejected sequentially from the opposite tube end.

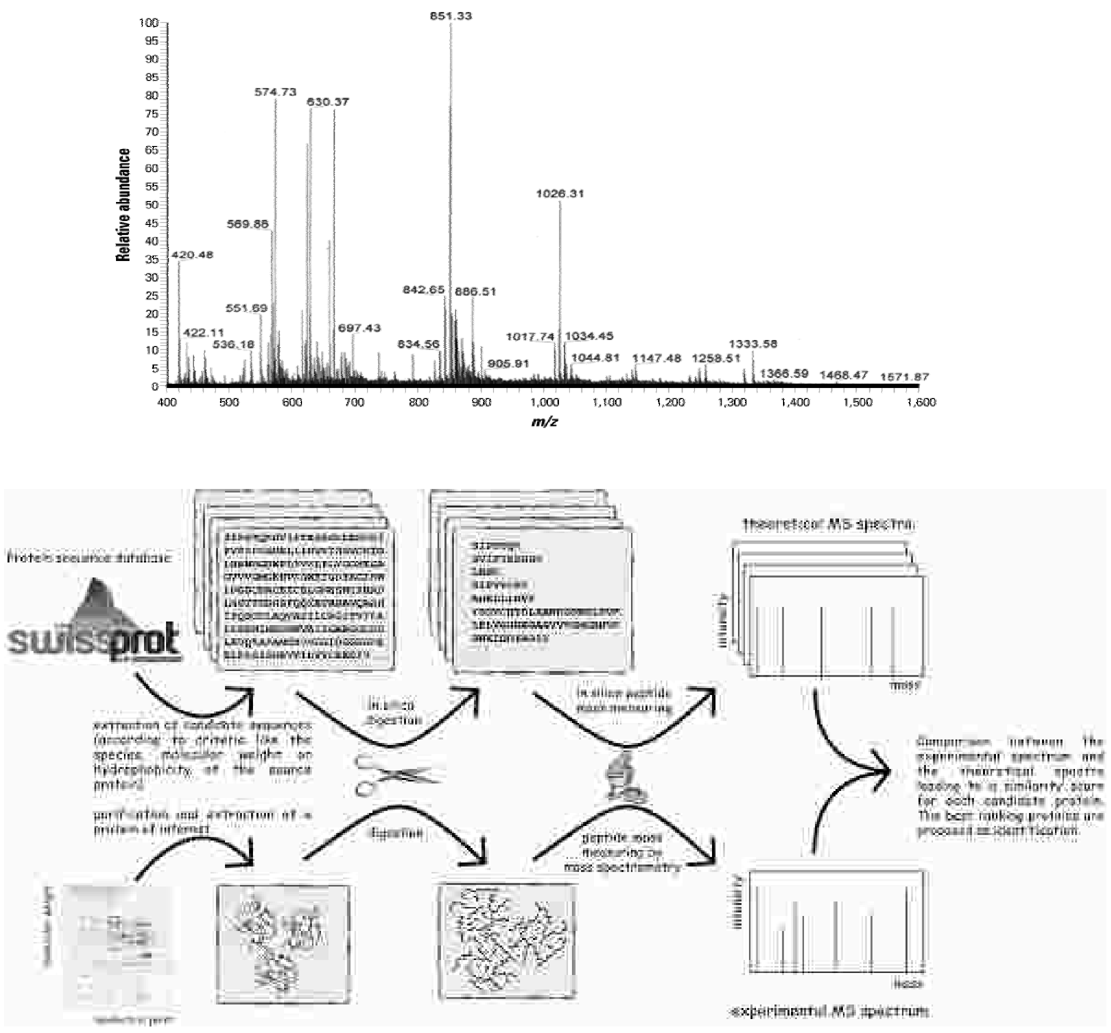
- Time-of-flight (TOF): In this analyser, the particles undergo a high positive voltage pulsed with a certain rate. While fragments with no charge pass their way with no deviation, the field repels the similarly charged fragments and sends them down the perpendicular TOF tube to the detector at its end. Ions react proportionally to their mass-to-charge ratio, with the lighter ones travelling the TOF tube faster than the heavier ones. The time spent by the different fragments is recorded, and quantification of the ratio is rendered.

- Quadrupole (Q) and Quadrupole ion trap (QIT): Quadrupole analysers consist of four parallel metal bars with opposing pairs electrically connected together. A radio frequency voltage is applied between one pair of bars, and an opposite voltage is applied on the other pair. An additional current voltage is superimposed on ions travelling between the quadrupoles. For a given ratio of voltages, only ions with a specific m/Z ratio will reach the detector, while other ions adopt an unstable trajectory and end up colliding with the bars. Variation of the voltages allows the scanning for different m/Z values. Quadrupole ion traps are a variation of quadrupoles operating with an additional ion trap.

- Fourrier transform ion cyclotron resonance (FT-ICR): Based on the cyclotron frequency of the ions in a fixed magnetic field. The ions rotate around a magnetic field with a frequency specific to their m/Z. Variation of the applied field results in changes to the frequency of rotation, which are measured and converted using Fourrier transformation.

TOF and FT-ICR are widely used in whole protein analysis. For digested peptides, QIT and TOF (generally combined with a MALDI source: MALDI-TOF) are the most commonly used instruments.

The final element of the mass spectrometer is the detector. It records the charge induced or the current produced when an ion crosses it or hits its surface. The detection produces a **mass spectrum**, a record of peptides intensity as a function of m/Z. This spectrum may provide sufficient information to search a sequence database using just several values of the revealed masses: this is called **Peptide Mass Fingerprinting**, or **PMF** (Figure Appendix - 6). Some additional features, like knowledge of potential post-modifications, are sometimes necessary. If the database search is not fruitful or is not deterministic (non catalogued or uncharacterised protein, inaccurate data, difficulty to distinguish between several entries in the database), then further information is required. A Tandem MS study offers the possibility to determine the amino acid sequences of individual peptides contained in the digest mixture. Further database searching can then be carried out.



Relative abundance
m/z
851.33
574.73
630.37
1026.31
569.88
420.48
842.65
886.51
551.69
422.11
536.18
697.43
834.56
1017.74
1034.45
1333.58
905.91
1044.81
1147.48
1258.51
1366.59
1468.47
1571.87
protein sequence database
swissprot
theoretical MS spectra
experimental MS spectrum
in silico digestion
in silico peptide mass measuring
digestion
peptide mass measuring by mass spectrometry
extraction of candidate sequences (according to criteria like the species, molecular weight or hydrophobicity of the source protein)
purification and extraction of a protein of interest
Comparison between the experimental spectrum and the theoretical spectra leading to a similarity score for each candidate protein. The best ranking proteins are proposed as identification.
intensity
mass

Figure Appendix - 6: PMF, an experimental spectrum, and an identification workflow at the Geneva University Hospital[1].

Candidate protein sequences extracted from a database are digested in silico according to protease specificity. Theoretical MS spectra are constructed and compared to the experimental MS spectrum, leading to a similarity score for each candidate protein. The candidate proteins are then sorted according to their score. The top-ranked protein is considered as the identification of the spectrum.

Tandem mass spectrometry (tandem MS or MS/MS) consists of several steps of mass selection, fragmentation and analysis. A tandem mass spectrometer begins by achieving a first MS analysis step where it measures peptides according to their m/Z ratio. Specific ions are then selected and fragmented sequentially (parent or precursor peptides). Resulting fragments are then separated and analysed in a second step. This

[1] Courtesy P. Hernandez, SIB / Geneva.

can be achieved by connecting in series different analysers ("in space" tandem MS, *e.g.*, Q-TOF, or TQ / triple quadrupole), or by performing consecutively the two steps within the same analyser ("in time" tandem MS, *e.g.*, QIT). For example, in a "in space" tandem MS, we may have one mass analyser to isolate one specific peptide entering the spectrometer. A second analyser shall stabilise the peptide ions while they collide with a gas that causes them to fragment by collision-induced dissociation (CID). Finally, a third analyser shall collect those induced fragments.

Spectra obtained by tandem mass spectrometry contain series of peaks that originate from different fragmentation positions in the precursor peptide sequence (Figure Appendix - 7). Tandem MS spectra are a raw signal that is primary processed into a generic peak list. Processed spectra include the parent peptide, as well as a list of peaks corresponding to the diverse fragments that have been produced within the spectrometer. This is a key property of MS/MS spectra, since information about the peptide sequence can be deduced from the mass differences between peaks (*e.g.*, De novo sequencing, peptide fragment fingerprinting) (Mann, Wilm 1995).

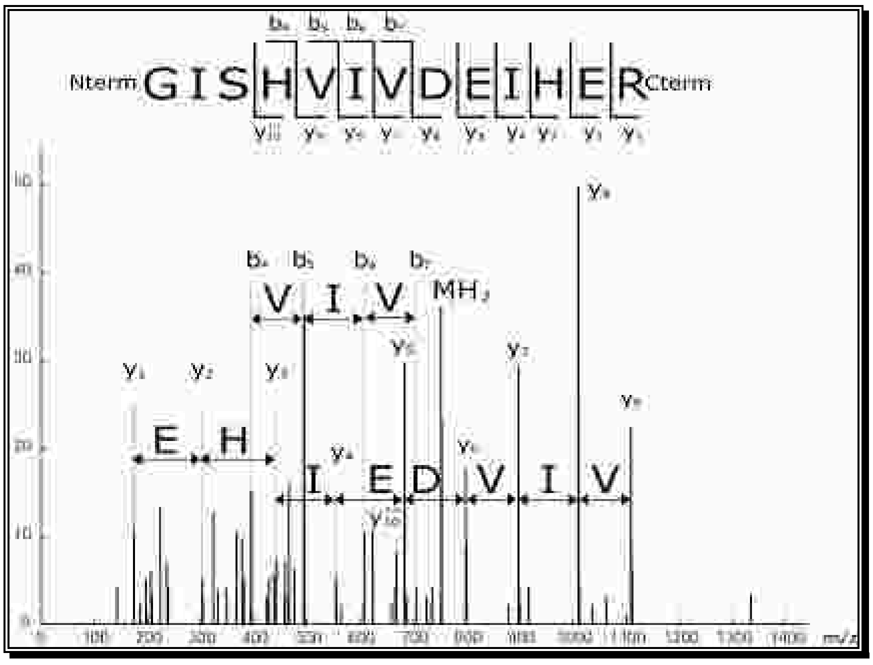


Figure Appendix - 7: An annotated MS/MS peak list spectrum.[1]

Based on knowledge of amino acids masses and their most common modifications, information about the peptide sequence can be deduced from peak differences. Non-annotated peaks may originate from unconsidered ion types.

[1] Courtesy P. Hernandez, SIB / Geneva.

In all these approaches, proper handling of mass spectrometric data is critical for the quality of the final interpretations. Peptide and/or fragment signals have to be extracted from the original mass spectra with as little noise as possible. Masses are then compared with their theoretical values calculated from protein or genomic sequence databases. Appropriate scoring functions are applied to evaluate the accuracy of the matches. Let us notice though that interpreting a mass spectrum is not a linear or a straightforward process. Understanding of the used technology, the potential type of ionisation and fragmentations undergone by the peptides, the likely chemical alterations of the proteins, and many various factors is crucial for a high quality interpretation. Many free and commercial tools dedicated to MS identification, differing in their approaches and algorithms, are available for researchers (Blueggel et al. 2004). As multiple MS/MS identification algorithms are available or have been theoretically described, difficulty resides in choosing the most adapted method for each type of spectra being identified. Finding the right tools among the many possibilities offered may be a challenge (Hernandez et al. 2006; Lisacek 2006). Besides, with high-throughput identification data (*e.g.*, with LC-MS/MS), looking at a single spectrum at a time may undermine a treasure of information, *e.g.*, differential expression. This type of additional information may be grabbed by considering the entire set of spectra all together; an approach is carried out using image analysis (Palagi et al. 2005).

APPENDIX III. A SURVEY ON THE DEVELOPMENT OF A PROTEOMICS DATA INTEGRATION SYSTEM

Developing a data integration system involves generally a methodology based on the following stages (Lacroix, Critchlow 2003a):

- Collection of specifications and requirements
- Conversion of the specifications into a technical representation (designing)
- Development process
- Deployment, evaluation and reconsideration of the system

Although these are indeed the key stages, they should be considered as a guideline, and not strictly as a fixed plan to follow. No stage has to be strictly complete before the next one starts (Schmuller 2004). Throughout the different stages, the project manager may often feel the need to reconsider some preceding phases for many reasons, among which the evolution of requirements, real-world technical inconveniences or new and unsuspected ideas that would benefit to the system.

The specification and requirements

The very first phase when designing a data integration system consists in collecting a set of requirements that have to be definite and unambiguous. First, it is essential to acquire a reasonable knowledge and a good understanding of the domain of concern. It is also imperative to know exactly who the users of the system are. What may they expect from such a system, or what they can be offered to assist them in their work? What kind of functionalities the system will put forward and how they will be presented? It is also important to investigate on the expected technical performance of the system, as well as on the available human and financial resources that will cover up its implementation.

User Profile

First, it is important to define what we might call the “user profile”. By “user profile”, we mean the description of who are the target users as well as the knowledge and experience they are supposed to have. It is very important to precisely specify the level of “computer literacy” target users are expected to have, as it will be determining in the choice of the methods thanks to which they are going to interact with the system. Having defined the user profile, the various tasks to be performed by the system are to be listed and analysed. Those consist of what is commonly called “use cases”: How and for what purpose will a user effectively utilise the system? What are the available data sources that should (and can) be integrated? How users will formulate their queries and what will be the extent of these queries? What is the format of the output? And so on. One particular challenge for bioinforamticiens in answering those questions remains the speed at which biological knowledge is evolving and the ever-changing requirements that accompany this evolution. This is during this same phase that early outlines of the data model are to be conceptualised.

Technical requirements

Technical considerations are one factor that will determine pure practical aspects of the system: the platform(s) the system will work on, the storage capacity, the efficiency with which the system will handle communication and integration overheads, etc.

Operational constraints

Operational constraints depend on both the financial and personal resources available for the project. An interesting survey on the subject is given by Birney (Birney, Clamp 2004).

Converting the specifications into a technical representation

At this point, we have to cope with the system optimisation taking into account the hardware constraints for data management when adopting the predefined specifications. Here, we also have to take in consideration the maximisation of the system's efficiency and the minimisation of its resources' cost. As an example, regarding a storage costs, we may decide which integrated items are to be materialised (using a warehouse approach) and which ones are to be only collected at query execution time (using a mediator approach). Those choices deeply depend on the purpose for which the system is intended. For example, a warehouse strategy is a more appropriate choice for a curation system in which some integrated components may be locally modified or re-annotated, while for integrated components that should be strictly up-to-date the mediator approach is more suitable.

The development process

This part can vary enormously in time, depending on the used techniques and the experience skills of the developers. It includes repeated loops constructing and testing the code. User interfaces are also built and connected to the code to test their functionality. In parallel, the technical documentation is written as well as the user manuals.

Deployment and Evaluation

A working system is then deployed on the appropriate hardware and integrated with the cooperative systems. Backup and recovery strategies are also set up. The system is evaluated in order to check if it fully performs as it is supposed to regarding its specifications. The performance is estimated in time (pre-processing time, query response time, etc.) and space (used memory, caching, etc.) costs, a task that may often imply the use of a set of benchmarks. User survey and feedback are also necessary to estimate users' satisfaction. Evaluation may often lead to augment or adjust the initial specifications.

The evaluation criteria

Criteria to evaluate a running integration system may be regarded from both the implementation (computer science) and the user (biological) perspectives (Lacroix, Critchlow 2003b). The criteria covers a wide range of issues and may be applied with a certain degree of flexibility, as their definitions may sometimes overlap or be slightly formless. Tradeoffs should be taken to balance the needs of the target users:

The implementation perspective

It consists in appreciating the system from a technical point of view driven by user requirements. For this perspective measurement, cost models are typically taken into consideration.

- Efficiency: A combination of query efficiency (ability to respond to user queries), data storage size, communication overhead (data transfer, frequency of commands executed remotely, potential timeouts) and integration overhead (complexity of the transformation performed on source data) define the overall efficiency of the system. Efficiency is evaluated for the pre-processing step and for the response to a query.

- Extensibility: This is a measurement of the efforts needed to extend or increase the system functionalities. The bigger the proportion of data materialisation, the higher the cost of extension will be.

- Functionality: This reflects the number, the type and the complexity of the queries the system can perform on the data.

- Scalability: The amount of data, number of users and number of data sources the system can handle simultaneously.

- Understandability: This is the clarity of the system design. It directly influences the time required by a developer to add or modify components, especially developers not involved in the original design.

- Usability: This evaluates the easiness with which a user becomes familiarised with the system functionalities. It also includes the ability of some user to modify the system behaviour and capabilities.

The user perspective

It is the users' points of view on the same aspects of the system. It reflects the overall satisfaction of the users regarding the system.

- Efficiency: Evaluated by the ability of the system to perform queries in a reasonable time.

- Extensibility: Characterises the efforts needed by a user to customise or add new types of queries or data, to extend the resources and to modify some parameters. A low requirement in programming skills often denotes a good scale of extensibility.

- Functionality: This, of course, reflects the different types of queries offered by the system. But it also reflects how those queries can be combined to form new ones, to answer more complex questions, and how answers can be sent to other third-party tools (*i.e.*, sending the output as an input for other programs). Generic systems and systems allowing the formulation of a diversity of queries are well rated.

- Scalability: In addition to the same attributes as those described in the implementation perspective, biological researchers are more and more concerned with the ability to perform batch queries (several sequential queries, as opposed to a single query at a time). Whenever a researcher needs to precede many queries in a large-scale process this criteria becomes really vital.

- Understandability: Users need to fully understand the meaning of the queries they are performing and what they exactly do (*e.g.*, for a keyword search query, what part of the data is being searched). All the semantics used by the interface and within the presented data should be clearly defined and unambiguous. This point has to be considered with special care for systems providing global views originating from various integrated sources, as the latter may have semantics that differs amongst them.

- Usability: Proposing intuitive access to the system enhances usability. Interactive and visual interfaces are well adapted when users are not familiar with the system. Developers may conceive additional parallel alternatives for more experimented users, (*e.g.*, command line interface, shortcuts, etc.). In the majority of situations, a good usable system should never assume or require its users to be or act as “programmers”.

Tradeoffs are unavoidable. They highly depend on the original requirements. Some of the previous criteria may be mutually exclusive. The inclusion of many resources (scalability) and semantic consistency (usability) are obviously contradictory. A system that necessitates presenting the most up-to-date data will probably opt for a non-materialised mediator approach, raising by the same occasion its extensibility. In fact, it may suffer a decrease in efficiency because of the extra time needed to contact non-local resources. At the same time, as the heterogeneity and the number of remote resources considerably affect the whole performance – a very common issue in biology areas – many bioinformatics systems tend to be more domain-specific rather than widely generic.

Ultimately, one should be aware that the overall evaluation is typically a subjective issue that tightly depends on specific users’ requirements and that it can profoundly vary between researchers.

APPENDIX IV. UML

The Unified Modelling Language

Definition

In system development projects, the Unified Modelling Language (**UML**) is a visual tool that bridges the gap between the vision of a system and its implementation. It helps capturing the vision of a system and then enables to communicate this vision to someone else. Many available works help conferring an overview of the basic elements of the language, e.g. Schmuller's "Teach Yourself UML" (Schmuller 2004). For an exhaustive documentation on the subject, one may refer to the "Unified Modeling Language Resource Page"[1], maintained by the Object Management Group and the UML consortium, and where UML specifications, tutorial and tools are supplied.

Readers of the present document should not consider our use of this language as an inflexible and one rigid manner to portray our models. In our opinion, there should never be any obligation to strictly stick to a specific notation. Our goal is to transmit the ideas and designs we adopted. Any agreed on conventions that can completely and unambiguously describe a specific system and that are able to communicate it are perfectly suitable. Actually, we will use some of the basic elements from the current UML 2.0 specifications, and we will notify the reader whenever any altered or hybrid annotation is being used.

We should distinguish between a **model** annotated with UML and the true implementation of a system. Indeed, the model shows what the system is supposed to do and how it should behave, but it does not tell how to implement the system. UML annotations consist of a number of graphical elements that combines by using definite rules to form **diagrams**. Thus, a diagram is a partial graphical representation of the system's model. Actually, many distinct diagrams are suitable for various representations. Each of them is in fact a specific point of view about the system in hand, but, as in real life, many points of views may overlap and be complementary, diagrams are permitted as well to mix and to become hybridised. We will list here some diagrams that we will be using - separately or mixed - in our document. However, we will first define some familiar object-orientation notions that are adopted in UML.

[1] http://www.uml.org/

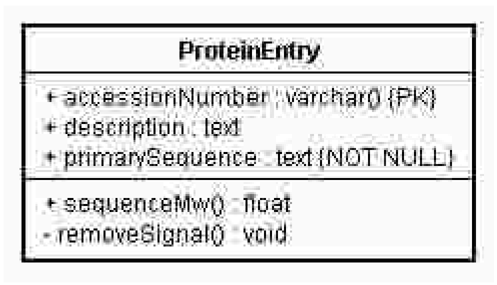
Many of these notions are also directly applicable to the **class diagram** representation, a diagram that we will broadly use throughout this document.

Object-orientation concepts and related UML elements

Objects and Classes

The UML allows designers to build easy-to-use and easy-to-understand models of **objects**. An object is an instance of a **class**. The latter is a category or a group of "things" that have the same **attributes** (properties) and **operations** or **methods** (behaviours). For example, *ProteinEntry* may be a class representing protein entries. All proteins falling in this category share the same type of attributes; they all have an accession number, a description text, some related genes, a primary sequence, etc. An operation may be attached to the *Entry* class to compute, for example, the molecular weight of the protein's primary sequence, and another operation to remove any starting signal from the provided sequence. By convention, class names are written with their first character uppercase, while attributes and operations – which are called features of the class - are not. Operations are assimilated to functions and their names are thus followed by a pair of parenthesis. Generally, all the names are written in a multiword that runs all the words together, with each non-initial word beginning with an uppercase letter. Examples are *ProteinEntry* for a class, *proteinDescription* for an attribute and *removeSignal()* for an operation.

The UML class is represented by a rectangle divided into 3 parts, the top one containing the class name, the two others bellow listing the class features. Features' names are preceded by a '+' sign (public, visible everywhere), by a '-' sign (private, only visible to the class itslef) or by a '#' sign (protected, only visible to the class itself and any derived class).



Note that, as we have chosen to present our relational model implementation using a class diagram, we have adopted many relational systems' related terms in our attributes' definitions. At the same time, classes will themselves be essentially assimilated to the physically implemented **relations** (the relational tables).

An attribute name is followed by a colon followed itself by the attribute's data type. In our case, data types are:

Type	**Description**
char(n)	A fixed-length string of n characters
varchar(n)	Character varying. When n is given, this means the string may have any length between 1 and n
text	A string attribute of any length. More commonly known as a string type
int	Integer, small-range integer or large-range integer may be précised by short and long
serial	Auto-incrementing integer
float	Numeric type with user-specified precision (if given between parenthesis)
date or timestamp	Date or date and time type
boolean	A boolean type with 3 possible states (true, false or unknown)
type[ ][ ]..[ ] or ArrayList	A data type may be defined as variable-length multidimensional array. The dimension is given by the number of brackets' pairs following the data type

Conditions on the attributes are given between braces "{*condition*}". We choose to use for our conditions any of:

- {PK}: The same sense as the 'Primary Key' in relational representation. Instantiation of the attribute must be unique and defined.

- {FK}: The relational 'Foreign Key' definition, which means the attribute references another class(es)' primary key(s).

- {Unique}: Instantiation must be unique but not necessarily defined.

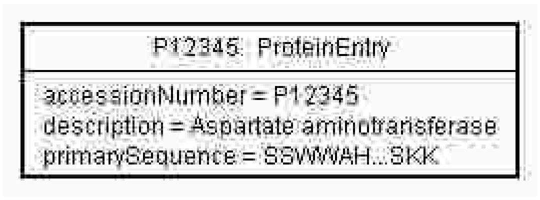
- {Not Null}: Instantiation must be defined but not necessarily unique

- {Check*: boolean condition*}: More specific conditions, *e.g.*, { Check: attribute = 2 or attribute = 3 or attribute = *someFunction*() }.

An attribute may also have a default value, which is given after an equal "=" sign, *e.g.*, "*description : text = my default description text*".

Operations, being assimilated to functions, return a value (which may be a void value). The type of the returned values is directly given after the operation's name. Whenever the operation name is written in *italic* in the operation compartment, this indicates an abstract operation, which is an operation defined, but not implemented by an abstract superclass. The operation must be implemented by all concrete descendant classes.

We have already defined an object to be an instance of a class. In our previous example, in the *Entry* class, an instance could be protein <P12345>. To name this instance, we attach the object name to the class it belongs to using a colon, and we generally underline the whole designation, *e.g.*, *P12345:Entry*. Every object has a specific value for every attribute given by the object's class. This value can be either a defined or an undefined (Null) value.


P12345: ProteinEntry
accessionNumber = P12345
description = Aspartate aminotransferase
primarySequence = SSWWAH...SKK

It is also possible to have anonymous instances. An anonymous instance of *ProteinEntry* is simply labelled *:ProteinEntry*.

Modelling with object-oriented concepts

Use of objects in modelling brings with it all the objects' related aspects. The most important are:

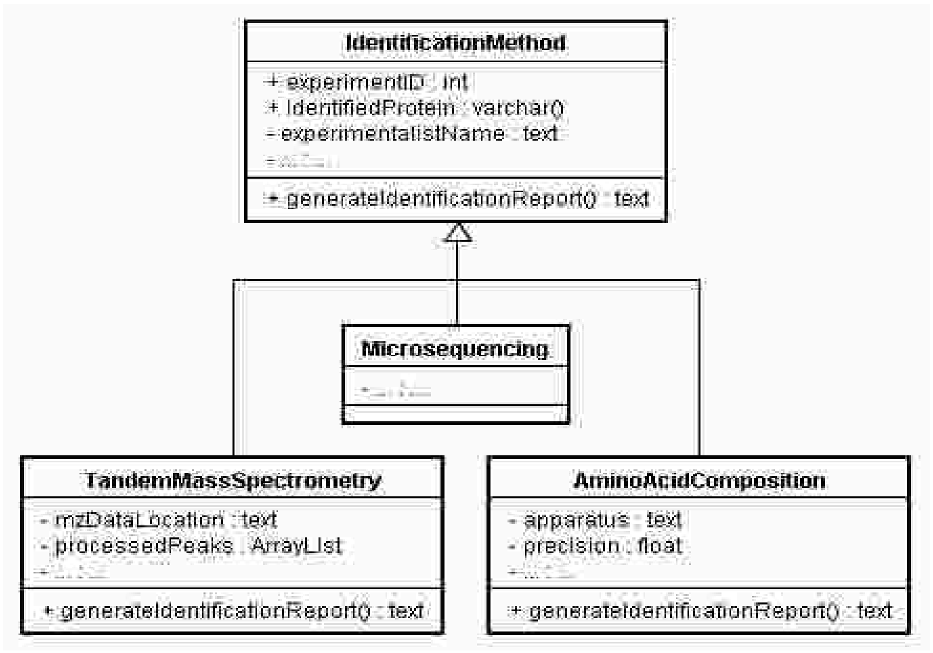
Abstraction

Abstraction means simply to filter out the many properties and operations of an object until only the ones we need are kept. This will greatly depend on the purpose of the object itself, and on how it is going to be used. For example, for people working with different spectrometers within one group, it may be important to know the mass spectrometer serial number, while it will be of no use to anyone outside the group.

A mass spectrometer serial number, for example, is important to know for people working with different spectrometers within one group, while it will not be of any use to any one outside the group.

Inheritance

In the world of object-orientation, classes may inherit all the attributes and operations from another class. This is called inheritance in object-oritentation or **generalisation** in UML, an ' *is a Kind Of* ' relationship. The inheriting classes are then considered as subclasses of the superclass. Subclasses can independently add their own attributes and operations, or even redefine the inherited ones but without affecting their superclass. As an example, *IdentificationMethod* may be the superclass of the subclasses *TandemMassSpectrometry*, *AminoAcidComposition* and *Microsequencing*. These three subclasses share together all inherited features from their common superclass. At the same time, they individually add attributes and operations specific to each of them. Subclasses are related to their superclass using a pointed array line.


IdentificationMethod
+ experimentID : int
+ IdentifiedProtein : varchar()
- experimentalistName : text
+ generateIdentificationReport() : text
Microsequencing
TandemMassSpectrometry
- mzDataLocation : text
- processedPeaks : ArrayList
+ generateIdentificationReport() : text
AminoAcidComposition
- apparatus : text
- precision : float
+ generateIdentificationReport() : text

Polymorphism

An operation may sometimes have the same name in different classes. In the given inheritance example, *genralIdentificationReport()* represents an operation that generates a text report describing the identification process. The method is inherited as is by the *Microsquencing* class, but it is redefined to proceed differently in both the other two subclasses (this is why we write down the method name in these two subclasses). To obtain an identification report, one only needs to perform the "same" operation on any of the different identification classes to obtain the corresponding output report. This is called polymorphism.

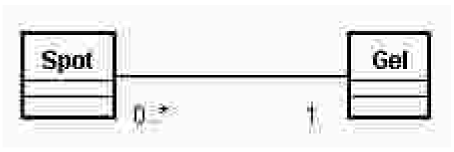
Encapsulation

Encapsulation means that an object can hide what it contains and what it does from other objects (and from the outside world). At the same time, the object needs to present a "face" to the outside world so it can perform some operation when another object is asking for it. To give an example, let us consider the *TandemMassSpectrometry* class as defined in the inheritance example above. We have chosen to make all the attributes private (symbolised by a '-' sign) within the class. This means that all these attributes are hidden outside of the class. Meanwhile, the operation *generateIdentificationReport()* is public (preceded with a '+' sign) so that any other object can ask for the identification text report without caring about what's inside the class. If, for some reason, the class is modified - for example if we choose to provide all the mzData content instead of just giving a file path – only the operation will have to be adapted. The other objects will not be aware of any such inner modification. *GenerateIdentificationReport()* acts as an operation that the *TandemMassSpectrometry* class may present to the outside world through some **interface**. To make use of an interface, the requiring objects will need to send a specific activation **message** that the interface is capable of interpreting and proceeding. We will see later in this section how interfaces are effectively presented in UML.

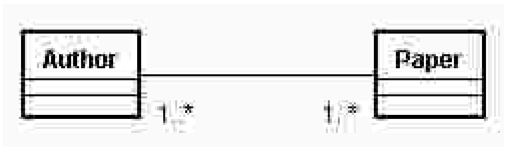
Associations

Objects are typically related to one another in some manner. We may for example define a class called *Gel* containing all the attributes related to the 2-DE gel itself. Now, when we define another class, the class *Spot*, to capture the properties related to the spots, we recognise that each spot is located on a specific gel, which itself is defined as a *Gel* object. There is obviously a relationship between a *Spot* object and a *Gel* object; this is called association. This is represented by a simple line linking two classes. Unless specified, navigation is bi-directional. Whenever the association is limited to just one direction, an arrowhead pointing to the direction of traversal is adorned at one end of the association line (in the relational schema representation, we will not need to explicitly represent navigation directions).

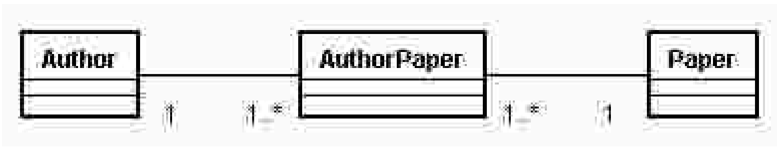
Multiplicity indicates the number of objects in one class that relate to a single object of the associated class. In our *Gel* / *Spot* example, a spot should refer to a unique gel whereas the gel may contain any number of spots. This is called a one-to-many association, and is represented by a '1' at one end and a '0..*' at the other end of the association line, the '0..*' telling that a *Gel* object may contain any number of spots ranging from zero to infinity. We may also give a list of all possible values separated by commas, '1,3,7' meaning for example that only a multiplicity of those three values is authorised.



In the relational model, a many-to-many association cannot be implemented in practice without using an intermediate table. For example, we may have a class for authors and another class for papers. One author may write several papers, while a paper can be written by several authors. The following association:

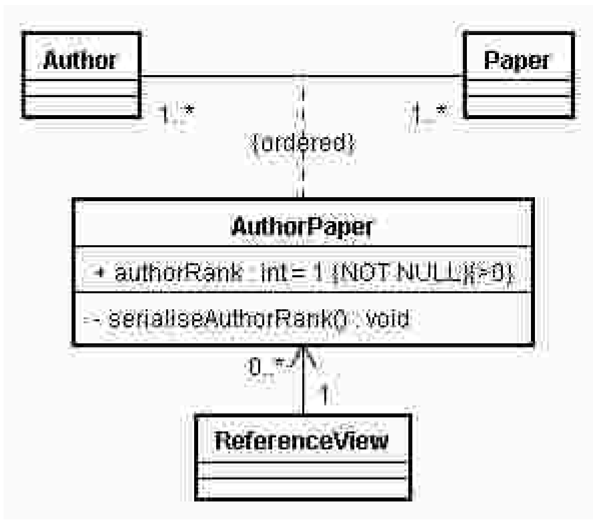


Is then implemented in a relational schema by:



Here, we create *AuthorPaper* objects that reference each a unique author and a unique paper. Reciprocally, an author or a paper may be referenced by any number of *AuthorPaper* objects. In a relational schema, *AuthorPaper* is called "junction table".

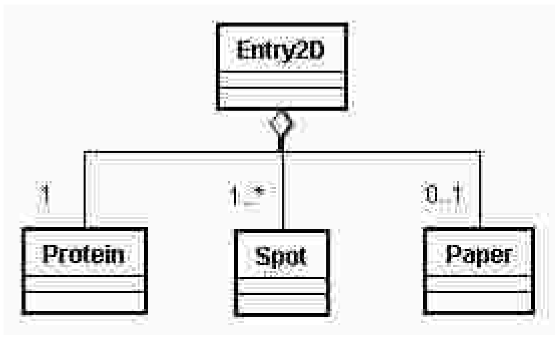
For a many-to-many association, we may prefer to emphasise the visualisation of the model in a better manner. This can be done using **association classes**. In fact, an association can have attributes and operations, exactly like any other class; it can also have associations to other classes. The next illustration shows the use of an association class in the same example as previously given:


Author
Paper
1..*
1..*
{ordered}
AuthorPaper
+ authorRank : int = 1 {NOT NULL}{>0}
- serialiseAuthorRank() : void
0..*
1
ReferenceView

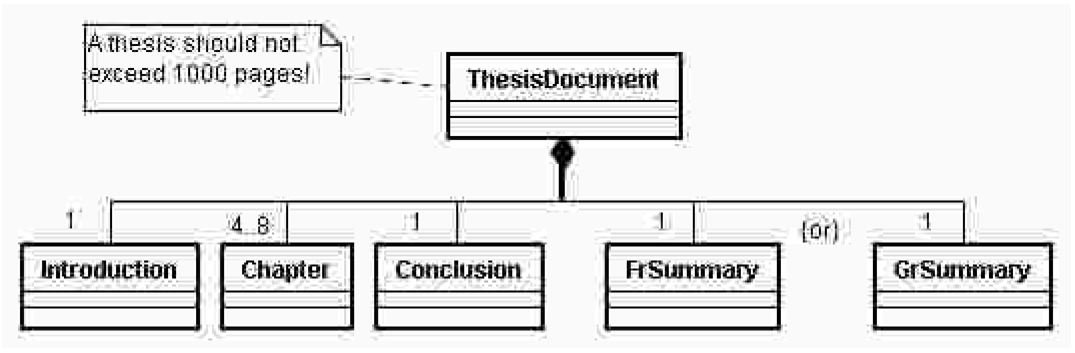
Here, we show the *AuthorPaper* association class making the junction between the *Paper* and the *Author* classes. An attribute, *authorRank*, defines the author's rank in the authors' list and is part of the class features, along with the operation *serialiseAuthorRank()*, which ensures that authors' ranks are serialised (1,2,..). We add the role {*ordered*} to indicate that some sequence ordering is being applied. We also see how *AuthorPapaer* can have associations with any other class, like being referenced by a *ReferenceView* class responsible for building reference views. We may notice that we used the optional navigation arrowhead for the association between *ReferenceView* and *AuthorPaper* to explicitly point out that the former gets information from the latter, but not the other way round.

Aggregation

An object may be composed by a number of different types of components. A protein entry in a 2-DE database may for example contain a unique identified protein, any number of identified spots (at least one) and any number of related publications. This is a ' *has a* ' (and reciprocally ' *part of* ') relationship. We represent these aggregation associations by using a diamond head on top of the associations:



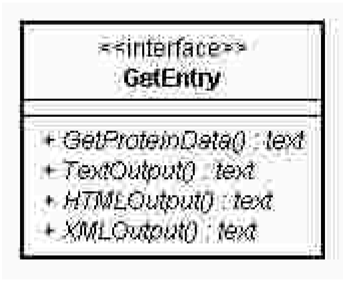
A stronger type of aggregation is the **composition** type. It occurs in situations where the component only exists within the composite object. For example, this document is composed, let us say, of an introduction, 4 to 8 chapters and a conclusion. Each of these components can only exist within this specific document. We may also include a French or a German summary (but not both at the same time):



We use a filled diamond head this time to highlight the composition character. We also introduced two new annotations here. One is the **constraint** implying that the composition should contain a *FrSummary* or a *GrSummary*, but not both. Constraints are written within braces. The second annotation is the **note** that we attached to the *ThesisDocument* class and which can contain any textual comment.

Stereotypes

A stereotype is not an object-orientation concept, but rather a UML feature that permits the creation of new concepts or symbols. Stereotypes are placed on existing UML elements and are presented enclosed in two pairs of angle brackets. This notion is indeed used with interfaces' representation. An interface can be considered as a class that has only operations but no attributes. Instead of using a new element for interfaces, we commonly use a stereotype, like in the following example:

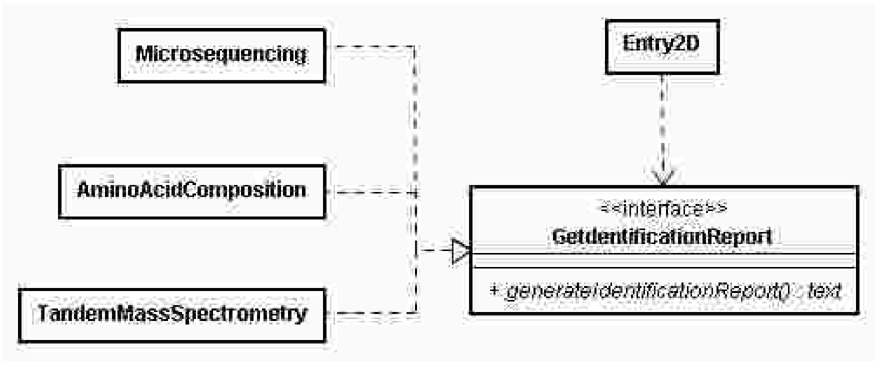


Interfaces and Realisation

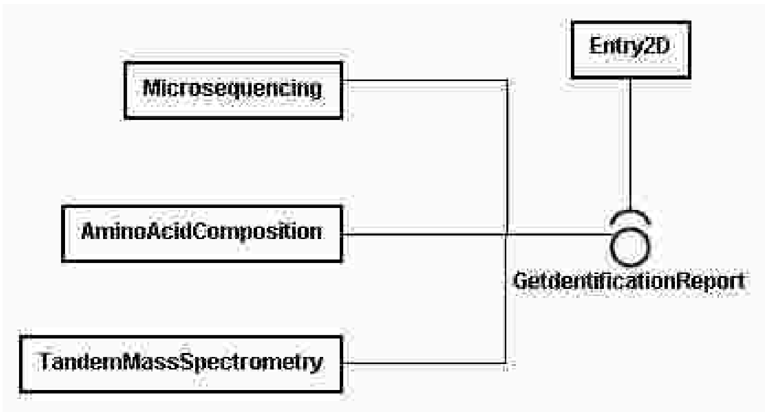
This brings up the discussion back to interfaces. You do not implement the operations in an interface class. A class that implements an interface implements each of the operations defined in the interface class. In other words, the interface class provides no method bodies for the operations it defines; the class that implements the interface provides method bodies for each of the operations defined in the interface.

The relationship between a class and its interface is called realisation. Several classes may "realise" the same interface, and a single class may also realise several interfaces. At the opposite side, a class that activates an interface – by sending it an appropriate message – generally falls into a **dependency** relationship towards this interface. We get back to our previous example on identification classes that implements the public operation *genralIdentificationReport()*. This operation can now be declared in some interface called, for example, *GetIdentificationReport*. All the identification classes should realise this interface. A class that contains, for example, materialised views[1] of 2-DE entries can rely on this interface to get the identification reports, and thus it depends on it. There are two ways for representing this in UML, the full and the elided notation.

The full notation uses for realisation a similar symbol similar to the one used in generalisation, except that the line is dashed. Dependency is also represented with a dashed line but with an arrowhead pointing to the interface:



The more recent elided notation represents the interface by a circle that is called a "ball-and-socket" symbol:



In this document, we will generally use the first notation whenever an interface is introduced for the first time, and the second notation for any further illustration including the already introduced interface.

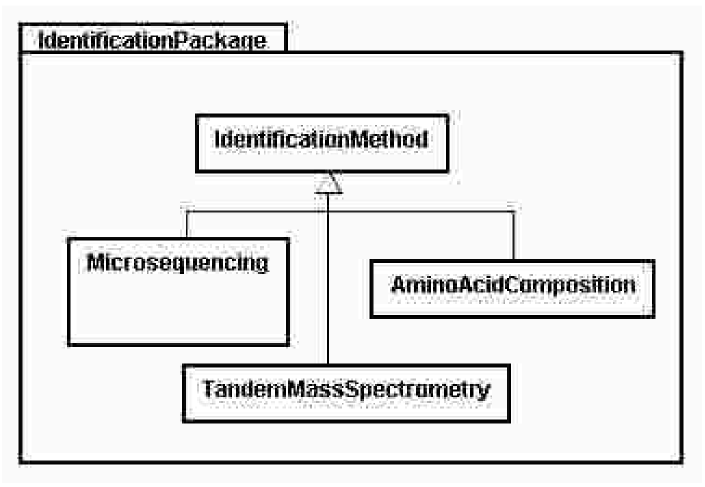
[1] A materialised view in the relational world is a concrete table, as opposed to a virtual view. Materialised views offer more efficient access, but at the cost of being potentially out-of-date. Due to their physical materialisation, we can consider them in modelling as full-fledged classes.

Diagrams

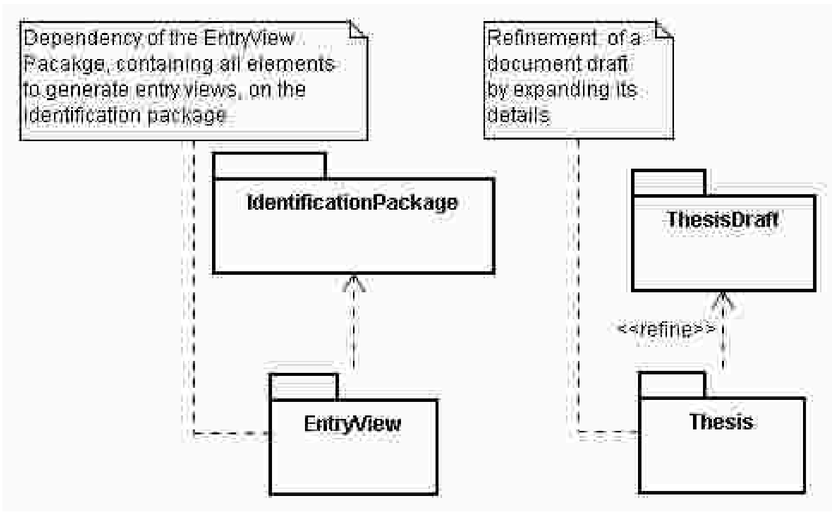
Class, object and package Diagrams

In the Unified Modeling Language, a class diagram is a type of static structure diagram that describes the **structure** of a system by showing the system's classes, their attributes and their operations or methods. Associations and relationships between the classes show how the different classes are related. This type of diagrams is perfectly appropriate for describing data structure and implementation design, but it is also very helpful in problem analyses. A class diagram, when adapted for, can also fit perfectly to portray a relational schema, and this is why we have opted to use it in rendering our database structure. Actually, relational data structures are quite commonly designed using the **Entity-Relationship** (ER) method, a non-UML representation that is meant for relational implementations (Chen 1976). However, the logical relations between objects may not be always obvious using such a method. We prefer to emphasise the logic of these relationships in our representation using class diagrams, and, when needed, some ER elements. Full technical implementation details will be accessible though through external Web links that we will give when necessary.

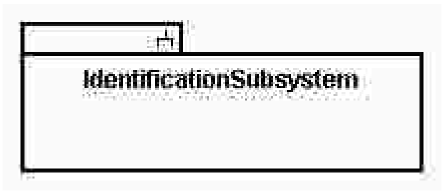
An object diagram simply shows how instances of classes are linked together in an instant of time (like in a snapshot). We have already introduced in the previous sections many of the elements that are employed in such diagrams, the class and the object diagrams, but we have not introduced packages yet. Although it is considered that package diagrams can support most of the many diagram types, we have chosen, for simplification purpose, to only introduce it in conjunction with class diagrams. A package, as its name implies, is designed to group the elements of a diagram. Graphically, it is a tabbed-folder surrounding elements that can be logically grouped together. A package has a name, which becomes the **namespace** of the grouped elements. Consequently, to reference an element within a package, we simply designate it by *PackageName::PackageElement*; this is called a fully qualified name, *e.g.*, *IdentificationPackage::Microsquencing*.



Packages may contain other packages. Besides, they can relate to one another in three different ways: Through generalisation (a package can inherit from another one), through dependency (package elements may depend on the elements of another), and through refinement (a package refines another when it contains the same elements but with more details). The first two relationships use the same symbols we have already used with classes, while the refinement employs a dependency symbol supplemented by the stereotype <<*refine*>>.

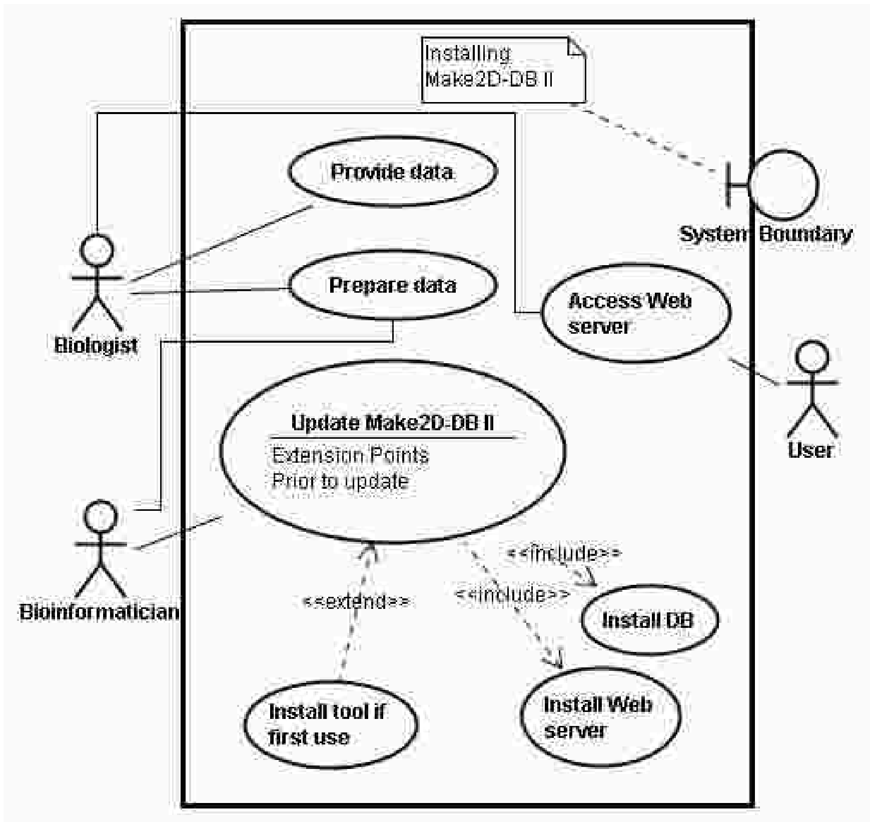

Dependency of the EntryView Pacakge, containing all elements to generate entry views, on the identification package
Refinement of a document draft by expanding its details
IdentificationPackage
ThesisDraft
<<refine>>
EntryView
Thesis

A **subsystem** is a set of elements, which is a system by itself, and a part of the whole system. It is considered both a package and a **classifier** (a category grouping several elements that have some common features). We could have grouped the *IdentificationPackage* elements inside a subsystem instead of a package due to the similarity of their features (and as a consequence of the generalisation they all get from their superclass). Subsystem notation is slightly different from the package one:



Use Case Diagram

Use case diagram is a description of a system's **behaviour** from a user's standpoint. The user himself is represented by a little stick figure called **actor** (an actor is not necessarily a physical person, but rather a defined role). Use case actions are represented by ovals. As an illustration, here is a very simple use case diagram presenting the users' perspectives when installing, updating and using the Make2D-DB II tool:



A use case diagram is noticeably self-explanatory. The different actors have well defined roles, but they may also perform the same use cases. A use case may include another use case using the stereotype <<*include*>> on a dependency symbol. A use case may also be extended by another one, using the stereotype <<*extend*>>. Conditions for the extension can be described in the extended use case.

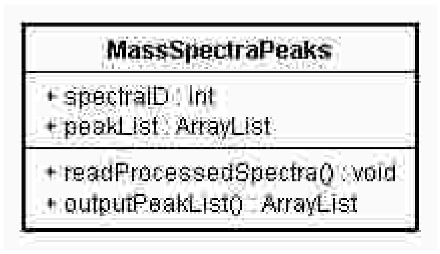
Component Diagram

As its name indicates, a component diagram contains components, along with interfaces and relationships. This type of diagram is especially useful to show how the constituents of a whole or of a subsystem interact together. A component is a modular part of a system that is strictly **logical**. It can perform operations and can provide interfaces to other components. Components are autonomous and they define a system's functionality, by contrast to data files, documents, executables or dynamic libraries that are all pieces of information called **artefacts** that a system uses or produces[1]. An executable, for example, is the implementation of a component, just as a component is a logical "implementation" of one or more classes.

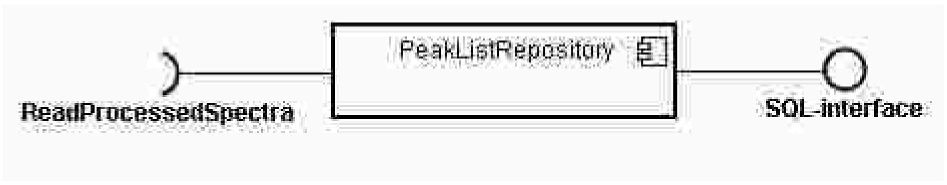
Component diagrams are useful because they provide a high-level, architectural view of the system. The idea is that we should easily be able to reuse or substitute any component implementation in a design because a component encapsulates behaviour and implements specified interfaces.

[1] Until UML version 2.0, artefacts were frequently considered as components, which have led to much confusion among modelers.

To illustrate this idea, let us think of a concrete example. We want to stock all mass spectra peak lists in a very simple repository that – for simplification purpose – only stores the different spectra peaks and gives each list a unique identifier. We may module this by a plain class that we call *MassSpectraPeaks*. This class has an operation that reads the processed spectra, and provides an operation that output the peak list values:

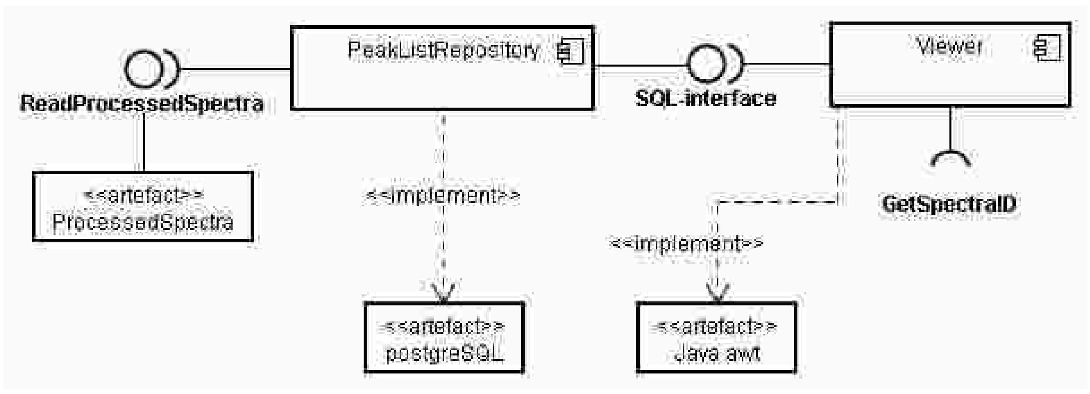

MassSpectraPeaks
+ spectralD : Int
+ peakList : ArrayList
+ readProcessedSpectra() : void
+ outputPeakList() : ArrayList

At that point, we make the decision to implement our model using a relational database implementation. This implementation must be able to read the processed files (being themselves artefacts) and to provide an interface that output peaks lists (*e.g.*, using SQL queries). This is the autonomous component that reflects the logical implementation of our system:



We may decide to physically implement our component using any suitable relational database management system, *e.g.*, a PostgreSQL or an Oracle implementation. This can be shown by a dependency relationship stereotyped <<*implement*>> between the component and the implemented artefact. The implementation choice should never affect the logic dictated by the component.

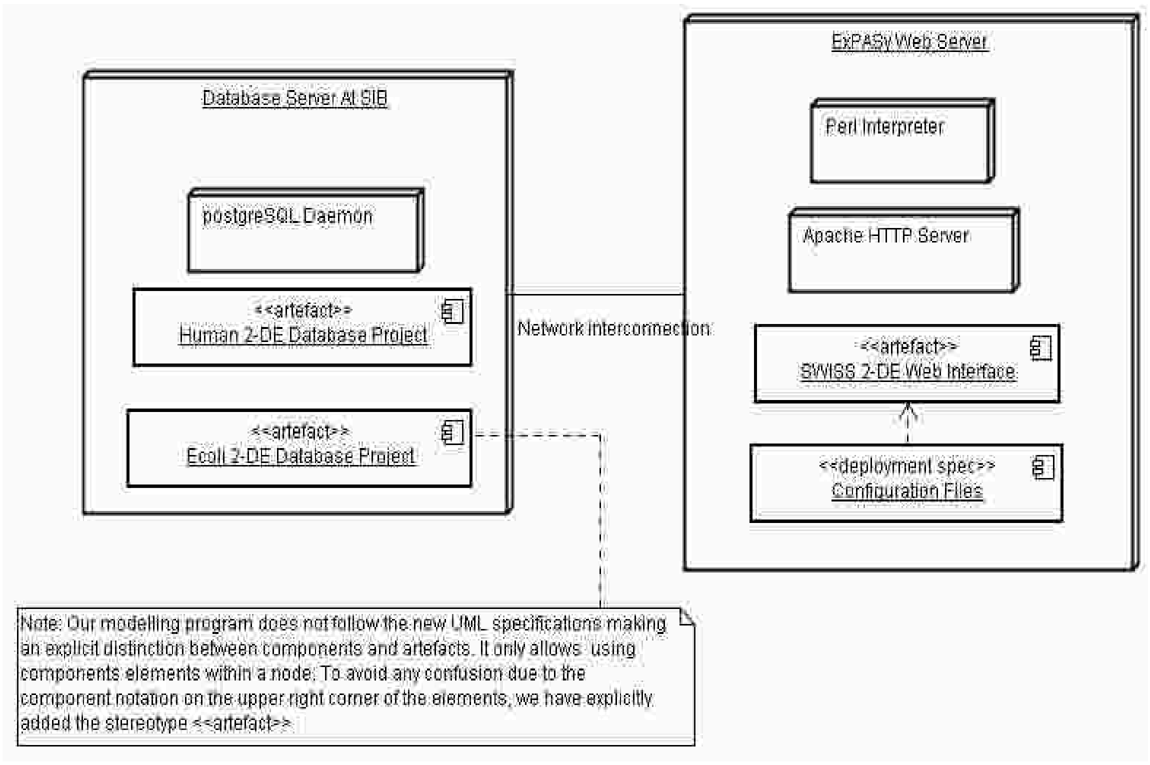
We may choose, in a further step, to link our component to another component, a viewer, which gets a spectrum identifier, generates a SQL query and then graphically displays the spectrum. The physical implementation of such a viewer may also be made in any appropriate language.



A component can inherit from another component and may contain inner components as well.

Deployment Diagram

A deployment diagram shows how the artefacts, already presented as components' implementation, are deployed on system hardware. The elements used in deployment diagrams are nodes (embodied by cubes), artefacts and associations (links). Outer nodes are devices, while inner nodes indicate execution environments (operating systems, language interpreters, etc.) rather than hardware. In addition to classic associations, elements may be related through generalisation and dependency relationships or communicate through interfaces.



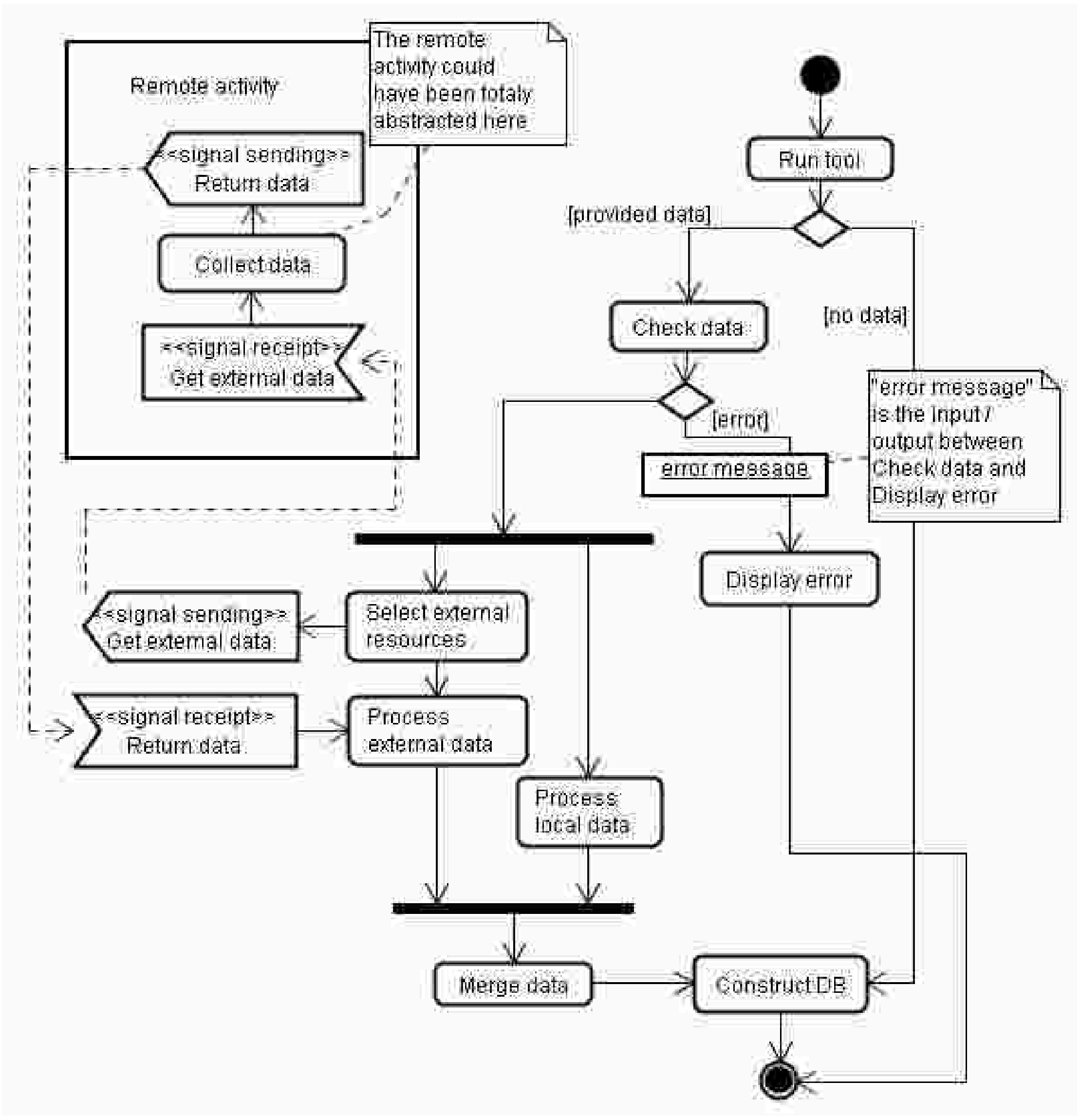
Note: Our modelling program does not follow the new UML specifications making an explicit distinction between components and artefacts. It only allows using components elements within a node. To avoid any confusion due to the component notation on the upper right corner of the elements, we have explicitly added the stereotype <<artefact>>

We have chosen to portray a fictive installation of two instantiated artefacts of Make2D relational databases installed on a specific database server (an instantiated node) that are accessed by the implemented query interface of the same tool installed on another specific Web server. We also see that it is possible to designate an artefact that provides parameters for another artefact using the stereotype <<*deployment spec*>>; this is what we have employed with the 2-DE Web interface in the same example.

Activity Diagram

An activity diagram is designed to illustrate what happens during an operation or a process. Activities are symbolised by a rectangle with rounded corners. When an activity is completed, a transition to the following activity occurs, which is represented by an arrow pointing to the next activity. Activity diagrams have a starting point (a filled-in circle) and an endpoint (a bull's eye). Whenever a decision must be made, the path is split into two (or several) mutually exclusive paths, each with its own condition written between brackets. A diamond is used to state that a condition has to be taken with the possible paths flowing out of it. Concurrency is represented with a solid bold line perpendicular to the transition with the paths coming out of it. The same kind of line is used to merge the concurrent paths together. Activity's input and output can be specified using small boxes over the transition. In addition, any activity process should be able to send **signals** to other processes; this is indicated by a convex polygon for the transmission event, and a concave polygon for the reception event.

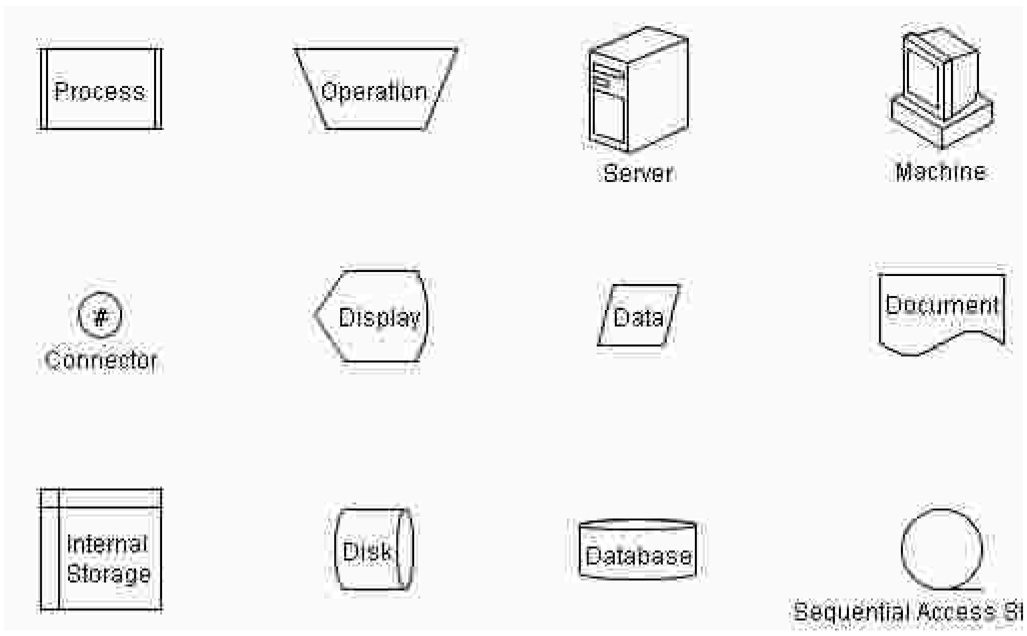
We display here a simplified activity diagram representing the process of launching the Make-2D-DB II tool. This part schematises, for illustration purpose, the checking of the provided data (if any), the integration of some external data, and the construction of the relational 2-DE database:



Remote activity
The remote activity could have been totaly abstracted here
<<signal sending>> Return data
Collect data
<<signal receipt>> Get external data
Run tool
[provided data]
[no data]
Check data
[error]
error message
"error message" is the input / output between Check data and Display error
Display error
<<signal sending>> Get external data
Select external resources
<<signal receipt>> Return data
Process external data
Process local data
Merge data
Construct DB

An activity diagram can easily be made hybrid by incorporating elements from other diagrams. For example, it can incorporate actors to show who does what, or objects that perform some operations.

We find it suitable to incorporate some standard flowchart elements within an activity diagram, even though flowchart elements are not UML elements. This will offer a much expressive visualisation of the activity details:



Before concluding this appendix, it is important to insist on the fact that this summary was not intended to exhaustively depict UML notations. We have for example skipped all dynamic and time-dependent representations, *e.g.*, state and sequence diagrams. Our intention was only to portray the elements that we will be using throughout this document.

APPENDIX V. RELATIONAL DATABASES

And the PostgreSQL ORDBMS

Relational databases

To present the relational model, relational databases and the SQL language is beyond the scope of the present document. There are many printed publications and Web tutorials covering these subjects, and the reader is welcome to consult them for more details. This appendix intends to summarise some of the terms that we are using throughout this manuscript, and to present some of the aspects of PostgreSQL, the database management system we have adopted for our project.

A database can be assimilated to a collection of related files. How those files are related depends on the model used. Early models relating files included the hierarchical model (a parent/child relation between files), and the network model (an owner and member relation). The relational database model was invented in 1970 by Codd[1] (Codd 1970). This was a significant advance, as it allowed files to be related by means of a common field. To be related a pair of files only needs to have in common one or several fields and that makes the model extremely flexible.

Relational databases are built in a Relational Database Management System: **RDBMS**. The most common definition of a RDBMS is a system that presents a view of data as a collection of rows and columns not based strictly on relational theory. The majority of popular RDBMS products do not necessarily implement all of Codd's 12 rules. Almost all RDBMS employ **SQL**, the Structured Query Language, as their query language. SQL is based on relational algebra. It is an ANSI and ISO standardised computer language used to create, retrieve, update and delete data from relational databases. An introduction to SQL is available at the w3schools Web site[2].

[1] http://en.wikipedia.org/wiki/Relational_database

[2] http://www.w3schools.com/sql/default.asp

Components

Strictly speaking, a relational database is a set of **relations**, commonly known as **tables**, in addition to other items that help to organise and structure the data.

Relations or tables

A relation, represented by a table, is defined to be a set of tuples (a finite and ordered sequence of distinct objects) having all the same attributes. A table is therefore organised in rows and columns, like in the following example (Table *Spot*).

attributes (columns)

tuples (rows)

'able *Spot*	*spotID* (serial)	*pI* (float)	*Mw* (int)	*identified* (boolean)	*identifiedProtein* (varchar)
tuple 1	100	5.50	22300	True	P12345
tuple 2	101	6.50	32500	False	Null
tuple 3	102	7.89	40000	True	P34567
tuple n	…	…	…	…	…

In a relational database all data is actually stored as relations. The term **relavar** is the "relation variable" which is physically schematised by a table. Some relvars do not have their data stored in them but are the outcome of the application of relational operations to other relvars. These relavars are then called "derived relavars", or simply **views**. At a certain level, derived relvars are not strictly considered as part of the relational model.

Constraints

Restrictions on the kinds of data that can be stored in the relation are called constraints. They are formally defined in the form of expressions that result in a boolean value.

Data domain

Data domain may be considered as a constraint, in the sense that it defines the set of possible values for a given attribute. In the *Spot* table, the *pI* float attribute should be limited to the numeric data domain between 0.00 and 14.00 not included, and Mw should be only positive.

Defined values and uniqueness

Some attributes should be defined, meaning that they should be given a value; a **Null** value being an undefined value, constraining an attribute to be defined is represented by the constraint {Not Null}. For example, *Mw* in the *Spot* table has a {Not Null} constraint. At the same time, some other attributes, if defined, should be **unique** in their relation; this is represented by the constraint {Unique}. A gene alias for example should be unique in a *Gene* relation but does not have the {Not Null} constraint. Whenever an attribute has both constraints, it falls in what is known to be a primary key.

Keys

A key is a kind of constraint that implies that a tuple, or part of the information it contains, is not duplicated in a table. This is achieved in a relational database using a **Primary Key** constraint {PK}. In the *Spot* Table, *spotID* is indeed a primary key. Each spot (a tuple) must have a defined and unique value for this attribute. A primary key may also be composed of several attributes, each of them having to be defined. Only the combination of these attributes has to be unique. In a *Person* relation, where we may define two attributes *firstName* and *FamilyName*, a primary key would be the combination of both the two attributes. Several people may be of the same family, and several people may share the same first name as well, but two persons should never have exactly the same first name and family names.

Foreign keys

A **Foreign Key** {FK} is a reference to a primary key in another table. The referencing tuple has, as part of its attributes, the same value(s) of a key in the referenced tuple. Foreign keys may reference defined {Unique} attributes as well as {PK} attributes.

More constraints

In the *Spot* table, *Mw* must be a positive value, which is expressed by {Check: *Mw* > 0}. We have also intentionally included the attribute *identified*, which tells if a spot has been identified or not. It seems logical that a non-identified spot should not present any protein in its *identifiedProtein* attribute. We can express such a constraint by the boolean expression: {Check: *identified* is True OR (*identified* is False AND *identifiedProtein* is Null)}. Constraints may contain also functions, procedures or operations in the boolean expression. If, instead of the attribute *identified*, there was a function that returns a boolean value telling whether or not a spot has been identified, we would employ a constraint of the form {Check: *hasBeenIdentified(spotID)* is True OR *identifiedProtein* is Null}.

Rules, stored procedures and triggers

A **rule** in a database is a way to rewrite some specific SQL queries, or to automatically add some additional SQL instructions to an initial one. An example would be to automatically switch a flag attribute to ON whenever some relevant information has been modified. The switch of an *annotationChanged* attribute to ON within an *EntryVersion* tuple whenever some specific annotations found in other tables and related to the *Entry* tuple are modified represents a concrete illustration of a rule.

Stored procedures are executable code associated with the database. The RDBMS may offer the possibility to perform some "complex" operations using one or several programming languages or scripts. Procedures are assimilated to functions. They may operate with some specific input parameters and they always return a value of a definite data type. Stored procedures may perform many kinds of common operations, among which:

- Simple control operations, *e.g.*, *hasBeenIdentified(spotID)* : boolean
- Statistical operations, *e.g.*, *numberOfIdentifiedSpots()* : int
- Select operations, *e.g.*, *selectAllIdentifiedSpotsSinceDate(Date)* : record/table
- *Update* operations, *e.g.*, *updateProteinEntryView(accessionNumber)* : boolean
- ...

Besides, a procedure may execute another procedure, and may even generate and compile code for new procedures.

Triggers are activation processes that are associated with some function or procedure and that are "fired" before or after a specific operation is attempted on a tuple. In insert, update or delete operations, triggers can "see" any potential old and new attribute value, which is very convenient when having to perform some special operations depending on data substance. For example, we may want to fire a trigger associated with a function, which increments a protein entry version by 1, if, and only if, some specific attributes are modified and that their new values are different from the initial ones.

Indexes and sequences

Indexes, based on one or several attributes, are used to improve performance when accessing tables. They help to reduce the scanned subset of tuples. Although it is the **optimiser** - the process responsible for setting up the best query plan to adopt – that decides whether to use a relational index to access data, it is up to the database manager to build up the appropriate indexes that will improve performance. RDBMS provide different indexing methods, among which the B-tree (a tree data structure) and the hash table (keys/values association) methods.

Sequence objects are special single-row tables that help to generate automatically ordered attributes' values. Sequences are usually used to generate unique sequential identifiers for a table's rows. The special integer data type **serial**, used above with *spotID*, is an example of the use of a sequence to generate automatic identifiers.

PostgreSQL

To physically realise and manage our data model, we needed a stable database management system that offers extended capabilities in implementing stored procedures, and in using slightly complex data types. Besides, the system had to be free of charge, our main condition in the choice of the component.

PostgreSQL[1] is an open source **ORDBMS** (Object-relational Database Management System) developed at the University of California[2], and running on all major operating systems. PostgreSQL is not controlled by any single company, but it relies on a global community of developers and companies to develop it.

In fact, it would be more correct to consider PostgreSQL as a conventional RDMBS enhanced with a layer simulating object-oriented characteristics. These characteristics are:

- Object Identifier (OID): each tuple (object) has a unique identifier within the same table and this identifier is independent from the tuple content. Tables containing rows have also a table object identifier.

- Inheritance: Tables can be set to inherit attributes and behaviours from a parent table. Data is then shared between parent and child(ren) table(s). Yet, some of the constraints are not currently inheritable.

- Complex data types: In addition to the multi-dimensional arrays, users can create their own complex data types.

- Large objects: PostgresSQL has a facility to store binary large objects, also known as BLOBs, such as graphical and XML files. Objects are then manipulated using their OIDs. However, portability of BLOBs is not supported.

[1] http://www.postgresql.org/

[2] Copyright © 1996 – 2007 PostgreSQL Global Development Group, under the BSD license

An overview of the main differences between Object-oriented Database Management Systems (ODBMS) and RDBMS is summarised in the following table:

Aspect	Object-oritented	Relational
Data accessibility	permanent data	stored data
Entities	objects (classes)	normalised relations (tables)
Identifiers	object identifiers (OID)	primary keys
Data structure	complex data types	atomic attributes
Functionality	object behaviour (operations, methods)	procedures, rules, triggers, functions
Modelling	object type	relational schema
Inter-dependency	inheritance, encapsulation, polymorphisme	independence of relations
Integrity / concurrency	risky	good
Speed performance	very high	average
Specifications' stability	low	high

A few non-commercial ODBMS have been offered only over the last couple of years. Before this, the non-availability of functional free of charge object-oriented systems was one of the main reasons why we did not consider the possibility to adopt an object-oriented database approach for our project.

Advantages using PostgreSQL

Being reasonably stable, PostgreSQL has a number of significant advantages - related to our work - when compared to conventional RDBMS (*e.g.*, MySQL[1]):

- It runs stored server-side procedures in more than a dozen programming languages, including Java, Perl, Python, Ruby, Tcl, C/C++, and its own PL/pgSQL[2], which is a procedural language similar to Oracle's PL/SQL.

- It has a rich set of native data types available to users. The most important are: arbitrary precision numbers, variable-length character texts, multi-dimensional arrays of any data type (unfortunately with very limited capabilities to exploit the arrays). It has as well a set of pseudo-types like the "any" (indicating that a function accepts any

[1] http://www.mysql.com/

[2] Make2D-DB II uses extensively PL/pgSQL (http://www.postgresql.org/docs/8.2/interactive/plpgsql.html)

data type), and "record" (for a function to return an unspecified row type). Users may also add new complex types.

- It extends standard constraints' expressions using "{Check *boolean expression*}".

- It introduces namespaces for objects through schemas' repartition.

- It optimises many management features, including user-defined indexation of tables and ensures integrity with MVCC (multi-version concurrency control). It also performs statistics for efficient execution plans of queries and for garbage collection.

- PostgreSQL also offers extended capabilities to work with regular expressions, which is very convenient in analysing data and in assembling human readable views.

There are also many library interfaces (APIs) allowing various languages, both compiled and interpreted, to interface with PostgreSQL. There are interfaces for Perl (DBI), Java (JDBC), ODBC, Python, Ruby, C, C++ and PHP to name the most common of them.

A significant drawback, which mainly originates from our extensive use of server-side procedures, is the need to cope with any specifications' changes between PostgreSQL public releases. Adaptations of our code were sometimes necessary to deal with some of the critical specifications' changes.

APPENDIX VI. THE INSTALLATION PROCESS

Screenshots / shell captions

Information regarding the tool using the *<help>* option (a truncated shell caption):

```
Perl make2db.pl -help

MAKE2DB(1)     User Contributed Perl Documentation     MAKE2DB(1)

The Make2D-DB II Package ( version: 2.50.1 / September 2006 )

       Before, launch:

         createlang plpgsql template1 --pglib "/pgsql/lib path"
         initdb -D [DB path]
         postmaster  -i -d 1 -D [DB path] > [DB path/server.log] 2>&1 &


       Execution:

       perl make2db.pl -<m> [option]

         where option is one of:

         config       -> Set up the configuration…

         check        -> Check the syntax and the consistency of your database…

         check-report -> Same as 'check', except it does not stop on major
                         errors…

         create       -> Create the relational schema for a database from
                         scratch…

         transform    -> Combine the 'check', the 'create' and the 'server'
                         options…

         update       -> For both updates of the schema structure and the
                         database…

         server       -> This option can be used independently if you wish to
                         host an interface to query others remote databases
                         without even having your own database. It is also to
                         be used when some errors are encountered due to
                         invalid permissions         while moving some of the
                         files to the HTTP server. The script can then be re-
                         executed with this option to only set up the HTTP
                         server files.

         e.g:  ' perl make2db.pl -m transform '

         If you want to specify another path for your configuration files to be
read
         or written (not the default one),
         use the switch '-c' followed by by the new path:
```

```
         ex:  perl make2db.pl -m config -c your_path_here

         A shortcut to run a default configuration process without any special
choice

              perl make2db.pl -m config default

         For help (diplays this text):
           type 'perl make2db.pl -h' or 'perl make2db.pl --help'

         To exit this manuel, press the letter 'q'

AUTHOR
      Khaled Mostaguir, khaled.mostaguir@isb-sib.ch

ACKNOWLEDGEMENTS
       An evolution of the make2ddb package concepts (Christine Hoogland and
al).

2006-09-25                 perl v5.6.1                 MAKE2DB(1)
(END)
```

Running the tool with the <*config*> option (a shell caption):

```
perl make2db.pl -m config

*** Make2D-DB II Package - version: 2.50.1 (04-Sep-2006) ***
==============================================================

 [The Configuration File Generator - version: 2.50]

* You are about to set up the configuration files for the Make2DB-DB II tool *

Two files can be set up:

- include.cfg  : configuration parameters for the instllation of the database
- 2d_include.pl: configuration parameters for the the WEB server


Please, choose between setting up your configuration files for a new database
installation, or changing the parameters of an already running WEB server:

--> [1] New Installation
--> [2] Changing a Running Server Parameters
--> [3] Generate a new maps' file
--> [4] Exit

→ [1]

* Prepare your configuration files for a new installation *

Please, choose between reading the default setting values from the default
configuration files, provided with the tool, or from your very last personal
configuration files.

You may also configure only a Web portal, to contact other remote interfaces,
without necessarily providing any personal data.

--> [1] Default configuration files
--> [2] Last personal configuration files
--> [3] Configure a Web portal
--> [4] Exit
```

Running the tool with the *<transform>* option (a shell caption):

```
perl make2db.pl –m transform

*** Make2D-DB II Package - version: 2.50.1 (04-Sep-2006) ***

--- Date: Mon Oct 15 18:02:39 CEST 2007
    Perl version: This is perl, v5.6.1 built for i386-linux
    Lang: en_US.iso885915
    System: Linux mordor.expasy.org 2.4.23 #1 Tue Dec 23 09:22:28 CET 2003 i686

    [option: -m transform]

--- Database official name: 'Test Database'
--- Database postgreSQL name: 'test_database'

... Updating the Cross-Reference list from the ExPASy server..
... Updating the Swiss-Prot tissue list and aliases from the ExPASy server..

Extracting annotation from maps / from text/xml reports in progress...

No flat file yet. Your data is assumed to be in tabulated (spreadsheet) format
or in Melanie XML generated files.

The following files are being analyzed and translated to build a flat file
ending with a 'spot data' section:
 -- /home/world-2dpage/test_database/FIGURE1.txt
 -- /home/world-2dpage/test_database/FIGURE2A.txt

Flat file automatically generated!

Do you want to examine or edit the new generated flat file (to add, for
example, some extra annotations or cross references)?
Type 'yes', otherwise press <RETURN> to continue

Importing mapping methods definition into the new database...

Now checking the flat file structure, syntax and consistency...

**UniProtKB/Swiss-Prot - Swiss-Prot --
...Checking entry P00277
...Checking entry P00350
…
Connecting to the Newt database to retrieve taxonomy data (please, be
patient)...
..... Taxonomy Retrieval Performed!
…
Connecting to the ExPASy SRS server to retrieve external data (please, be
patient)...

WARNING: one of your UniProtKB accession numbers (P31059 / TaxID=83333) has
been demerged!!
[1] P0A7C0; P31059; P97542; [TaxID:83334] {Escherichia coli O157:H7.}
[2] P0A7B9; P31059; P97542; [TaxID:217992] {Escherichia coli O6.}
[3] P0A7B8; P31059; P97542; Q2M8M8; [TaxID:83333] {Escherichia coli (strain
K12).}

assigning the following UniProtKB entry for P31059 => P0A7B8

..... SRS Retrieval Performed!

Connecting to the ExPASy server to retrieve external links for computable
maps...
..... Computable Maps Retrieval Performed!

Your DataBase file has been checked with success!
You can check the file 'last_STDOUT.log' for a detailed report:
   '/home/world-2dpage/Make2D-DB_II/temp/last_STDOUT.test_database.log'

Ready to start the 'test_database' DataBase Construction and to upload your
data into it.
Press 'RETURN'! (ctrl-C to abort)
…
```

The relational database implementation using the <*transform*> option (a truncated shell caption):

```
…
Now Creating your new Database...

Creating a new postgreSQL user named 'select2d' to query the database without
owning it...
< Press 'RETURN' to continue >

SCHEMA CREATION correctly performed!
SCHEMA USAGE correctly granted!

Loading functions...
SET
CREATE FUNCTION
COMMENT
...
Functions properly uploaded.

Constructing the referential tables...
SET
CREATE SEQUENCE
...
CREATE TABLE will create implicit sequence "xrefdbparent_xrefdbcode_seq" for
"serial" column "xrefdbparent.xrefdbcode"
CREATE TABLE / PRIMARY KEY will create implicit index "xrefdb_pkey" for table
"xrefdb"
CREATE TABLE / UNIQUE will create implicit index "xrefdb_xrefdbname_key" for
table "xrefdb"
NOTICE:  merging column "xrefdbcode" with inherited definition
CREATE INDEX
CREATE TABLE
COMMENT
...
Database tables structure properly built.

Adding userstamp and update columns on each table...
NOTICE:  merging definition of column "userstamp" for child "xrefdb"
...

Setting up the log (backup) part...
Log (backup) setting properly built.

Loading triggers...
SET
CREATE FUNCTION
COMMENT
CREATE TRIGGER
COMMENT
...
Triggers properly uploaded.

** Loading your DATA into the referential tables... **
INFO:  vacuuming "core.tissuesp"
INFO:  "tissuesp": found 1148 removable, 1148 nonremovable row versions in 27
pages
INFO:  "tissuesp": moved 1120 row versions, truncated 27 to 14 pages
INFO:  index "tissuesp_pkey" now contains 1148 row versions in 15 pages
-- TissueSP table is being treated...
-- TissueSP table has been filled in
...

Writing the general database related data to the appropriate table...
Loading done.

Uploading functions to update/alter some internal data related to external
ones...
SET
CREATE FUNCTION
COMMENT
...
Internal data update functions properly integrated (not yet performed).

Loading full entry view functions and tables (materialized views)...
SET
```

```
CREATE FUNCTION
COMMENT
...
Full entry views functions and tables properly loaded.

Update Internal Data And Construct VIEWS:
NOTICE:  Entry construction in progress...
NOTICE:  Analyzing tables...
NOTICE:  CREATE TABLE / UNIQUE will create implicit index
"buffer_make2db_reunit_refs_referenceid_key" for table
"buffer_make2db_reunit_refs"
...
NOTICE:  ...entry P00816 is processed
NOTICE:  ...entry P09394 is processed
...
NOTICE:  Full Entries Table has been constructed/updated!

NOTICE:  Protein list in progress...
NOTICE:  Protein Lists are being now processed for each Map. Please wait!...
NOTICE:  ... Protein lists for the different maps are still in progress...
NOTICE:  ... FIGURE1 is being processed
...
NOTICE:  Data Updates Performed With Success!
Database Internal Data Updated and Views Constructed!

Performing database displayed statistics...
Database statistics performed!

Session information inserted into the 'Make2DDBTool' table.

Updating entry views for version update...
NOTICE:  Entry construction in progress...
NOTICE:  ...entry P00816 is processed
...
Database Conversion has been properly performed.

SCHEMA public revoked from PUBLIC.
granting 'select2d' SELECT rights on common.Database
...

Copying indexes to public schema...
creating index (CREATE UNIQUE INDEX release_pkey ON release USING btree
(releasenum, subrelease))
...

Export To Public Data:
NOTICE:  Entry construction in progress...
NOTICE:  Analyzing tables...
NOTICE:  ...entry P00816 is processed
...
NOTICE:  Full Entries Table has been constructed/updated!
NOTICE:  Protein Lists are being now processed for each Map. Please wait!...
NOTICE:  ... FIGURE1 is being processed
...
NOTICE:  Data Updates Performed With Success!
Database Public Export Performed With Success!

Analyzing Database to ensure optimal performance. Please, be patient as this
may take some time...
Analyze performed with success!

Database Installed!
```