PFDB: A Generic Protein Family Database integrating the CATH Domain Structure Database with Sequence Based Protein Family Resources

Adrian J Shepherd\textsuperscript{1,2*}, Nigel J Martin\textsuperscript{2}, Roger G Johnson\textsuperscript{2}, Paul Kellam\textsuperscript{3}, Christine A Orengo\textsuperscript{1}

\textsuperscript{1} Department of Biochemistry and Molecular Biology, University College London, Gower Street, London WC1E 6BT, United Kingdom
\textsuperscript{2} School of Computer Science and Information Systems, Birkbeck College, Malet Street, London WC1E 7HX, United Kingdom
\textsuperscript{3} Wohl Virion Centre, Department of Immunology and Molecular Pathology, Windeyer Institute of Medical Sciences, 46 Cleveland Street, London, W1P 6DB, United Kingdom

* To whom correspondence should be addressed.

Abstract

The PFDB (Protein Family Database) is a new database designed to integrate protein family-related data with relevant functional and genomic data. It currently manages biological data for three projects – the CATH protein domain database (Orengo et al., 1997; Pearl et al., 2001), the VIDA virus domains database (Albà et al., 2001) and the Gene3D database (Buchan et al. 2001). The PFDB has been designed to accommodate protein families identified by a variety of sequence based or structure based protocols and provides a generic resource for biological research by enabling mapping between different protein families and diverse biochemical and genetic data, including complete genomes.

A characteristic feature of the PFDB is that it has a number of meta-level entities (for example aggregation, collection and inclusion) represented as base tables in the final design. The explicit representation of relationships at the meta-level has a number of advantages, including flexibility – both in terms of the range of queries that can be formulated and the ability to integrate new biological entities within the existing design. A potential drawback with this approach – poor performance caused by the number of joins across meta-level tables – is avoided by implementing the PFDB with materialised views using the mature relational database technology of Oracle 8i. The resultant database is both fast and flexible.

This paper presents the principles on which the database has been designed and implemented, and describes the current status of the database and query facilities supported.
Introduction

The Protein Family Database (PFDB) is a new database that currently manages biological data for three projects – the CATH protein domain database (Orengo et al., 1997; Pearl et al., 2000), the VIDA virus domains database (Albà et al., 2001) and the Gene3D database (Buchan et al., 2001). Future additions to the database will include protein families identified for specific biological systems or organisms. The database is currently being extended to include protein families identified in the eye (EyeSite database, Slingsby et al., personal communication). The PFDB provides a mechanism for integrating protein families identified using independent classification protocols.

The earliest protein family databases were constructed using sequence-based methods for recognising evolutionarily related proteins. There is currently a wealth of such protein family databases, many now using powerful profile based or hidden Markov model methods for recognising distant homologues (e.g. Pfam, Bateman et al., 2000; PROSITE, Hofmann et al., 1999; PRINTS, Attwood et al., 2000; and ProDom, Corpet et al., 2000). In many of these databases, protein families often contain more than one protein domain whilst some use more sophisticated approaches to facilitate recognition of individual domains or functional units (e.g. ProDom, PRINTS). Although structural data is more sparse with approximately only 13,000 protein structures currently determined compared to over 12 million sequences, structure is much more highly conserved within a protein family allowing more distant homologues to be more readily identified. Furthermore, domain boundaries can be more easily determined from the 3D structure, both automatically and manually. Several structure databases have now been established (e.g. SCOP, Lo Conte et al., 2000; CATH, Pearl et al., 2001; see Orengo et al., 2001 for a review).

The motivation for PFDB is the need to integrate protein families identified using structure-based protocols with those determined using various sequence-based protocols together with all the available functional and genomic data for these families. Such integration enables queries between families and thereby facilitates mapping of individual structural domains onto sequence based families. Mapping is achieved using a common identifier (GenBank, NCBI, Benson et al., 2001) where possible and using simple pairwise sequence alignment methods (e.g. FASTA) where necessary.

A key data resource underpinning the PFDB is the CATH domain structure database which currently contains some 1200 protein superfamilies identified using both sequence and structure based protocols (Pearl et al., 2001). Individual domains are identified automatically using a consensus approach (Jones et al., 1998) and a recently developed method that detects recurrent domains (Harrison et al., 2001). Any ambiguous assignments are validated manually (Pearl et al., 2001).

Originally a flat-file database of protein structural domains, CATH has expanded dramatically over the past 18 months to include large quantities of new data – notably sequence families derived from ~300,000 GenBank sequences, and genomic data from GenBank (Benson et al., 2000). These have been identified using profile based methods and hidden Markov models (PSI-BLAST, Altschul et al., 1997; IMPALA,
PFDB: A Generic Protein Family Database

Schaffer et al., 1999; SAMt, Karplus et al., 1998), and a DomainFinder algorithm (Pearl et al., 2001) which determines the sequence region corresponding to a given structural domain.

Since CATH is so widely used within the biochemistry community, it is essential that the new data in CATH is managed effectively. Furthermore, there is increasing need to map between the structural families identified in CATH and other local protein family resources (e.g. VIDA, EyeSite) and future databases established within the MRC Cooperative which aims to use structural data to improve understanding of the molecular basis of disease. The decision was therefore taken to establish a generic protein family database, incorporating CATH, VIDA, EyeSite and other related database resources, using a database management system (DBMS).

The VIDA database contains a complete collection of homologous protein families derived from open reading frames from complete and partial virus genomes of particular virus families (currently herpesviruses, coronaviruses and arteriviruses). These are mostly sequence-based families that have been identified using a protocol based on the profile-based method (PSI-BLAST, Altschul et al., 1997; MKDOM, Corpet et al., 1988). This method attempts to identify domains within gene sequences using the concepts of domain recurrence to detect related domain sequences in different multidomain contexts. The forthcoming EyeSite database and other resources being developed within the MRC Cooperative will contain families derived using a similar approach based on MKDOM. Both VIDA and EyeSite contain a small proportion of domain families identified structurally by mapping to families in the CATH database. Although MKDOM attempts to identify individual domain regions within multidomain proteins, this is not always achieved as some homologous domain sequences may have diverged too far for any significant sequence similarity to be detected. Therefore some of the families in VIDA and EyeSite will contain multidomain sequences.

Whole genome data is currently being maintained within the Gene3D database (Buchan et al., 2001). This identifies CATH protein domain families within 36 completed genomes using a PSI-BLAST based protocol (DomainFinder, Pearl et al., 2001; DRange, Buchan et al., 2001). Sequence relatives for families in Gene3D are derived using a less stringent domain-boundary prediction protocol than that used to derive sequence domains for the CATH database itself.

Two factors favour a direct collaboration between CATH, VIDA and the EyeSite databases. Firstly, there is a substantial overlap between much of the core data from the three projects – notably the protein sequence and structure data, and the genomic data. However, there may be several conflicting definitions of a protein domain for a particular amino-acid sequence – the CATH definition, the Gene3D definition and/or the VIDA or EyeSite definition. Allowing users to assess the similarities and the differences between these different ways of defining protein domains is one of the core functions of the PFDB. Secondly, all three projects are institutionally related; CATH and VIDA are both based at University College London, and both the Department of Biochemistry & Molecular Biology at UCL (CATH) and Department of Crystallography at Birkbeck (EyeSite database) collaborate in a Joint Research
School in Biomolecular Sciences, in which the MRC Cooperative to establish biological databases is based.

Significant future developments of the PFDB have already been planned or anticipated. A collaboration with the Macromolecular Structure Database (MSD) (Keller et al., 1998) at the European Bioinformatics Institute (EBI) is already underway and will lead in due course to the incorporation within the PFDB of extensive, high-quality data derived from the Protein Data Bank (PDB) (Berman et al., 2000). It is also intended that the PFDB will be extended to handle microarray data (notably the virus expression data generated by Paul Kellam’s lab, Albà et al., 2001) and metabolic pathway data.

The PFDB database

**PFDB data sources**

The PFDB integrates data from a variety of different sources. The starting point is information about the protein classification schemes of the various databases (CATH, VIDA, MRC Cooperative databases). These databases provide descriptions of protein domain families together with the boundary definitions of individual domains within each family. In CATH, domains may be discontiguous with respect to the underlying amino-acid sequence and future releases will contain multi-chain domains, i.e. domains that span more than one chain of amino acids within a multi-chain protein.

The amino-acid sequence data relevant to the CATH and VIDA domains are extracted from the GenBank flat file of non-redundant proteins. For a single sequence in the GenBank non-redundant file, multiple entries are loaded into the database whenever that sequence relates to the separate chains of a protein in the PDB, or when it is attributed to more than one source organism. This information, together with any synonym identifiers (GIs, SWISS-PROT codes) for a given GenBank sequence, is extracted from the concatenated header information that precedes each sequence in the GenBank file.

The source organism information extracted from the GenBank file of non-redundant proteins is mapped into the preferred taxonomic names (both common and scientific) specified by the NCBI taxonomy database (Wheeler et al., 2000; Benson et al., 2000). For entries in the GenBank file that ultimately derive from SWISS-PROT (Bairoch & Apweiler, 2000), this mapping is achieved using the SWISS-PROT specalist file. For entries that ultimately derive from the PDB (Berman et al., 2000), the mapping is achieved using a copy of the NCBI’s PDBeast table.

Functional annotations for the various sequences in the PFDB are currently derived from a variety of publicly available resources. For example, SWISS-PROT keywords (Bairoch & Apweiler, 2000) and EC (Enzyme Classification) numbers (Bielka et al., 1992). E. Coli is one of the most widely annotated genomes, and this data is stored in the EcoCyc (Karp et al., 2000) and GenProtEC (Riley & Serres, 2000) databases. Functional data from GenProtEC can be readily extracted and is currently captured in
PFDB: A Generic Protein Family Database

the PFDB. Future collaborations with Pfam (Bateman et al., 2000) and InterPro (Apweiler et al., 2001) providing mappings between these resources and CATH will create additional functional annotations, as will links to the Gene Ontology (GO) (which includes broader functional descriptions, e.g. cellular location, phenotype as well as biochemical data) that is being modelled by the Gene Ontology Consortium (2001) using data from the yeast, worm and fly genomes.

Information on specific structural features associated with function will also be modelled. For example, information on clusters of residues within enzyme active sites that are critical to substrate binding or catalytic mechanisms. This data is currently stored within the PROCAT database (Wallace et al., 1997) and mapping between the two databases is achieved via the PDB identifier.

Information about each of the structures in the Protein Data Bank is also stored within the PFDB. Currently only a subset of the available information about a given PDB structure is stored, information that is extracted from the secondary data source PDBsum (Laskowski, 2001). However, the aim is to incorporate substantially more information as soon as clean PDB data becomes available via the Macromolecular Structure Database (MSD) (Keller et al., 1998). In addition, how the structural sequences from the PDB (which omit residues that were not resolved in the relevant crystal structure) map into their corresponding GenBank amino-acid sequences is calculated (using a sequence alignment protocol developed by Lee, personal communication) and stored outside the database, though accessible as table data using SQL functions (Reinwald & Pirahesh, 1998).

In addition to protein sequence information, the PFDB stores information about whole genomes, viral genome fragments (the latter being relevant to the VIDA viral database) and the constituent genes within these genomes. All of this data is extracted from the relevant GenBank genome files. For protein-coding genes, links to the relevant amino-acid sequence data are made by matching the GI identifiers in the GenBank genome files to those in the GenBank flat file of non-redundant proteins. The explicit mappings between genome and gene sequences are extracted from the GenBank genome files.

The interrelationship between the various data sources used by the PFDB is shown schematically in Figure 1. Integrating data from diverse sources is not a trivial task, and one that is made more difficult by inconsistencies between key primary resources, as highlighted by Karp et al. (2001). These difficulties are compounded by the inconsistencies encountered within a single data source. Another recent paper (Karp, 2001) emphasises this point with respect to GenBank entries for complete microbial genomes, but similar difficulties are routinely encountered when handling data from the PDB and other biochemical data sources.
PFDB: A Generic Protein Family Database

The logical model for the PFDB was developed using Unified Modelling Language (UML) (Booch et al., 1999). The UML diagram in Figure 2, showing part of this model, represents object classes as rectangles. Generalisation relationships (e.g. CathDomain and VidaDomain are both sub-classes of the more general class Domain) are indicated by an arrow pointing to the more general entity. Other relationships are indicated using a line annotated with numbers to indicate the cardinality of the relationship (e.g. a Species may have zero or many Genomes in the PFDB – denoted by 0..* – whereas a Genome belongs to a single Species – denoted by 1). Relationships that represent a part/whole relationship (e.g. a Gene is part of a Genome) are distinguished from other relationships by putting a diamond at the whole end of the line. Relationships may themselves be associated with classes defining their characteristics. These are indicated by a dotted line. For clarity, only one example is shown in Figure 2.

The PFDB logical model in Figure 2 centres on a number of highly-abstract generic classes - Unit, UnitRelationship, Association, Inclusion, Collection and Aggregation – from which all other classes are derived. The basis for these is discussed in the next section.
Implementing the PFDB

There is nothing in the PFDB logical model to suggest that it should be implemented as a relational database rather than an object database. Many researchers in this field prefer object databases to relational databases because they permit conceptual models of biological data to be implemented in a manner that is both natural and direct (e.g. Eilbeck et al., 1999). However, object database technology lacks the maturity of established commercial relational databases. A pilot project, which implemented a subset of the PFDB data using the object DBMS O2 (Deux, 1991), failed to achieve the level of performance required for the PFDB. Further, system support for management and maintenance of a database of reasonable size and complexity were found to be inadequate.

We have, therefore, opted for the mature relational database technology of Oracle 8i. This not only meets our performance requirements, it also has the added advantage that the same RDBMS is being used to manage data at the EBI - in particular the MSD database (Keller et al., 1998), which will in due course become a primary source of data for the PFDB.
One of the key features of the PFDB is the way relational tables are used to represent the meta-level classes in the logical model (Figure 2). The Unit and Association tables are used to represent binary relationships between biological objects of interest. The Inclusion, Collection and Aggregation tables represent relationships with additional semantics: set-subset relationships in the case of Inclusion; set membership in the case of Collection; part-whole in the case of Aggregation. The explicit representation of relationships at the meta-level has a number of advantages:

- It aids clarity by ensuring a closer correspondence between the logical model and its implementation.

- It makes it possible to ask (meta-level) queries that encompass disparate biological entities. For example, the PFDB unit table includes entries for the following biological entities: whole protein structures; protein chains; protein domains; protein domain segments; genes; genomes; each CATH Class, Architecture, Topology, as well as each VIDA domain family.

- It provides support for managing semi-structured data (Buneman, 1997) about biological entities, since the tables can directly represent a general graph structure.

- The meta-level tables can be used to store additional information about a relationship. Two prime examples are: information about the period for which a relationship is valid, which makes it possible to support historical ‘versions’ of the PFDB; and the degree of certainty with which a relationship is believed to be true, such as the position of domain boundaries.

- It facilitates the future updating of the PFDB by providing a framework for the introduction of new entities.

Built on top of the meta-level and other base tables are a set of materialised (i.e. precompiled, static) views that bring together (denormalised) data from the underlying tables – notably the internal and external identifiers – for improved performance by precomputing results. Finally, on top of the materialised views are a set of standard, dynamic views which present all the relevant attribute information about a particular entity that a typical user is likely to require.

To illustrate how the design works in practice, let us consider a single example – how data about CATH domain 2minB3 is handled in terms of the base tables, meta-level tables, materialised views and standard views. 2minB3 is the domain’s external identifier consisting of a PDB code (2min), a chain identifier (B) and a domain number (3). The domain is classified in CATH as 3.40.50.10.2.1.1, which means that 2minB3 belongs to class alpha-beta (3), has a three-layer (aba) sandwich architecture (3.40), is an example of a Rossmann fold (topology 3.40.50) and has been assigned to an homologous superfamily associated with nitrogen fixation (3.40.50.10). A schematic diagram of this example is given in Figure 3. Non-arrow lines between base tables indicate foreign key relationships between rows, while arrow lines from views or materialised views to other tables indicate rows which are referenced in a view or materialised view. For clarity, some tables referenced by domain_id_mv have been
omitted as well as a number of table columns. Also, information about segments that make up a domain is ignored in the following analysis.

**Base tables**

Five base tables – four of which are meta-tables – are used to store protein domain-related information in the PFDB:

- The *domain* table stores some basic attribute information about the domain, notably: *n_segments* (the number of segments that the domain has); and *n_chains* (the number of chains in a given domain). Note that the *domain* table is *not* used to store information about external identifiers, domain classifications, or relationships between domains and other structural entities.

- The *unit* table stores the external identifier (2minB3) external label (3 – i.e. the domain number), the type of unit (‘DO’ for domain), the external identifier type (‘CA’ for CATH), the version, and the dates between which the domain is valid. Table *unit* also has an entry for each node in the CATH hierarchy. For example, there is a unit entry of type ‘CA_T’ (for CATH topology) for the topology 3.40.50.

- The *inclusion* tables stores the relationships between adjacent levels in the classification hierarchy, for example that between the parent CATH class (3) and

![Diagram of PFDB tables and views](image-url)
the child CATH architecture (3.40). Internal, rather than external, identifiers are used in this table together with some basic typing information (in this case ‘CA_C$CA_A’ for a CATH class/architecture relationship).

- The *collection* table stores the relationship between a particular domain (2minB3) and its classification (3.40.50.10.2.1.1), the type of classification (‘CA’ for CATH), the degree of confidence in the classification, and the dates between which the domain classification is valid.

- The *aggregation* table is used to store information about the relationship between a domain and its parent protein (PDB 2min), and between a domain and its child segment(s). These relationships are known as a ‘PR$DO’ – protein/domain – and ‘DO$DS’ – domain/segment – relationship respectively. The degree of confidence in the association and the dates between which it is valid are also stored.

- The *association* table is used to record the relationship between a domain and the method used to detect its boundaries.

Since there exist both domains that consist of discontiguous segments of a protein chain (multi-segment domains) and domains that span multiple amino-acid chains within a single protein (multi-chain domains), there is a many-to-many relationship between amino-acid sequence and CATH/VIDA domain. This many-to-many relationship is naturally represented by two 1-many relationships in the aggregation table.

**Views**

There are two materialised views used to store protein domain-related information in the PFDB:

- Materialised view *domain_id_mv* maps the external identifiers for a domain – its protein identifier (2min), chain identifier (B), domain identifier (3) and its family identifier (3.40.50.10.2.1.1) – to their corresponding internal identifiers. The information is drawn from the *unit*, *aggregation*, *collection* meta-tables.

- Materialised view *cath_hierarchy_mv* maps the external identifiers from table *unit* for a particular CATH classification (e.g. 3, 40, 50, 10, etc.) into their corresponding internal identifiers from table *inclusion*. This materialised view effectively performs a join across fifteen tables.

Finally, a single dynamic view draws together all the relevant information about a CATH domain (excluding the internal identifiers): View *cath_domain_v* combines the external identifiers for individual domains (from *domain_id_mv*) and for the CATH classification (from *cath_hierarchy_mv*) together with the attributes stored in the *domain* table.
**Incremental updates**

New data is imported into the PFDB by a three-stage incremental update procedure. Firstly, data is extracted from one or more source files (see ‘PFDB data sources’ section above) using dedicated data extraction scripts. These scripts output putative PFDB data into separate files, one for each destination table. For example, the script designed to process PDB data reads in a list of PDB codes, retrieves structural data for each code from the relevant PDBsum files and outputs this data to two files – one for the unit table and one for the structure table. The output data is labelled with the appropriate column names of the destination table, taking the form

```
column1=data_item1| column2=data_item2 | column3=data_item3
```

The second stage of the update procedure concerns the merging and reconciliation of data from different sources into a single data file for a particular database table. For example, data generated by the PDB data extraction script mentioned above is merged with data generated by the script used to extract information about how PDB sequences map into GI sequences. Discrepancies between the information accumulated from different sources are identified at this stage, errors are flagged for putative entries that lack essential data (such as a primary key) and default values are inserted into blank fields where available. Information about the names of database columns, about which columns cannot be left blank (i.e. columns labelled “not null”) and about default column values are extracted from the SQL table definitions used to create the PFDB. This stage is accomplished using a single, generic data-merging script.

The final stage is the data-loading phase. The putative new data from stage two is compared with the existing data in the PFDB - retrieved using the Perl DBI (database independent interface) - and divided into three categories: completely new data that need to be inserted into the PFDB; data that are identical to existing PFDB data and can therefore be safely ignored; and modified versions of existing database entries. In the last case, the existing PFDB entry will be archived (by setting its unit table to date column to today’s date) and the new entry inserted (with its unit table from_date column set to tomorrow’s date – thereby preventing both old and new versions of the same entry from coexisting on the same date).

As with the second phase of the incremental update procedure, the final stage is largely accomplished using a single, generic script. This is made possible by the PFDB’s consistent handling of disparate biological entities and their inter-relationships using the explicit meta-level approach described above. The advantage is clear – it greatly reduces the amount of database redesigning and script writing required for the integration of new entities, attributes and data sources within the existing database framework.

**The PFDB interface**
A Web interface to the PFDB is currently being developed using the Perl DBI. A number of preformulated queries (i.e. parameterised queries with a set format) have already been written based on the needs identified by the developers of the CATH, VIDA and EyeSite databases, and on a requirements analysis carried out within the Biomolecular Structure and Modelling Unit at UCL. Questions that can be asked via the PFDB preformulated query interface include the following:

- What are the PDB codes and GenBank identifiers of sequences belonging to a particular CATH family?
- What products are associated with a particular CATH fold or family?
- What CATH folds or sequence families occur in kingdom $x$ but not in other kingdoms?
- What genomes contain at least one example of a particular CATH fold or family?

Taking as an example query “what genomes contain a particular CATH fold, namely 3.20.20 (TIM barrels)?”, the corresponding SQL query references 3 materialized views (each twice), 3 meta level tables (1 twice) and 2 ordinary base tables. The query itself runs with an elapsed time varying between 1 and 2 seconds depending on system load and extent to which database blocks are already cached. The web interface and results page for this query are illustrated in Figures 4 and 5 respectively.

![Figure 4. Example of the PFDB Preformulated Query Interface](image_url)
PFDB: A Generic Protein Family Database

Result: Genomes containing members of CATH Topology 3.20.20

Organisms = 17, Chains = 99

Archaea:
- Aeropyrum pernix: 420311
- Archaeoglobus fulgidus: 470281

Bacteria:
- Bacillus halodurans: 530346
- Bacillus subtilis: 410207
- Canis familiaris: 627376
- E. coli: 628778
- Escherichia coli: 573747, 573748, 573749, 573750, 573751, 573752
- Haemophilus influenzae: 628779, 628780, 628781
- Helicobacter pylori: 628782, 628783
- Helicobacter pylori: 555615, 555616, 555617
- N. gonorrhoeae: 555618
- Pasteurella multocida: 555619
- Pseudomonas aeruginosa: 555620
- Ricinotis proteus: 555621, 555622
- Synechocystis: 555623, 555624

Eukaryota:
- Caenorhabditis elegans: 1106210, 1106211, 1106212, 1106213, 1106214

Virus:

Figure 5. PFDB results page for the query given in Figure 4. Amino-acid sequences that have domains classified in CATH as 3.20.20 (TIM barrel) are grouped by kingdom and species. Each amino-acid sequence is identified by its PDB chain identifier (blue-grey) or GI identifier (black).

Accessing the PFDB via preformulated queries has several advantages. Preformulated queries are easy to use, they can be highly optimised to guarantee fast response times, and they prevent users from running queries that are inefficient and/or require excessive amount of CPU time. However, preformulated queries do not offer the kind of flexibility that many users desire. It is planned, therefore, that a flexible interface – one which allows users to compose ad hoc queries, but does not require a knowledge of SQL – will be developed in the near future.

Discussion

The PFDB is a new resource that aims to provide fast and flexible access to protein family-related data. It is being used to integrate and manage data for several protein-related databases – the CATH, VIDA, Gene3D and EyeSite databases.
From a database design perspective, the primary characteristic of the PFDB is that it incorporates an explicit meta-level of abstract entities. The most important benefits of this approach are that it offers a flexible framework for adding new entities into the database, and it provides a mechanism for answering complex queries about disparate biological entities.

Given the rapidity with which the bioinformatics field is developing, the ability of the PFDB to integrate new entities and relationships into the existing design with relative ease is of paramount importance. New types of data that we intend to introduce into the PFDB in the next 12 months includes: protein-protein interactions, metabolic pathways, transcriptomic data and proteomic data.

We have already experienced the benefits of our flexible, meta-level approach in our ongoing work aimed at reconciling the PFDB schema with that developed independently for the MSD database (Keller et al., 1998). MSD models structure in much greater detail than the PFDB, reflecting its concern with the detailed crystallographic structure of proteins down to the atomic level (details that lie outside the scope of the PFDB). The changes that need to be made to the PFDB schema in order to establish explicit, well-defined relationships to entities in MSD are negligible, being confined to a small number of base tables. Apart from incorporating some additional typing information, no changes to the underlying meta-tables are required.

The absence of atomic-level data from the PFDB points to another of its key characteristics. Rather than attempt to be comprehensive, the PFDB is by design selective in the data it allows users to search on, preferring high-level information to vast quantities of low-level information (such as atomic-level data). This selectivity has clear performance benefits.

We believe the design decisions taken in the construction of the PFDB have proved to be sound. The combination of meta-tables and conventional relational base tables has given us the modelling power and flexibility of a graph or binary-relational database system without the performance penalties often incurred with such systems when all data is mapped to a graph format. Similarly, we have avoided the performance limitations and weak system management facilities of many current object-oriented database systems. We believe the approach will naturally generalise to incorporate the more general graph structures of metabolic pathways and other transcriptomic and proteomic data.

Acknowledgements

This work was funded by the BBSRC/EPSRC Bioinformatics Initiative.
Bibliography


PFDB: A Generic Protein Family Database


