Chapter 2

Therapeutic target database development

This chapter describes our work in developing a publicly accessible drug target database, Therapeutic Target Database (TTD), which provides information about the known protein and nucleic acid therapeutic targets together with the targeted diseases / conditions, their pathway information and those corresponding drugs / ligands directed at each of these targets. An ontology-like database structure is devised to manage the drug target information as well as maintaining the maximum flexibility to accommodate new interests in drug mechanisms. Web interfaces built on this database structure inherits this flexibility. The work of TTD has been extended to the construction of two other drug mechanism information databases, namely Drug Adverse Reaction Database (DART) and drug Absorption Distribution Metabolism and Excretion Associated Protein database (ADME-AP).

2.1 Introduction

Pharmaceutical agents generally exert their therapeutic effects by binding to some particular protein or nucleic acid targets [89,113]. So far, hundreds of proteins and nucleic acids have been explored as therapeutic targets [89]. Rapid advances in genetic [114,115], structural [29,30] and functional [116] understandings of disease

related genes and proteins not only raise strong interest in the search for new therapeutic targets, but also promote the study of various aspects of known targets including the molecular mechanisms of their binding agents, related adverse effects [117], and pharmacogenetic implications [118], etc. The knowledge gained from such studies is important in facilitating the design of more potent, less toxic, and personalized drugs. Development of advanced computational methods for bioinformatics [119], molecular modeling [120], drug designing and pharmacokinetics analysis [54,56,111] increasingly uses known therapeutic targets and drugs to refine and test algorithms and parameters. Therefore, a database that provides comprehensive information about therapeutic targets will be helpful in catering to the needs and interests of the relevant communities in general and those unfamiliar with a specific therapeutic target in particular.

Database development is one of the major concerns in the field of bioinformatics [121]. The motivation for design and development of bioinformatics databases comes from the challenge of bridging the gap between knowledge and their efficient management (storage, retrieval and processing) in biomedical sciences. It is said that in the post-genomic area, the annotation of sequences would be a major direction of bioinformatics [121-123]. The development of specialized domain knowledge databases such as TTD can be regarded as part of this effort.

In order to provide a background for readers who are not familiar with biological databases, a brief history of bioinformatics with the focus on publicly accessible databases is briefly introduced below.

Although the term bioinformatics was first coined in the 1980s, the idea of using computers to store and manage biological data was actually initiated by X-ray protein crystallographers in the 1960's [124]. Their early work led to the establishment of the first bioinformatics database in 1971: Brookhaven National Laboratory's Protein Data Bank (PDB), a database of 3D protein structures [125].

However, the advent of what we call bioinformatics today was mainly driven not by X-ray crystallographers but by the development of improved automatic DNA sequencing technology [126,127]. Prior to these Nobel-prize winning developments, it would take a laboratory at least two months to sequence just 150 nucleotides. By the end of 1970s, it was possible to sequence around 200 bases per day. Owing to the introduction of fluorescence labeling technology [128] and multiplexed capillary electrophoresis [129-131], fully automated DNA sequencers soon appeared. Now with instruments such as the ABI 3700 or the Pharmacia Megabase 500, it is possible to sequence 500,000 bases per day on a single machine. Today, companies such as Celera, Incyte, Monsanto and others are capable of sequencing up to 100 million bases a day.

Because of these new technologies, DNA sequencing activities became heavily dependent on computer software for assembling, storing and managing DNA sequence data [132-135]. The rapid accumulation of DNA sequence data also stimulated much interest in the development of statistical methods and computer programs for analyzing DNA and protein sequences [136-139]. The need for computational tools was especially amplified with the launch of the Human Genome

Project in 1990 [140]. Beginning as a 15 year effort coordinated by the U.S. Department of Energy and the National Institutes of Health, its ultimate goal is to map sequence and identify all 30,000+ genes in the human genome. The first draft of human genome was completed on June 25, 2000 and released publicly on Feb 15, 2001 [141-143]. It is expected that a finished version of human genome will be released very soon. By then it is also expected that the genomes of many other organisms will have been sequenced. Not only has bioinformatics played a key role in handling, sorting and storing this genomic information, it is also expected to help with the new challenges ahead in inferring gene and protein functionality [122,123], which is critical in the advances of biomedical and pharmaceutical sciences.

Another stimulus for the rapid advances of bioinformatics has been the spectacular growth in computer technology [144]. It is uncannily predicted by Gordon Moore in 1965: “The processing speed of a microchip will double about every 18 months”. Today, this trend still holds true and it is know as the Moore’s Law [145]. Such rapid rate of computer hardware development has led to the creation of a thriving computer industry that delivers very high performance machines at relatively low prices. This in turn has led to the ubiquitous distribution of desktop computers, allowing easy access to computational tools among biologists and drug designers. Also, there is another significant reason for the rapid growth in computer usage among biologists and pharmaceutical researchers, which is the emergence of the “Information Superhighway” – Internet [146-148]. Originally developed in 1969 by the U.S. Department of Defense for research in communication networkings, ARPANET

[149] (as it was called then) grew from a text-only messaging system to a graphical, interactive communication medium, enabling rapid information exchange [150,151]. By 1993, Internet uses exploded with the introduction of browsers such as Mosaic and Netscape. These web browsers and their special communication language, HTML (Hypertext Markup Language) [152], have greatly facilitated the access and communication between individuals, research labs, universities and other large research organizations. Taking advantages of the information highway, centralized biological databases have been established. Dedicated bioinformatics web servers such as EXPASY (http://www.expasy.org) [153] and National Center for Biotechnology Information (NCBI) (http://www.ncbi.nlm.nih.gov) [154] heralded the establishment of the Internet as the primary means of communication among biological and pharmaceutical researchers, causing the field of bioinformatics to truly take off. So far, there are more than 500 published biological databases and this number keeps growing annually [155-157].

While many databases have been built for knowledge exchange and discovery with different focuses, a public database focusing on therapeutic target information has not yet been established. Although probably all the proteins and nucleic acids targeted by drugs are in other databases, they are not specified as therapeutic targets and it is troublesome for researchers to search for information across multiple databases. Moreover, because of the different focuses of these databases, the annotations they provide may not be so relevant to drug discovery. A database focusing on therapeutic targets are therefore needed as a basic tool in

pharmaceutical sciences to support the multi-disciplinary effort of modern drug discovery. For example, with comprehensive drug target information, *in silico* approaches may be applied to facilitate the discovery of novel therapeutic targets and therapeutic mechanisms, which are discussed later in the next two chapters.

2.2 Collection of therapeutic target information

A survey of modern drug design approaches reveals that the information on three types of molecules is of great interest to relevant communities: drug targets (proteins or nucleotides), drugs / chemicals that bind these targets and natural ligands of these targets [48,50,54,56,74,113,158].

Drug targets are the primary focus of the database. Important properties of a target include its synonyms, related diseases and pathways. Unlike small chemicals, there is no systematic naming protocol designed for macromolecules. Contemporary naming for proteins and genes are not unified. For example, prostaglandin H2 synthase, a well known therapeutic target for inflammation, is also known as cyclooxygenase [159], while the two names bear no obvious morphological similarity. The prevailing heterogeneous naming in literature makes it necessary to enforce a standardized or systematic nomenclature for drug targets. Therefore, a unique identifier needs to be assigned to each target, which is also the solution adopted by major sequence databases such as SWISS-PROT and NCBI. In TTD, the most popularly used target name is chosen for each target and other names of the protein are stored as its synonyms. The therapeutic effects achieved through the regulation

of target activity are no doubt the most important feature of a therapeutic target. To understand the therapeutic effects, this regulation of target activity shall be examined in the complex pathways in the host organisms [160]. The pathway information is therefore very useful to a variety of applications such as finding alternative therapeutic targets, designing a therapeutic intervention strategy which involves multiple co-operating targets, and analyzing potential drug-drug interactions. As introduced in Chapter 1, receptor 3D structure based approaches require the 3D structures of target molecules. In case that the 3D structure of a target has not been resolved, the primary sequence of the target may be used to derive its 3D model. Therefore, cross reference to PDB, the protein 3D structure database, and SWISS-PROT, a major protein sequence database with wealthy annotation, shall be established whenever possible.

A new drug discovery process can also start from the structural information of the small molecules that bind a certain target. In this case, a series of known binders of a target are analyzed to derive a structure activity relationship model. Information on drugs, investigational drugs, and other chemicals that have activities on a certain target is therefore very important. A target may have multiple binding sites [161-164]. Different drugs may bind to different binding sites of a target and exert different regulatory effects on the target activity. Therefore, drugs of different types may have different binding sites and shall be differentiated as their structure activity relationship may be different.

Drug binding is competitive in nature [165-169]. This binding competitiveness is

an important factor in drug design. Natural ligands of drug targets are prevailing competitors. A drug is less likely to be effective if it binds to its receptor non-competitively against the natural ligands of the receptor. Thus the information on natural ligands that can bind to the known drug targets is also desirable.

Drugs and natural ligands are generally small chemicals. Although there is a standard IUPAC (International Union of Pure and Applied Chemistry) name for every small chemical, they are not the most widely used ones. Therefore, a unique identifier is assigned for drugs and natural ligands in the database. With the help of IUPAC names, it is much easier to identify the synonyms of the same chemical. In this database, the molecular formula, molecular weight, CAS RN (Chemical Abstract Service Registration Number, an identification number given to each registered chmeical), and chemical classification of a drug or natural ligand are provided.

Because the information collected is mostly reported in experiments using different methodologies, equipment and reagents, the heterogeneous quality of data in this database requires that the references to the original information sources be provided. The citation of the literature is therefore provided wherever applicable.

With the rapid advances in new drug discovery, more and more information about explored drug targets and new drug targets are being generated. An automated literature information extraction system, if available, is desired with this consideration. However, biological literature are unstructured materials, which are considered very difficult for automated information extraction [170-172]. A survey of current literature information extraction technology showed that there are some

major obstacles in this application besides those inherited from natural language processing. First, the molecule names are difficult to recognize [173]. They are often composed of several words that also have their own meanings respectively. This makes it difficult to determine the boundary of protein names. Also, various abbreviations are used for proteins and nucleotides. The same abbreviation may or may not mean the same thing in different contexts. Previous successful attempts in automated biomedical literature information retrieval usually work in small domains and use name dictionaries to avoid the problem of recognizing protein and nucleic acid names [174]. This is not a feasible solution to our application as the work needed to construct a complete name dictionary of human proteins and nucleic acids is too heavy to be afforded. Second, pronouns are extensively used when describing complex relationships between molecules. The determination of the objects indicated becomes particularly difficult especially when there are more than one pronouns in the same sentence [175]. The third difficulty lies in the understanding of generalized terms and narrowed terms. For example, rhodopsin is a kind of G-Protein Coupled Receptor (GPCR). The description of the common characteristics of all GPCRs will also apply to rhodopsin. To address this problem, it needs the biological domain knowledge to be “hard-coded” into the information extraction system [176,177], which is an extremely difficult work. Therefore, fully automated literature information extraction methods may not be ready for this application until the above mentioned problems are sufficiently addressed. However, simple automated text retrieval programs based on key word searching are developed to

facilitate our search for therapeutic target information, which are proved to be helpful in reducing the burden of data collators.

With the help of a few automated text retrieval programs developed in PERL, we downloaded all the literature in NCBI that contains the phrase “therapeutic target” in their abstracts. Efforts have been made to manually extract information from available literature. Only those proteins and nucleic acids that had already been explored by current therapies or had been suggested explicitly by the author as potential therapeutic targets were included in the database. A total number of 433 targets were found in the literature. It has been reported that approximately 500 therapeutic targets have been exploited in the currently available medical treatments [89]. The search for therapeutic targets aims to collect as many known targets as possible. However, descriptions of some of the targets in the literature were not specific enough to point to any particular protein or nucleic acid as the targets. Hence these targets were not included in this database.

2.3 Therapeutic target database development

Before undertaking a discussion of therapeutic target database development, it is important to determine the expected system functionalities and guidelines for further designing processes. The technology platform and software tools suitable for this project shall be selected accordingly.

2.3.1 Requirement analysis

The database system is expected to store and manage the information about

known therapeutic targets. The interested types of information have been discussed in the previous section. In consideration of the fast enriching data about known drug targets or potential drug targets, a convenient updating mechanism is needed for this database. Also, a good system should not be designed to satisfy only the current needs; it should also take full consideration of the possible change and extension in future. Rational drug design is pacing fast nowadays. Different designing approaches take interests in different facets of drug targets. In the foreseeable future, more types of information will be needed by new or improved rational drug design approaches and therefore needed to be added into this database. Small changes or a complete overhaul of the database structure may be needed with this regard. No database or other software can be suitable for use forever; however, a flexible database structure that can be extended to incorporate these new interests with minimum necessary changes is desired. Moreover, the data collection work of two other drug mechanism information databases was in progress parallel to the development of TTD. As an augmented goal, it would be better that the design and implementation codes of TTD can be re-used in the development of these two databases.

Before the actual database development, it is also to be determined which technology platform and software tool are to be used to establish this database.

2.3.1.1 Databases development approaches

There are several approaches to establish a database in past bioinformatics practices. The common ones include the flat-file approach, the relational approach,

and the manual approach.

In the flat-file approach, data is organized in text files where individual records of data are represented by a set of lines in strict order with symbols that allow the computer to find and retrieve specific pieces of information [178,179]. SWISS-PROT, EMBL, GenBank and OMIM are examples of the databases using this approach.

In the relational approach, a set of tables (also known as relations) are created by a database developer to reflect the inter-relationship between the data stored in the database [180-185]. Typically database management software such as SQL Server, Access, IBM DB2, or Oracle is used to manage the querying, updating and re-structuring of the database. The relational approach is so far the most widely utilized and dominant mainstream approach to data management.

In the manual approach, the information is manually coded into static web pages. Data are organized hierarchically and can be navigated following the hyper-links from the portal page. Usually no software helping the search and management of the data is used and limited search facility is provided. This approach mainly serves databases with very small scale and highly specific scope, such as the protease inhibitor database (http://www.yorvic.york.ac.uk/~proteinase/).

Manual approach is the easiest way to create an information repository without the requirements of any specific software. However, its limitations in search facility and data maintenance are severe drawbacks for middle and large scale databases. The scale of TTD rules out the manual approach as a good choice.

The flat-file approach comes from the earliest way of exchanging biological

information -- distribution of copies of flat-files to researchers. Since then, a large number of biological repositories have emerged, and the availability of the Internet has made it possible for researchers to access them without having to install and manage local copies of the data. Due in parts to the flat-file based origins, many of the major present day biological data repositories, e.g. SWISS-PROT, are established using flat-file indexing systems. They are mainly efficient searching engines built on the concept of indexing. Information retrieval is performed by keywords, or by conjunctive or disjunctive combination of a set of keywords. Numerous interfaces [186,187] has been built to allow one to search for desired information in a collection of heterogeneous databases.

The advantage of such flat-file based systems is self-explanatory content that is optimized for human readability. The incorporation of hyperlinks into such records further allows for extensive cross-referencing. However, the flat-file approach is relying on mere text-matching indexing. It shows significant drawbacks in comparison to the more comprehensive relational approach, which provides a number of desired capabilities for complex queries and data maintenance with the support of a relational database management system (RDBMS):

1. Complex query support is limited. Present flat-file based repositories offer HTML forms that accept search terms as input. Search engines parse indexes of keyword to find matches, and retrieve matching records. A more elaborate search form allows the specification of field specific terms and Boolean combination of different search terms. It is important to note that, in the majority of cases, queries

are limited to the text matching approach. There is no support for complex queries with analytical requirements or nested queries such as “what are the targets unique to liver?” This kind of comprehensive listing of organ-wise unique targets has the potential to minimize drug side effects. In the contemporary relational databases, Structured Query Language (SQL) [188,189], an industrial standard, is supported almost unanimously, which could be used to construct virtually all kinds query logic in order to get comprehensive and specific results. For instance, the above mentioned question may be answered by a SQL query like the following:

```
SELECT DISTINCT [Target Name]
FROM [Target Table]
WHERE [Organ] = ‘liver’
      AND [Target Name] NOT IN
                (SELECT DISTINCT [Target Name] FROM [Target Table]
                  WHERE [Organ] <> ‘liver’);
```

This SQL statement contains nested query which is not supported by indexing engines such as the one in SWISS-PROT.

2. Data maintenance is tedious and difficult. A flat-file indexing system only provides the functionality of a search engine with no support for data maintenance. For example, when updating the content of a flat file, there is no constraint that can be enforced in normal text-editors to check whether reasonable data have been entered in the correct format. For example, one might enter “30-Feb-1997” in a text editor as a date by mistake. This obvious error could only be identified by a thorough check of the data later. However, A RDBMS will refuse to accept this kind of “illegal”

data if appropriate constraints have been enforced. This will significantly reduce the error rate in data preparation. Also, any updates in the data files will not automatically take effect in the flat-file indexing system. It usually requires a re-indexing process to make the changes effective. For big repositories like SWISS-PROT and TREMBLE, a complete re-indexing would take more than 10 hours. However in RDBMS, efficient algorithms are implanted to keep indices up to date on the fly while you are modifying the data.

3. Views cannot be decoupled from underlying data. In other words, records are always retrieved in entirety. This will reduce the query performance due to the operations wasted in retrieving irrelevant data. In RDBMS, SQL is able to control which information is needed by defining appropriate views upon the tables. An RDBMS is able to carry out the queries optimally according to the user defined SQL.

In view of these advantages of the relational approach over the others, it is considered to be the best one for the development of TTD

2.3.1.2 Selection of RDBMS

The relational approach requires the support of an RDBMS. The RDBMS enables users to create and maintain a relational database. They are designed to control data redundancy, restrict unauthorized access, provide persistent storage for program objects and data structures, permit inference and actions using rules, provide multiple user interfaces, represent complex relationships among data, enforce integrity constraints, and provide backup and recovery supports [92,190].

Although the direct objective is to develop a therapeutic target database, the

selection of the RDBMS should take further considerations for the subsequent database projects and the data analysis requirements by other projects of our group. Specifically, a desired RDBMS should be able not only to support several small databases, but also to hold local copies of existing major public biological databases and provide integrated analytical query ability. Therefore, a high performance large scale RDBMS is needed.

As illustrated in Figure 2.1 (May 2001 IDC report on 2000 RDBMSs), the market share of RDBMSs is: Oracle (Oracle 9i) 46%, IBM DB2 (DB2 UDB 7.2) 24%, Microsoft SQL Server (SQL Server 2000) 7% and others 23%. For better compatibility with other existing applications, we decided to choose a RDBMS from the best selling ones, namely Oracle 9i, DB2 UDB 7.2 and SQL Server 2000.

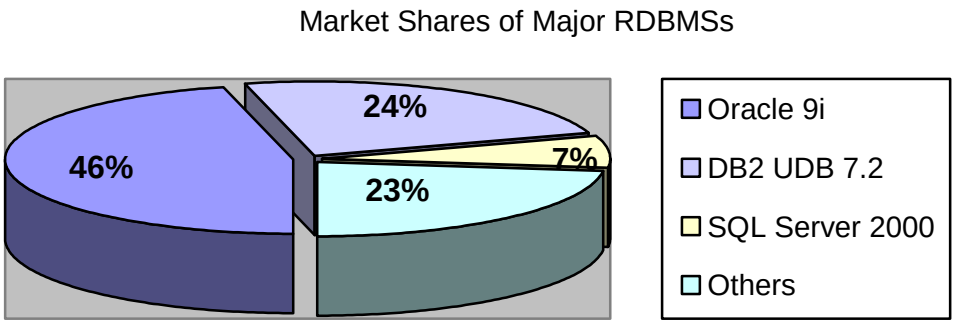


Figure 2.1: Market shares of major RDBMSs. Based on May 2001, IDC report on 2000 RDBMSs

The factors affecting the selection of RDBMS include: platform and system requirements, supported data types, program language supporting, application

development features, manageability features, security features, analysis ability, internet ability, price and performance.

After a careful technical assessment of the major RDBMSs, Oracle 9i is selected. It is clear that SQL Server 2000 was the least expensive of the three. However, it could only run on the Windows platform, which limited its performance. DB2 UDB 7 had just moved from its main frame to Client/Server based database market. Similar to Oracle 9i, it could run on many operating systems, and its data types were best compatible with ANSI (American National Standards Institute) SQL definitions. However, Oracle 9i had the most variety of modules and development tools, including modules for data mining and online analytical process, which is essential for high-end data analysis purposes. For many years, it had led the way in indexing and query optimization technologies while it is not worse than its competitors in other important aspects. Also, Oracle 9i is a fully object-oriented database, which conforms to the trend toward Object Oriented Programming (OOP). Oracle had kept the biggest market share for years. According to the 2001 statistics, over half of the fortune 100 corporations used Oracle as their database servers. And finally, it was said that the price for a full featured Oracle RDBMS was comparable to that of a full featured DB2 RDBMS. Therefore, Oracle 9i was selected to be the platform for our database projects and other data analysis tasks.

After the determination of approach and software platform, the actual database design begins.

2.3.2 Database design

The database designing process can and should be divided into three phases: conceptual design, logical structure design and physical design, as given in the widely accepted Information Engineering Methodology (IEM) [191,192]. The conceptual design phase consists of defining the types of information to be stored in the database and documenting them. The logical design phase consists of putting the conceptual design into practice in the software of choice by creating data tables and the relationships between them. Physical design phase allows the designer to determine how the data is to be stored on the magnetic media of a computer.

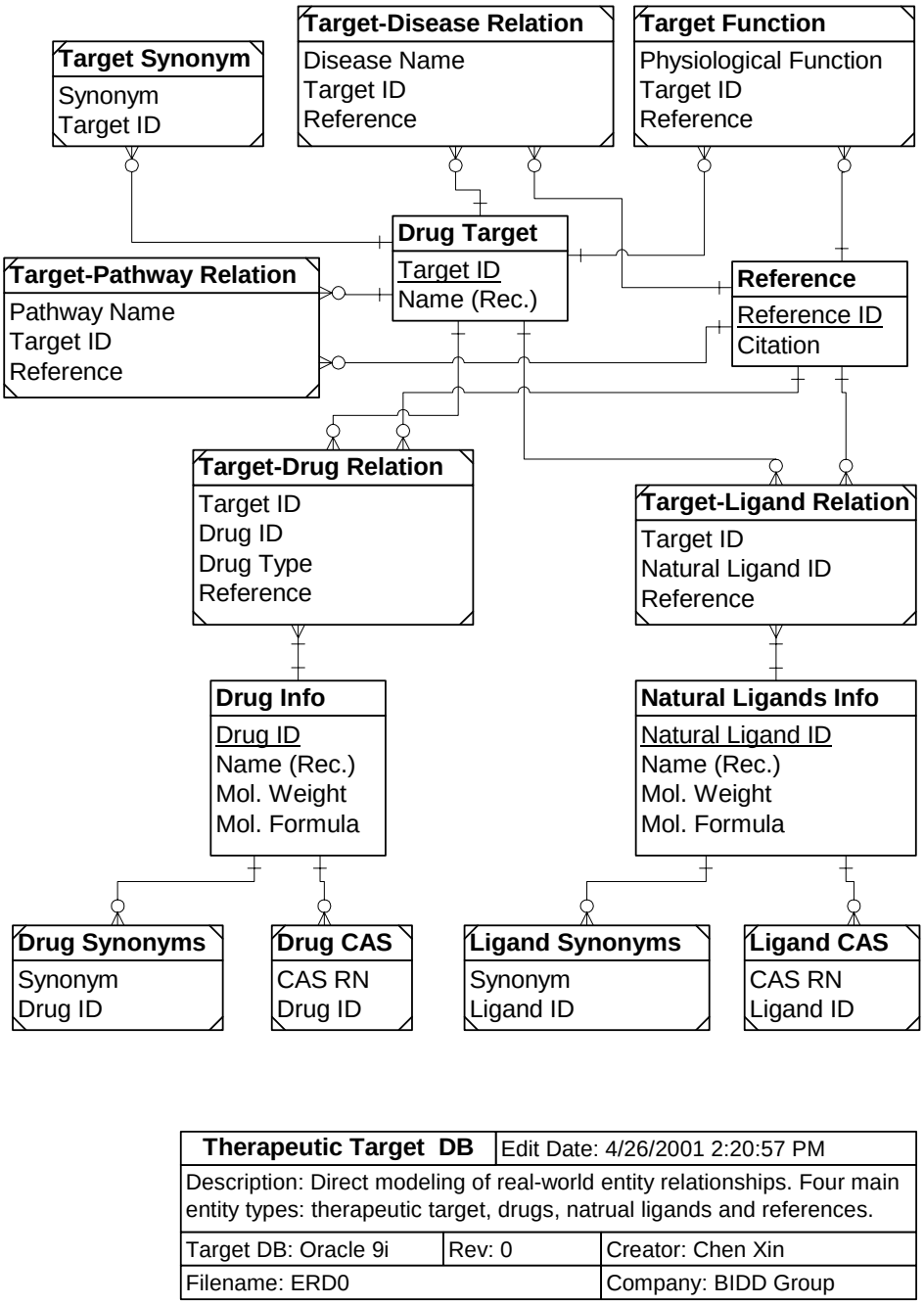
2.3.2.1 Conceptual design

The conceptual design phase is a “high-level” phase of design and is independent of the choice of database management systems. The result of this designing phase is a set of documentation diagrams, whose purpose is to create discussion and understanding of the database design before the implementation work begins.

The design documentation that depicts the semantic relationship between data is called the entity relationships diagram (ERD) [193,194]. This data modeling technique breaks data types down into entity types, attributes and relationship types. An entity type is a collection of entities that share common properties or characteristics. The properties or characteristics of an entity type or relationship type that are of interest to the organization are called attributes. Relationship types are meaningful associations between / among entity types. There are three categories of relationship types: one-one relationships, one-many relationships and many-many

relationships.

A natural way of drawing the entity relationship diagram is to take target molecules, drugs, ligands, and references as entity types and their respective properties as their attributes while relationships are established accordingly. For instance, the entity type target molecule will have attributes including a unique ID, synonyms, cross references, related diseases, functions, pathways involved. And they are linked to natural ligands and different types of their drugs by many-many relationships. However, this simple method is not applicable because it may result in potential multiple-valued attributes which violates the first norm in database design and will cause problem in query and data maintenance [195,196]. For instance, the attribute “synonyms” may have multiple values for one target. It would be hard for applications and SQL queries to distinguish different synonyms within the same data item. To address this problem, these “attributes” that might have multiple values are “promoted” to weak entity types. A weak entity is an entity of which its existence depends on the relation with another entity (the identifying entity). For example, the identifying entity type for a weak entity of “target synonym” is an entity of “target molecule”. The existence of “target synonyms” depends on their relationship to a “target molecule”. A weak entity has no key attribute because it cannot exist without the relation it has to its identifying entity. The resulted first draft of ERD (using "Crow's Foot" notation, which is the accepted IEM convention) is shown in Figure 2.2. To have a clearer view of the attributes of relationships, in this ERD, any relationship with attributes is drawn as weak entities identified by both sides of the relationship.



Therapeutic Target DB		Edit Date: 4/26/2001 2:20:57 PM
Description: Direct modeling of real-world entity relationships. Four main entity types: therapeutic target, drugs, natrual ligands and references.		
Target DB: Oracle 9i	Rev: 0	Creator: Chen Xin
Filename: ERD0		Company: BIDD Group

Figure 2.2: The first Entity Relationship Diagram of TTD.

2.3.2.2 Logical design

In the logical structure design phase, the structure of the database (also known as database schema) is created, which is a set of data tables and their connections. The data tables and the connections between them can be directly derived from the ERD. However, this approach does not, by itself, assure a good relational database for every purpose. The first design of TTD schema was derived from the above ERD, which was evaluated on a small set of sample data and disclosed several problems about this design. Accordingly, a series of modifications on the first schema is carried to introduce more unification on the data representation and flexibility to accommodate extensions. The latest revised schema resembles a semantic network in ontology research very much. These designing processes are detailed as follows.

2.3.2.2.1 ERD derived database structure

The algorithm for translating a sound conceptual design into a relational data structure is given in [197,198] as the following four steps:

First, construct a table for each entity type, containing all the attributes of the entity type and having a primary key or a unique identifier filed.

Second, construct a table for each many-many relationship type containing the unique identifier for each side of the relationship along with the attributes of the relationship.

Third, for each one-many relationship type, add the unique identifier from the “one” side to the table corresponding to the entity on the “many” side, along with all the attributes of the relationship.

Finally, for each one-one relationship type, add the unique identifier from either

side to the table for the other side, along with the attributes of the relationship.

Using this algorithm, an initial version of therapeutic target database can be constructed, which consists of sixteen tables, as shown in Table 2.1. One table is created for the target molecule entity type, which includes two columns -- the unique identifier assigned to each target and its recommended name. Those types of information with one-many relationships to a target, such as synonyms, related diseases / conditions, functions, pathways involved, are stored in their respective tables with the unique identifiers of their identifying targets. The many-many relationship between target molecules and their different types of drugs are stored in one table, which contains the unique identifiers of targets, the unique identifiers of drugs that bind the targets, and the types of the drugs. The many-many relationship between target molecules and their natural ligands are also stored in one table structurally similar to the table of target-drug relationship. To provide the information sources for data quality assessments, the tables storing information collected from literature, such as the tables for target functions, target pathways, different types of drugs and ligands, all contain one column storing the unique identifier of corresponding references. For each drug, a unique identifier was assigned and it was stored in one table with the recommended name of that drug, its molecular weight and its molecular formula. Those types of information with one-many relationships to a drug, such as drug synonyms, CAS RN, and chemical classification, are stored in their respective tables with the unique identifier of their corresponding drugs. In this design, CAS RN is treated as an entity type with

one-many relationship to a drug, which is different from most of the current available chemical databases, such as ACX and the Merck index, which usually provide only one CAS RN for each chemical. This is because while in most of the cases, a chemical only has one CAS RN and vice versa, a small number of exceptions do exist. For example, GTP (guanosine triphosphate) has two CAS RNs: 86-01-1 and 56001-37-7. Drug and natural ligands are all small chemicals that have the same types of information. Therefore, the data tables created for natural ligands are similar to those for drugs. The table storing information about references is relatively simple. It only contains two columns of unique identifier and reference citation.

Table 2.1: The data tables created in the first design of TTD.

Table Name	Columns	Table Relationships
TTDTG: Drug targets	TID: Target unique identifier NM: Recommended target name	
TTDDG: Drugs	DID: Drug unique identifier NM: Recommended drug name MW: Molecular weight MF: Molecular formula	
TTDLG: Natural Ligands	LID: Natural Ligand unique identifier NM: Recommended ligand name MW: Molecular weight MF: Molecular formula	
TTDRF: References	RFID: Reference unique identifier RF: Reference citation	

TTDTS Target synonyms	TID: Target unique identifier SN: Target synonym	TID ∈ TTDTG.TID
TTDTP: Target involved pathways	TID: Target unique identifier PW: Pathway name RF: Reference unique identifier	TID ∈ TTDTG.TID RF ∈ TTDRF.RFID
TTDTD: Target related diseases	TID: Target unique identifier DN: Disease name RF: Reference unique identifier	TID ∈ TTDTG.TID RF ∈ TTDRF.RFID
TTDTF: Target functions	TID: Target unique identifier FN: Target function RF: Reference unique identifier	TID ∈ TTDTG.TID RF ∈ TTDRF.RFID
TTDTDG: Drugs that bind targets	TID: Target unique identifier DG: Drug ID TP: Drug category RF: Reference unique identifier	TID ∈ TTDTG.TID RF ∈ TTDRF.RFID DG ∈ TTDDG.DID
TTDTLG: Natural ligands that bind targets	TID: Target unique identifier LG: Natural ligand ID RF: Reference unique identifier	TID ∈ TTDTG.TID RF ∈ TTDRF.RFID LG ∈ TTDLG.LID
TTDDS: Drug synonyms	DID: Drug unique identifier SN: Drug synonym	DID ∈ TTDDG.DID
TTDDR: Drug CAS RN	DID: Drug unique identifier CAS: CAS Registration Number	DID ∈ TTDDG.DID
TTDDC: Drug chemical classification	DID: Drug unique identifier DC: Drug classification	DID ∈ TTDDG.DID
TTDLS: Ligands synonyms	LID: Natural ligand unique identifier SN: Natural ligand synonym	LID ∈ TTDLG.LID
TTDLR: Ligand CAS RN	LID: Natural ligand unique identifier CAS: CAS Registration Number	LID ∈ TTDLG.LID

TTDLC: Ligand chemical classification	LID: Natural ligand unique identifier LC: Natural ligand classification	LID ∈ TTDLG.LID

After the tables and their relationship (constraints) were created, a web interface of the database was sketchily implemented and the system was analyzed on a small set of testing data. The above data structures showed several weaknesses in the test run. They can be summarized as below:

First, the information retrieval of a single target involves many tables. It is not a big issue when the tables are queried or analyzed by SQL statements. However, when building the web interface for the database, each table needs a distinct record set object in the web server, which will lead to inefficient use of web server resources. Also, the big number of objects requires a corresponding size of codes to manipulate them, which makes the debug and maintenance of these codes troublesome.

Second, the interface codes developed on this structure are fairly rigid to accommodate extension of the database. For example, in the later designing stages, it was suggested that it would be better to classify the therapeutic targets into different categories according to its related diseases / conditions, in order to facilitate systematical studies. Accordingly, the classification information for therapeutic targets should be added into the database. With the rapid progress in rational drug design approaches, it is foreseeable that more types of information will be needed by novel or improved methods. The ability to accommodate these extensions with minimal changes in the database structure and all its application codes is desired. In

this design, in order to incorporate the classification information, a new table storing the unique identifier of targets and their classes needed to be added into the database. Big changes on the interfacing codes were required to use the new table. Actually, not only the web interface but also all the database applications written in programming languages requiring extensive coding (such as C/C++) will suffer from this problem. The expandability and life cycle of this database are therefore considerably compromised. Also the database structure and interface codes, developed in this manner, are unable to be re-used in the development of other databases.

2.3.2.2.2 Revised database structure

A careful analysis of the first design revealed possible directions to address the above problems. First, in the above data structure, most of the tables were quite similar in terms of the data types stored in them. Many of them consisted of three columns: a column for the unique identifier of an entity type (target, drug or ligand), a column for a property of the entity type (i.e. natural ligands of targets or synonyms of drugs) and a column for the corresponding unique identifier of a reference citation. This similarity provides the possibility of reducing the number of tables by merging similar ones. Second, in order to minimize the effort needed when new types of information were to be added into the database, the applications shall be able to “notice” the changes in the database and adapt to the changes automatically. One possible solution would be providing the “database structure” information in one of the tables.

The above considerations led to the revised ERD shown in Figure 2.3. Each kind of relationship type, regardless of its type (one-one, one-many or many-many), are given a unique identifier. The unique identifiers of different relationships and their descriptions are stored in one table so that applications will be able to access the types of relationships stored in the database. The relationships are divided into two categories. One category of relationships links an entity to its attributes, regardless of single-valued attributes or multi-valued attributes. The other category of relationships links two entities together, where the other entity is represented by its unique identifier. All the information in this database can therefore be represented in a quadruple notation and stored uniformly. The quadruple notation includes the unique identifier of an entity, the unique identifier of a relationship, the right hand side of the relationship (either a property value or a unique identifier of another entity, according to the relationship category) and the unique identifier of the reference where this piece of information comes from. For example, “prostaglandin H2 synthase is a therapeutic target for treating inflammation” will be stored in the database in the quadruple form:

(“TTT0000600”, “D105”, “Inflammation”, “10878289”)

where “TTT0000600” is the unique identifier for the target prostaglandin H2 synthase; “D105” is the unique identifier for the relationship between targets and their related diseases; “10878289” is the unique identifier of the reference where this information is extracted. Similarly, “prostaglandin H2 synthase has an inhibitor, aspirin” will be stored in the database in the quadruple form:

(“TTT0000600”, “D141”, “TTD0000354”, “8728890”)

where “D141” is the unique identifier for the relationship between targets and their inhibitors, “TTD0000354” is the unique identifier of the drug aspirin and “8728890” is the unique identifier of the reference where this information is extracted. In this revised schema, there are only three tables and applications can access the “logical structure” of the database by reading the information in the relationships table. Also, to support the continuous work of data maintenance, a housekeeping field storing the latest update time of each piece of information is added. This revised schema is shown in Table 2.2

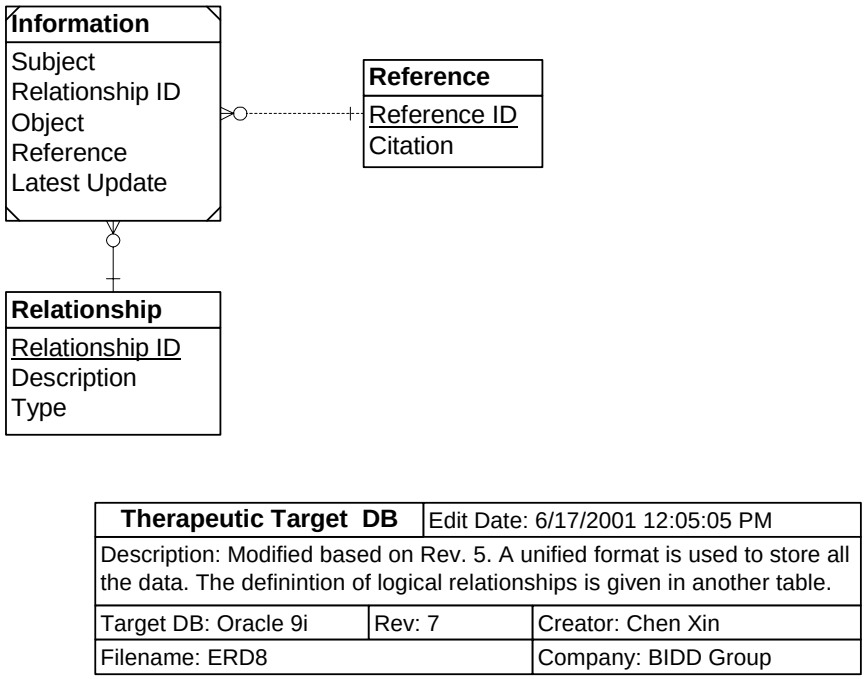


Figure 2.3: Revised Entity Relationship Digram of TTD.

Table 2.2: The data tables created in the revised design of TTD.

Table Name	Columns	Table Relationships
TTDINFO: Information	ID: Unique identifier of an entity TP: Unique identifier of a relationship CT: Property value RF: Unique identifier of the reference UPD: Latest update time	TP ∈ TTDTP.TP RF ∈ TTDRF.RFID
TTDRF: References	RFID: Reference unique identifier RF: Reference citation	
TTDTP: Relationships	TP: Unique identifier of a relationship NM: Relationship name MD: Relationship category.	

2.3.2.2.3 Further analysis of the revised database structure

This quadruple notation is applicable not only to this database but also to the other drug mechanism databases that were developed in parallel, which enabled the reuse of the TTD database schema and interface codes in the development of these databases. This universal applicability of this schema may be partly explained by its resemblance to semantic networks that are used in ontology research.

Ontologies are frameworks developed principally by the AI (Artificial Intelligence) community in the 1970s and 1980s to represent the key concepts in any research field and their inter-relationships [199]. Many bioinformatics specialists believe that they are necessary not only to make database annotations accessible to analysis tools, but also to facilitate information retrieval. For example, searching for all G-protein-coupled receptors in a database would be easier with software that “knows” that these proteins might be variously annotated as “transmembrane

protein", "7TM protein", "GPCR" or "opsin". In the 1960s and 1970s, the artificial intelligence (AI) community developed several systems to embody this sort of complexity. One such system, the semantic networks, represents concepts as nodes in a graph, which are joined by arcs that specify their relationships [200-202]. Semantic network is the approach taken by many bioinformatics ontologies. It is hoped that by building these lattices of semantic associations and by hooking database entries to the appropriate points, ontologies can be used to resolve the problem of database integration.

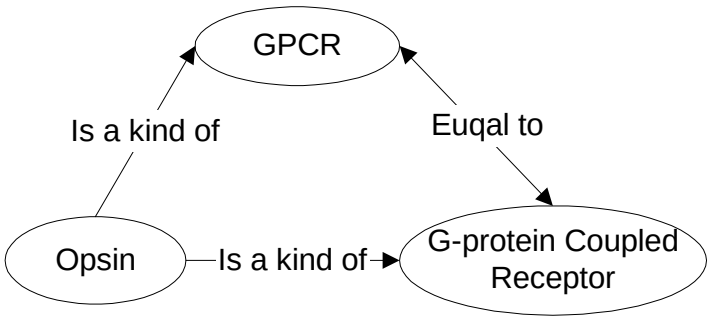


Figure 2.4: An example of semantic networks.

When a semantic network notation is devised, it is necessary to specify not only the types of the node and arcs, as well as the ways they can be combined, but also their meanings. In the above example, there are three issues. First, synonyms are used to describe the same entity (for example, "GPCR" and "G-protein-coupled receptor"). Second, relationships are required to group related concepts together (for example, "opsin" is a kind of "GPCR"). Third, there are subtle differences between terms (for example, "transmembrane" describes the location of a protein that might or might not be a GPCR). Unless the meanings of the relationships are specified

precisely, the semantic network is meaningless [203]. An example of a semantic network is shown in Figure 2.4.

The revised data structure could be regarded as an extended semantic network. In this sense, the quadruple notation could be interpreted as if two nodes were linked by an arc with the reference. The exact meanings of the arcs (relationships) are stored in the relationships table. Ontologies are built for the integration of databases. Therefore, this ontology-like data structure is also expected to enable easy integration with other databases and maximizes the re-usability of the application codes developed on this data structure.

Normalization is also a very important issue in database design [195,198]. The following normal forms (NFs) have been defined: 1NF, 2NF, 3NF, BCNF, 4NF, and PJ/NF(5NF) [204]. Normalization theory is simply a formalization of the “one fact in one place” principle of good design. If a database schema satisfies a specific set of rules, it is said to be in some normal form. Normal forms are thus specific sets of rules. Each higher normal form includes all the rules of all lower normal forms. For example, a table in the third normal form satisfies all requirements for 1NF, 2NF and 3NF. The normal form requirements up to 3NF are listed below:

1NF: This normal form specified the granularities of data. It requires no relations within attributes, no composite attributes, no multi-valued attributes and no nested relations.

2NF: In this normal form, all the requirements of 1NF shall be satisfied besides every non-prime attribute is fully functionally dependent on the key.

3NF: In this normal form, all the requirements of 2NF shall be satisfied besides no non-prime attribute can be determined by another non-prime attribute.

Normalization may be an abused principle. Excessive normalization will lead to a poor performance of the database. Thus normalization should be carried out cautiously [204]. In general, the third normal form (3NF) is regarded sufficient for therapeutic target database and the revised schema conforms to the third normal form.

One problem that the revised data structure might face is that the referential integrity could no longer be enforced by simply adding foreign key constraints on relevant columns. However, this problem could be solved by setting up triggers to check the referential integrity in Oracle 9i.

The revised schema and the first ERD derived schema are mutually convertible. In a sense, the revised data structure could be viewed as each table in the first schema was attached by a tag of its identity and stored together. To convert data from the revised schema to the first ERD derived schema, one only needs to group the records according to their relationship type column (TP) and store different groups of records in different tables. To convert data from the first ERD derived schema to the revised schema, one just needs to add the relationship type information into the each table and then store them centrally. During the development of the database, a script was written to do these conversions automatically when needed.

2.3.2.3 Physical design

After the data structure was fixed, how the data would be stored on the magnetic media of the computer needed to be determined. As most of the modern database management systems, Oracle 9i can take care of this problem for the user. This designing phase is mentioned here as a distinct section only for the sake of completeness.

After all the above designing activities, the database can be then implemented and the maintenance work will keep on going all across the life cycle of the database.

2.3.3 Implementation

There are two parts of work needed to be done to establish a publicly accessible database. One is on the RDBMS side and the other is on the web interface side.

According to the above revised design, tables were set up in the RDBMS and the referential integrity was enforced by setting up triggers. A package was created to collect all the PL/SQL programs written for this database, including scripts to load data, check referential integrity, convert data formats, do housekeeping work as well as support the functionalities of the web interface [92,190].

Various technologies can be used to build the web interface which creates dynamic web presentations according to a viewer's interest. Common techniques include JSP (Java Server Pages) [205], PHP (Personal Home Page) [206], ASP [207], and Perl (Practical Extraction and Reporting Language) DBI (DataBase Interface) [208] based applications. Among them, ASP has a big advantage on its ease of use. It supports ADO (ActiveX Data Objects) objects [209], ODBC (Open

DataBase Connectivity) [210] and OLE-DB (Object Linking and Embedding DataBase) [211] which make the development of web interface for databases much easier. Although in terms of run-time resource usage, ASP is not as good as JSP, it is still better than traditional CGI (Common Gateway Interface) [212] based approaches such as Perl DBI based applications. It is said that ASP technology is very good for middle-range applications, which fits TTD very well.

Using the ASP technology, the interface of TTD was developed, which has a URL at http://xin.cz3.nus.edu.sg/Group/ttd/ttd.asp. The portal page is shown in Figure 2.5. TTD is searchable by target name or drug/ligand name. It can also be accessed by selection of disease name, drug/ligand function, or drug therapeutic classification from the list provided in the corresponding selection field. Searches involving any combination of these five search or selection fields are also supported. Each search or selection field in this page will match one or more types of information in the database. For example, if any of the “recommended name” or “synonyms” of a target matches the term specified in the “target name” filed, this target is considered a hit.

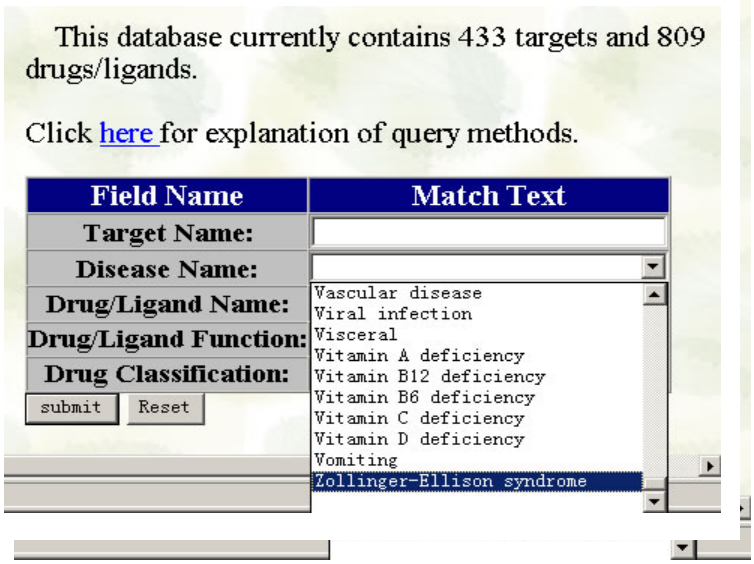


Figure 2.5: The portal page of TTD web interface

The search is case insensitive. In a query, a user can specify full name or any part of the name in a text field, or choose one item from a selection field. Wild characters of "*" and "?" are supported in the text field. "?" represents any one character and "*" represents a string of characters of any length. For example, input of "phosphatase" in the target name field finds entries containing "phosphatase" in their name, such as Cdc25A phosphatase or tyrosine phosphatase. On the other hand, input of "Cdc25? phosphatase" finds entries with names like Cdc25A phosphatase, Cdc25B phosphatase and Cdc25C phosphatase. Likewise, input of "Cdc* phosphatase" finds the same entries as above. In this case, "*" represents "25A", "25B" or "25C". "*" and "?" are not the wild characters used in SQL, therefore, all the terms are pre-processed so that they can be correctly interpreted.

The query conditions are persevered throughout a query session by cookies. The result of a typical search is illustrated in Figure 2.6. SQL query statements were

dynamically constructed to pick out the summary information of the targets satisfying the criteria specified in the first page. In this page, all the therapeutic targets found are listed along with the disease conditions to be treated, drugs or ligands that bind the target, and its classification. This summary information is generated automatically by a PL/SQL scripts with parameters specifying which types of information shall be included. ASP codes for this page were written in a manner that they are able to automatically adapt to different types of summary information. Detailed information of a target can be obtained by clicking the corresponding target name.

The interface displaying the detailed information of a target is shown in Figure 2.7. ASP codes for the generation of this page read the relationships type table TTDTP in the database and display all types of information about the target currently available. From the page shown, one finds target name, corresponding disease condition and cross-link to Karolinska disease database (http://www.kib.ki.se/), target function, pathway, corresponding natural ligand, known drugs or ligands directed at the target, drug type (such as inhibitor, antagonist, and blocker etc.), drug therapeutic classification, and additional cross-links to other databases that provide useful information about the target.

For Detailed Information, Please click on the Target Name

Target Name	Disease	Drug/Ligands	Drug Class
Histamine receptor 2 (H2 receptor)	Peptic ulcer, Zollinger-Ellison syndrome	Ranitidine, Cimetidine, Famotidine	Anti-gastric secretion
H+,K+-ATPase (Proton pump)	Zollinger-Ellison syndrome, Peptic ulcer	Omeprazole, Lansoprazole	Anti-gastric secretion

First Prev Next End Page : 1 of 1

Figure 2.6: The TTD web interface of a search result

Target Name	H+,K+-ATPase (Proton pump)
Disease / Condition	Zollinger-Ellison syndrome, Peptic ulcer Related links (Karolinska)
Drug / Ligand	Omeprazole, Lansoprazole
Drug Function	Inhibitor
Drug Classification	Anti-gastric secretion
Natural Ligand	ATP |H+ |H2O |K+
Natural Ligand CAS Number	56-65-5; 1476-84-2; 2964-07-0; |12408-02-5 |7732-18-5;13670-17-2; 13768-40 |
Natural Ligand Function	Important metabolic coenzyme; fundamental role in biological energy transformations. Used in the treatment of supraventricular tachycardia| |Commonest solv. Commercially available in purified form |
Location in Pathway	ATP + H2O + H+(in) + K+(out) = ADP + Orthophosphate + H+(out) + K+(in)
Target Properties	Brief Description (OMD) Protein Sequece and Other Info (SwissProt) 3D Structure (PDB) Related Literatures (PubMed) Ligand Binding Properties (CLiBE) Enzyme Nomenclature

Figure 2.7: The TTD web interface of the detailed information of a target

2.4 Preliminary analysis of TTD

A total number of 433 protein and nucleic acid targets were collected in TTD. As shown in Figure 2.8, two major classes of molecules contribute to more then three quarters of the total therapeutic targets, which are enzymes (44%) and receptors (33%). Other significant classes of therapeutic targets include hormones and factors (10%), ion channels (4%) and nucleotides (3%). This composition is generally in agreement with that reported in 1997[213]. These targets cover 125 different diseases / conditions, and 809 distinct drugs / ligands directed at these targets are collected in this database.

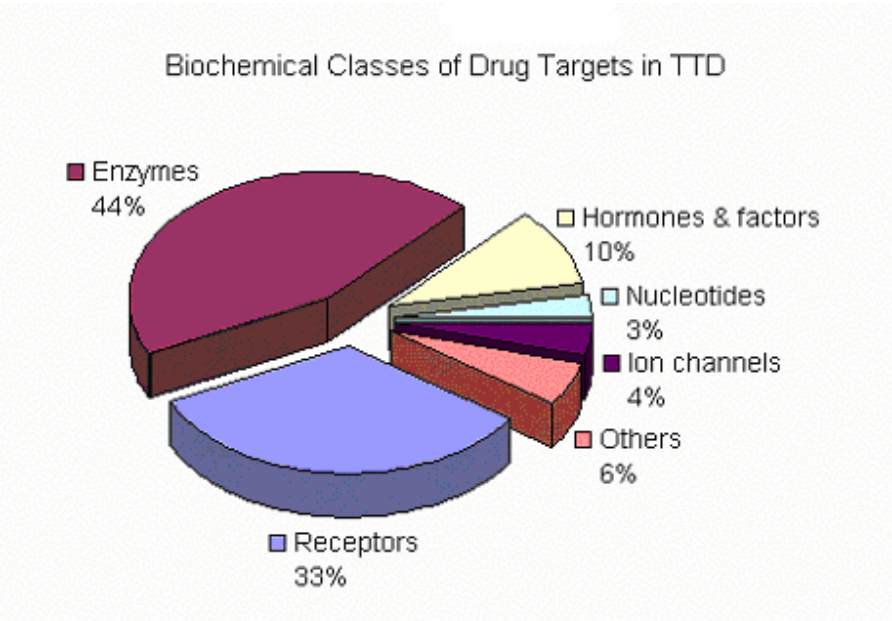


Figure 2.8: Biochemical classes of drug targets in TTD

2.5 Extension of the TTD database schema and interface

The information on drug adverse reaction and drug Absorption Distribution

Metabolism and Excretion (ADME) associated proteins are collected in parallel to the development of TTD. These are also very important aspects that affect the success of a drug.

While developing the database schema and web interface for the therapeutic target database, attentions have been paid to develop re-usable modules. By reading the relationship table, the ontology like database schema of TTD and its interface codes can virtually adapt to any predefined sets of information types, including those needed by drug adverse reaction information and drug ADME associated proteins information. Therefore, the work of TTD is readily extendable to the development of these two drug mechanism databases, namely the Drug Adverse Reaction Database (DART) [214] and drug ADME associated protein database (ADME-AP) [215].

Drug adverse reaction is often induced by the interaction of a drug or its metabolites with specific protein targets related to toxicity or side effects [117,216-219]. Knowledge about these targets is both important in facilitating the study of the mechanism of drug adverse reaction and in new drug discovery. It is also useful in the development and testing of rational drug design and safety evaluation tools [220-223]. The Drug Adverse Reaction Database (DART) is intended to provide comprehensive information about toxicity and side effect targets to the relevant communities. DART contains information about known toxicity and side effect related proteins described in the literature together with physiological function of each target, related diseases, corresponding agonists / antagonists /

activators / inhibitors, IC50 values of the inhibitors, and the toxic effect or side effect resulting from the binding of a drug. Cross-links to other databases are also introduced to facilitate the access of information about the sequence, 3D structure, function, and nomenclature of each target along with drug/ligand binding properties, and related literature. Each entry can be retrieved through multiple methods including target name, target physiological function, toxicity or side effect, ligand name, and biological pathways. This database can be accessed at http://xin.cz3.nus.edu.sg/group/dart/dart.asp.

Drug absorption, distribution, metabolism and excretion are the processes prior to and after drug-target interaction. It often involves interaction of a drug with specific proteins [224-229]. Knowledge about these ADME-related protein targets is important in facilitating the study of the mechanism of drug transportation, disposition as well as therapeutic action. It is also useful in the development and testing of rational drug design and pharmacokinetics prediction tools [225,230-235]. The ADME associated protein database ADME-AP is intended to provide information about proteins acting as ADME targets described in the literature. This database gives description about physiological function of each target, membrane location and tissue distribution, transport direction, driving force, substrates that bind to a target, pharmacokinetic effect in terms of ADME classification, synonyms and gene name. Cross-links to other databases are also provided to facilitate the access of information about the sequence, 3D structure, function, genetic disorder and nomenclature of each target along with drug/ligand binding properties, and related

literature. Each entry can be retrieved through multiple methods including target name, ADME class, ligand/substrate name, and target physiological function. This database can be accessed at http://xin.cz3.nus.edu.sg/group/admet/admet.asp.

2.6 Summary

Therapeutic target database is developed from information in available literature, which is a result of collective and persistent effort over the years. It integrates the general information of therapeutic targets such as physiological functions and their therapeutic related aspects. With the rapid development of proteomics [95,236] and pathway analysis [237], the relevant information can be incorporated or the corresponding databases can be cross-linked to TTD to provide more comprehensive information about the drug targets and their relationship to other biomolecules and cellular processes.

An ontology-like database schema is designed for TTD which can easily incorporate new interests in therapeutic targets. Interface codes developed on this schema are also highly flexible. The work in TTD has been extended to develop two other drug mechanism information databases – DART and ADME-AP.

The completion of TTD not only provides a convenient way of looking up therapeutic target information, but also brings new research opportunities, such as the study of novel approaches in discovery of new therapeutic targets and new therapeutic intervention strategies, which are discussed next.