Published in final edited form as:

Comb Chem High Throughput Screen. 2013 December ; 16(10): 764–776.

Chemical Informatics and the Drug Discovery Knowledge Pyramid

Gerald H. Lushington^{*,#}, Yinghua Dong[%], and Bhargav Theertham^{\$}

[#]Molecular Graphics & Modeling Laboratory, University of Kansas, 1251 Wescoe Hall Dr., Lawrence, KS 66045

[%]University of Kansas Center of Excellence for Chemical Methodology and Library Development, 2034 Becker Dr., Lawrence, KS 66047

^{\$}CVS Caremark, 1 CVS Drive, Woonsocket, RI 02895

Abstract

The magnitude of the challenges in preclinical drug discovery is evident in the large amount of capital invested in such efforts in pursuit of a small static number of eventually successful marketable therapeutics. An explosion in the availability of potentially drug-like compounds and chemical biology data on these molecules can provide us with the means to improve the eventual success rates for compounds being considered at the preclinical level, but only if the community is able to access available information in an efficient and meaningful way. Thus, chemical database resources are critical to any serious drug discovery effort. This paper explores the basic principles underlying the development and implementation of chemical databases, and examines key issues of how molecular information may be encoded within these databases so as to enhance the likelihood that users will be able to extract meaningful information from data queries. In addition to a broad survey of conventional data representation and query strategies, key enabling technologies such as new context-sensitive chemical similarity measures and chemical cartridges are examined, with recommendations on how such resources may be integrated into a practical database environment.

Keywords

Chemical Cartridge; Chemical Database; Fingerprints; Frequent Subgraphs; Hashing; Molecular Structure Representation; Similarity

Introduction

Science is a vast pyramid where building blocks of information (carved from nature through systematic analysis) are stacked upon each other to reach ever greater heights of knowledge. As depicted in Fig. 1, the process of drug discovery and development in an excellent example of such a knowledge pyramid, whereby an extensive body of biochemistry and organic synthesis forms the basis for chemical biology, which in turn supports pharmacology and lead optimization. These studies listed on the left side of the diagram correspond directly to the analytical understanding depicted on the right, which progresses from vague and voluminous to specialized and refined as one ascends the pyramid. Within the subfield of chemical research underlying drug discovery, the dependence between past

^{*}To whom correspondence should be addressed: glushington@ku.edu; 785-864-1140.

analysis and future knowledge is similarly pyramidal, with key progress being achieved via scenarios such as:

- a. synthesis protocols identified in the past provide a practical basis for attempting ever more sophisticated synthetic pathways, leading to a greater and greater spectrum of organic chemical diversity available for prospective therapeutic discovery,
- b. past discoveries of chemicals that modulate therapeutically relevant biochemistry provide a basis for developing and refining the chemical biological understanding upon which therapeutic discovery and optimization is based, and
- c. physical principles that have been found to yield useful information about how chemicals interact with biological systems provide the basis for analytical instrumentation for probing chemical biology, and gradually evolve in terms of sophistication, resolution, accuracy and efficiency as new biophysical understanding is available to be integrated.

The development of knowledge pyramids is predicated on the assumption that past information will be readily available to today's researchers, and that the ever greater volumes of new understanding will similarly be propagated to future generations in an intelligible and efficiently accessible form. In the past, such information was effectively compiled within scientific literature, but in our current generation, the volume of underlying data has become far too vast to effectively manage and mine within the print publications alone. Data from some types of analysis require more disk space for a single experiment than would be required to encode the entire contents (including scanned pictures) of the Encyclopedia Britannica (currently 1–4 GB, depending on compression and resolution), and high throughput technologies are making it ever more convenient to pursue a large number of related experiments in parallel or on a non-stop 24/7 serial basis. While the immensity of the resulting data is intimidating, it is dictating new paradigms that should substantially facilitate the growth of knowledge pyramids. At one time, the successful conclusion of a given experiment and analysis would see the underlying raw data printouts stacked, forgotten, in closets, shelves and corners of labs, but the desire for greater resolution, accuracy and detail has pushed the old stacks aside in favor of space-saving digital storage, which has proven far more amenable to post-facto mining and integration into much larger scale information warehouses wherein the original experiment can live on, breathing life into unforeseen future applications.

Chemical informatics is a growing field that is riding the wave of increased emphasis on the effective storage and analysis of information associated with the synthetic and analytical chemistry and chemical biology research that forms the basis for a number of practical disciplines but provides an especially important foundation for drug discovery. The advent of combinatorial chemistry and high throughput screening technologies in the 1990s has necessitated commensurate growth in the efficiency and sophistication of schemes for data/metadata storage, chemical representation, and data mining, thus transforming both the focus and scope of the field. This paper seeks to review general aspects of chemical data representation, storage and retrieval issues and their current and emerging challenges.

Chemical Structure Representation

A foremost concern of chemical informatics is the effective representation of chemical structures in a format that is **1)** computationally efficient (so that a large number of structures can be stored and accessed in a convenient manner), **2)** intuitive (so that researchers accessing the structures can quickly recognize the chemicals that they represent) and **3)** scientifically meaningful (so that modeling and mining protocols can correctly

associate the structural information with existing knowledge or use it to formulate new understanding).

The most common electronic formats for chemical structure representation are described in the next four paragraphs:

Character strings

ASCII (e.g., SMILES[1], SLN[2]) characters can be readily used to encode the atomic composition and internal connectivity information required to accurately specify a given chemical structure (including unambiguous representation of unique structural and stereochemical isomers) in a format that can be readily exploited by a variety of analytical models (e.g., for approximate physicochemical characterization or molecular similarity assessment) or structure rendering tools (e.g., ChemBioDraw[3], ChemSketch[4], Marvin[5], AccelrysDraw (originally IsisDraw)[6]). All of the common string notation syntaxes basically encode the molecule as a sequence of intuitive atomic symbols (e.g., C=carbon, N=nitrogen, O=oxygen, etc.) in an order that corresponds to interatomic connectivity (i.e., the first atom will implicitly be connected to the second, and the second will be connected to the third, etc.). Side chain functionality and rings are handled in the various notations via different types of brackets. By default, atom connectivity is assumed to be via single bonds, while special characters inserted between atoms can be used to imply bonds of higher order. Atomic valences are generally assumed to correspond to neutral species as dictated by Lewis octet rules, while charged exceptions can be invoked with special characters immediate following the affected atom. Still other special characters can be inserted to imply specific stereochemical disposition. Each of these character string syntaxes is designed to ensure that a given molecule is represented unambiguously (i.e., a given string represents only one type of molecule) but generally does not guarantee uniqueness (i.e., a given molecule may be correctly represented by more than one distinct string).

Bit Strings

While the above ASCII representations can be readily encoded in binary digit format with a corresponding reduction in disk storage, this is not always done because of the computational operations required to convert them back and forth to their more informationally rich alphanumeric representation (wherein you can visually determine the types of atoms, bonds, rings and stereochemistry in a molecule). However, fixed-length words composed of binary strings are frequently used to characterize the chemical composition of molecules by identifying the presence or absence of specific substructure groups within the molecule. Given careful rules for how such bit strings are assembled, it is generally possible to specify them in a manner that ensures uniqueness in the sense that any given molecule will only have one bit string assigned to it. This attribute, and the fact that bit strings are very convenient for computers to store, access and compare, have made such representations very popular for rapid retrieval of molecules with specific functionality or with a desired level of structural similarity relative to a given species of interest. The most common bit string forms are generally classified as either 'keys' or 'fingerprints'. The key format is a conceptually simple but informationally limited system whereby each binary digit within a string of fixed length (typically ~1000 bits) is assigned a value of either 0 or 1 depending on the absence or presence of a prespecified atom type or chemical substructure. More widely used than keys, fingerprint formats such as Daylight [7] and UNITY [8] are conceptually similar, however they enable more dense storage of chemical information by hashing and folding manipulations. In the Daylight formalism, a maximum interatomic path length of seven atoms is considered, while in UNITY the formalism attempts to encode all paths up to length six and all non-hydrogenic connectivity paths up to length four. In both

cases, it may be possible to fully encode the molecule within the bit limit for small or structurally simple molecules, but many pharmacologically relevant species exceed the limit. In such cases, the aforementioned hashing and folding schemes provide a useful compression scheme such that the full slate of unique paths can be encoded into the available space. While both keys and fingerprints exhibit limitations in capacity for fully distinguishing between some similar chemical species (because the bit representation reflects presence or absence of a given feature, but not the number of times it may be present in the structure), the folding and hashing processes introduce an additional degree of ambiguity into fingerprints, such that they may not definitively encode whether a given substructure is truly present in the molecule. Ultimately, fingerprint representations attempt to achieve a compromise between two factors that compete for a limited bit space: accounting for the molecular structure as completely and inclusively as possible, and doing so as accurately as possible. In general, the compromise is successful for the simple reason that the majority of all substructures that might conceivably be present within a given molecule are in fact absent, thus most of the prospective bits that would be represented in a molecular key are zeros. When the zeros of key-like representations are hashed and folded into fingerprints, the result is akin to overlaying zeros over zeros which, given a preponderance of zeros, does not incur much loss of information. I.e., if you conceptually overlay (which is to say 'add') a zero that indicating the absence of feature A onto a zero that indicates the absence of feature B, you get a zero which you could interpret as suggesting the absence of both A and B. Of course, the presence of both A and B would also give a zero ($1+1=0$ in binary representation), but this is a much less probable scenario due to the preponderance of zeros in the original key string, so the approximation is usually correct. Thus, while hashing, folding and the limits on encoded path lengths result in molecular fingerprints that are not necessarily unambiguous (i.e., different molecules can have identical fingerprints), the representation has proven very useful for pattern recognition applications such as searching chemical databases for specific compounds, substructures and analogs.

Simple image formats

low-quality formats such as GIF and PNG can readily encode images that are directly analogous to those drawn by hand or generated with the aforementioned rendering software, and incur only a fraction of the computational requirements imposed by photograph-quality formats such as JPG, TIF and BMP. These formats are immediately interpretable by a broad range of software, including web browsers, publication and presentation software.

Cartesian Coordinates

ASCII files that encode the atomic composition of molecules and plausible relative spatial coordinates of the atoms (as would be analogous to chemical structures resolved crystallographically or via NMR solutions) can unambiguously represent unique structural and stereochemical isomers and even specific conformers. This is of value to analytical applications (e.g., physicochemical characterization, chemical biology modeling) and structure rendering.

At this point, no single representation format has proven to achieve all three previously stated goals (i.e., efficient, intuitive and meaningful). Small image formats are very convenient for human recognition, but are not the most efficient form for electronic storage and retrieval, and is nearly useless for analytical applications. ASCII or bit strings are more practical from a storage and retrieval perspective, but they are non-intuitive to human perception and are only directly applicable to a fraction of the potentially relevant analytical applications. Finally, Cartesian coordinate representations are generally desired by the remaining portion of analytical applications, but they are also non-intuitive from a human

perspective and are non-optimal for efficient storage and retrieval. Fortunately, these mutual incompatibilities are readily resolved either by associating multiple forms of representation in the same file (as is performed with vendor-specific data exchange formats developed in conjunction with the aforementioned structure rendering programs, and within community formats such as Chemical Markup Language (CML) [9] and Structure Data Format (SDF) files [10]), or via format conversion tools that can interpret a specific representation and translate it to other desired forms.

Chemical data exchange formats are not broadly used within chemical database environments as they are typically less efficient for information storage and retrieval than representations devised from structured query language (SQL) and its popular extensions (e.g., MySQL, Oracle, etc.), however these exchange formats are (as the term implies) widely used for communicating information about specific chemicals of compound collections, and many chemical databases are implemented in a manner that facilitates direct incorporation of chemical structures and data from these exchange formats into the database environment, a task sometimes accomplished by specially designed data cartridges that ensure direct correspondence between exchange format fields and corresponding SQL objects. Commonly used exchange formats include:

SDF: the oldest and most widely accepted chemical exchange format among visualization, data mining, analytical software and database cartridges relevant to the drug discovery community, the SDF structure encodes molecular structure as either two dimensional (2D) or three dimensional (3D) atomic coordinates indexed by atom type, employs special integer fields to specify bond types and local stereochemistry, and supports an open-ended array of tagged information field that is suitable for communicating molecule-specific metadata (e.g., the source or names of compounds, text information about synthetic protocols, quantity, purity, toxicity, etc.) as well as scientific data points (e.g., molecular weight, solubility, ClogP, assay bioactivity, etc.). Since the population of data/metadata fields is not predefined, most software capable of processing SDF files will either ignore the information or identify the available tags, and either report the information in a simple tabular form or prompt the user for input on how the information is to be processed.

CDX/CDXML: whereas SDF is the most universally accepted chemical data exchange format, the CambridgeSoft CDX (binary) and CDXML (an XML-compliant variant) are likely the most commonly used, arising from the popularity of the ChemDraw software that is widely employed by bench chemists for depicting and storing information about chemicals of interest to them. This format is much better suited than SDF for annotating a given compound (e.g., displaying its disposition within a complex reaction scheme, or describing combinatorial reagents within a Markush diagram) and preserving diagrammatic visual information, but is less conducive to user-specified data/metadata tagging (CDX provides no support for this, and such tags are outside of the CDXML standard) and thus is not commonly used for communicating tabular information or uploading data into databases.

CML/MRV: a compromise between the data-oriented SDF and graphically-targeted CDX, the CML format (and its MRV extended variant used by ChemAxon software) is capable of conveying tabular data and metadata via user-specified tags analogous to SDF, but also enables specification of more visually-oriented information in a manner comparable to CDX. CML has some claim to being a practical structure/data representation standard for the chemical informatics field, and its continued active development can be considered an important intellectual exercise, but its use within the practical drug discovery is secondary: MRV files are commonly exchanged among ChemAxon Marvin users in the same manner that CDX files convey information to

ChemDraw users, but the more powerful data representation capabilities are not widely exploited.

A variety of other common file specifications (e.g., the Isis/Symyx/Accelrys SKC and the ChemSketch SK2 formats) exist, with attributes that are comparable to CDX and CML.

Given the different strengths and weaknesses of these different forms of chemical structure representation, it is not surprising that they are all employed in different aspects of the informational environment within which drug discovery research is conducted. Cartesian coordinate representations are a favorite of the *in silico* community since they provide a convenient basis for constructing inputs for computational simulations (many of which apply classical or quantum mechanical models of interatomic interactions to predict molecular attributes and bioactivity) and shape-based algorithms to augment the relatively modest amount of information that can be extracted from connectivity alone. ASCII string representations are a compact format for unambiguous specification of molecular structure, and thus typically form the basis for database structural representation of chemical compound collections. Bit strings, however, are significantly more conducive to rapid retrieval and comparison, and thus compound databases will often also include this form of representation to efficiently address CPU-intensive data mining such as chemical substructure and similarity searches entail. When a database is called upon to provide a visual depiction of chemical collections and their associated data, it is theoretically possible to embed Java-based structural viewers (e.g., MarvinView [5]) that can translate ASCII string or Cartesian structures into visually intuitive web-accessible representations, however computational efficiency of large scale databases (e.g., PubChem [11]) is much easier to achieve with low-overhead graphical representations. While all of the above requirements could theoretically be unified within a single CML-like format, the data storage requirements of this representation can be prohibitive for a large operation, and thus the effective communication between such requirements is often better achieved through format conversion. Among format conversion tools, the most powerful and widely used is OpenBabel [12], which is currently capable of interconverting between 110 formats and representations commonly used in the drug discovery, chemical informatics and computational chemistry communities. Other useful tools include VEGA [13], CACTVS [14], UNITY Translate [15], CONCORD [16] and CORINA [17]. It should be noted that while OpenBabel appears to have the broadest range of supported interformat conversions, the other programs have useful functional extensions. For example, CACTVS and VEGA support rapid generation of simple image format representation of structures, VEGA, CONCORD and CORINA enable rapid generation of 3D molecular structures from 2D projections and line notations (UNITY Translate can also accomplish this by calling CONCORD as a helper), and VEGA has a graphical interface that can provide a use with access to more advanced functionality such as publication-quality graphics, molecular dynamics simulations, etc. The choice of which routine one might wish to use depends on the task at hand: someone wishing to automate the conversion of a large number of structures would likely choose simple command line tools such as those provide by OpenBabel, CACTVS, CONCORD, etc., that incur little computational overhead (i.e., memory or graphics card use) and can be readily incorporated into a script of a web-driven utility, while those seeking to immediately interact with the structure in an analytical sense would likely choose a graphically powered tool such as VEGA.

Data Representation

Beyond the nuances of chemical structure representation, other aspects of chemical data management and exchange differ little from the requirements in other disciplines. Nonetheless, it is useful to review some basic principles of effective data communication

that will be relevant to information flow within a drug discovery endeavor. The long range model for representing large-scale data (such as that associated with chemical compound collections or high throughput screening experiments) may evolve over time, especially with the emergence of new environments such as cloud computing, but for the time being the most popular environment for sizeable efforts is that of an SQL-based relational database system.

A database is a system optimized for efficiently organizing, storing, and retrieving large amounts of data. Databases usually contain several tables, each of which has internal functionality that is conceptually analogous to spreadsheets such as those developed in Microsoft Excel. I.e., database tables consist of columns and rows, where each row corresponds to a single record and each column contains a different type of attribute of those records. Columns may also encode relationships with other columns (e.g., if one column records the dates on which a specific chemical lot was synthesized, another column may use that information to compute the age of that lot). A natural question to ask is that if a database is so much like a spreadsheet, then why not just use a spreadsheet? In truth, databases are more demanding to construct than spreadsheets, but they provide vastly superior data manipulation capabilities because they exploit more computationally efficient and scalable internal protocols for storing, manipulating and retrieving information. Thus, there are some actions that you can perform on a large database that would be difficult to achieve via spreadsheets, including automatically bulk updates of records, rapid retrieval of all records that match certain criteria and, in the case of *relational databases*, complex cross-referencing across different tables.

It is with relational databases that the greatest power to address complex data associations is enabled. A relational database is a database consisting of separate tables, having explicitly defined relationships, and whose elements may be selectively combined as the result of queries. Data is thus matched via common characteristics found within the data set. For example, a chemistry database may contains several relational tables:

- **User:** contains information about participating scientists and staff, their institutional associations, roles and data permissions
- **Compound:** contains structure, associated synthetic and analytical data, and relevant metadata
- **Plate:** contains chemical identifiers and metadata regarding availability and current distribution for screening and other applications
- **Bioassay:** contains metadata regarding specific bioassays, plus data on the bioactivity of specific compounds

The software used to achieve multi-table grouping is called a relational database management system (RDBMS). The current typical relational database environment is constructed from protocols based on structured query language (SQL) implementations such as Oracle, MySQL, etc.

The process of creating a robust and useful database for a particular application is a complex endeavor, but it may be roughly characterized as entailing a number of standard steps:

Step 1. Understand the requirement

The primary concern of the average bench scientist or technician is to be able to easily specify exactly which data they want to see and in precisely the form they want to see it. Database designers have the responsibility of interpreting this real world desire, understand which data supports which operationally relevant tasks, and devise a scheme for achieving a

reasonable compromise relative to the limits of database structures. If the designer chooses an intelligent implementation plan, the resulting database should be flexible and retain *data independence*. Data independence entails logical conditions such that application programs are independent of the logical structure of database (e.g., if we add more attributes to a relation in database, the data access methods used by application should not be affected) as well as physical independence whereby application programs do not depend on physical data organization (i.e., applications should not require modifications when the physical data organization is changed).

Use cases, which are part of the Unified Modeling Language (*UML*, a standard modeling language in software engineering containing diagram sets, including use case diagrams, class diagrams, object diagrams, component diagrams, sequence diagrams, etc.), are often used to represent problems in a real world context, and to communicate with the end users. An example, illustrated in Fig. 2, outlines the case of a drug discovery research organization, wherein users need to save compound information and related bioassay data, search them by different SAR-relevant conditions (or combinations of multiple conditions), and export data for communication with collaborators.

Step 2. Analyze, create and refine a data model

While the first step maps out the scope of the developing system, including all of the tasks that users will need to carry out, this next step determines what data must be stored in order to achieve these tasks. This requires transforming the real world problem to an abstract representation in a manner that clearly maps the user's and developer's views, which consists of a stored query accessible as a virtual table composed of the result set of a query. It is a dynamic, virtual table computed or collated from data in the database, thus creating an initial data model. This data model may be refined by identifying relationships between classes, as can be represented via class diagrams such as those depicted in Fig. 3. Proceeding with the example introduced in the previous step, the use case diagram suggests three different types of data classes, which can be named 'Compound', 'Bioassay', and 'Plate'.

Step 3. Schema generation

At this point we begin to implement the database by a) creating a table for each class in data model, b) generating fields for attributes of each class, c) choosing a *primary key* (unique identifier for each record in the table, consisting either of a single attribute or multiple attributes in combination) and/or *foreign key* (a field in a relational table that matches the primary key column of another table, often used to cross-reference tables) for each table, and d) identifying the relationships between tables. At this point it may make sense to define specific database *constraints*. A constraint is a rule that the database developer may specify regarding attributes within tables, for example one might specify a Compound table constraint that no entries in the 'LotNumber' column should be NULL in the, since each compound should have a lot number. Note that specification of a foreign key is a constraint in that it specifically maps references between tables that must be satisfied.

Step 4. Database normalization

Normalization is the formal process of organizing data to minimize redundancy. Through this process, we can examine the fields and verify that they are in the right table or whether some restructuring or new tables are required in order to ensure accurate data representation and associations. There are several levels of normalization called normal forms:

- First normal forms (1NF)
 - Eliminate duplicate columns/features from the same table

- Create separate tables for each group of related data and identify primary keys
- Second normal forms (2NF)
 - Meet all the requirements of the 1NF
 - Remove subsets of data that apply to multiple rows of a table and place them in separate tables
 - Create relationships between new tables and their predecessors via foreign keys
- Third normal forms (3NF)
 - Meet all the requirements of the 2NF
 - Remove columns that are not dependent on the primary key
- Fourth normal forms
 - Meet all the requirements of the 3NF
 - A relation is retained in 4NF if it has no *multi-valued dependencies* (i.e., no full constraints between two sets of attributes in a relation)
- Fifth normal forms and up
 - Seldom used in real projects

Following the normalization rules, we may find there are duplicate data in the Bioassay table if two or more compounds are related to the same bioassay. So a new table named 'Bioactivity' is created to present the many-to-many relationship between Compound and Bioassay. After normalization, the ER model is illustrated in Fig. 5.

Step 5. Refine ER model

The ER model may then be refined by adding more keys and constraints.

Step 6. Get information from database

Having implemented the data storage model, it is then important to implement protocols through which users can intuitively query for, and access, specific combinations of data. To achieve this, the developer should assemble relevant queries to retrieve data in requested combinations, and then create the *indexes* (a data structure that improves the speed of data retrieval operations on a database table by storing frequently used columns of a database table, sometimes sorted in a manner that prioritizes access to most commonly retrieved items) and *views* (stored queries composed of the result set of a query and accessible as a virtual table computed or collated from data in the database that will facilitate user access to information).

Step 7. Develop user interface

constructing a user interface to enable facile data acquisition into the database and information retrieval from the system is the last step of database design. Formulation of a friendly user interface that points end users to all reasonable combinations of information that they are likely to need is critical for implementing the knowledge pyramid with which growing understanding of chemical biology is fostered and the insight from which to pursue new drug discoveries is enabled. The user interface is also where data security considerations must be implemented, ensuring that users are accorded data entry and retrieval permissions only where appropriate, ensuring that the only information in the

database that can be changed are entries that logically should be modifiable, and restricting modification or deletion privileges only to those people who justifiably need the capacity.

Information retrieval

As is the case with most disciplines, the value of the knowledge pyramid from the perspective of chemical informatics support for drug discovery efforts is derived from the ability of researchers to ask questions that might be answered by existing data and have the appropriate information returned in an effective and efficient manner. Some examples of the variety of questions that might be asked include:

1. Are any chemicals known that have inhibited target A at an activity level greater than X_A , but have been found to be inactive relative to target B?
2. Given compound J which is known to be active against target A, can we find other species that contain the same piperazine substructure found in J, have not yet been tested against A, and are known to be inactive against B?
3. Given compound J which is known to be active against target A, can we find other species that are at least 85% similar to J, have not yet been tested against A, and are known to be inactive against B?
4. Among those compounds identified in the previous three searches, are there any that were synthesized by Chemist M within the past year, are considered nontoxic, and are available in gram quantities?

The query used to address question 1. is readily implemented through standard SQL syntax alone which, for our example database system, could be expressed as follows:

```
SELECT * FROM Compound
WHERE CompoundID IN (SELECT CompoundID FROM Bioactivity WHERE desc like '
%target A%'
AND outcome='Active' AND data>XA)
AND CompoundID IN (SELECT CompoundID FROM Bioactivity WHERE desc like '
%target B%' AND
outcome='Inactive')
```

However the three subsequent questions pose a more complicated requirement on the database: it must support the rapid retrieval of specific compounds and their analogs, and it must enable complex queries that mix structure/similarity searches with conditional values for reported quantitative data and qualitative metadata. In practice, the computational implementation of the search differs depending on whether one wishes to retrieve an exact match for a specific compound, retrieve compounds containing a given substructure of interest, or retrieve those that are chemically analogous to a given species.

In the case where one is searching for an exact match for a specific compound (i.e., whole structure identity search), the task appears to be a straightforward of exact pattern matching, however one is faced with the problem that standard line notation representations of molecules are unambiguous but not unique (i.e., there is only one molecule for any given SMILES or SLN string, but there may be multiple valid strings for any given molecule) while fingerprints are unique but potentially ambiguous (there is only one valid Daylight or Unity fingerprint for any given molecule, but multiple molecules may share the same fingerprint), neither of which support foolproof compound retrieval. In practice, therefore, the process occurs via a two-step procedure whereby a computationally expedient protocol

such as fingerprint matching is used to narrow a large search pool down to a small number of compounds that are substantially similar to the target species (i.e., share identical fingerprints), and then a rigorous evaluation is performed to confirm or reject these analogs as identical matches. For the latter, the most commonly accepted protocol is to apply graph theoretical representations in which each molecule can be reduced to a unique and unambiguous collection of nodes (to describe constituent atoms) and edges (bonds). An informative review of the application of graph theoretical algorithms to chemical structure description is provided elsewhere [18].

Substructure searching requires a somewhat more challenging variation of this recipe: compounds with the required substructure may be substantially different from the target substructure itself (i.e., the substructure may be only a minor component of a significantly larger molecule), thus exact fingerprint is not an effective prescreen. Furthermore, because the manner in which the hashing data compression can lead to differences in the way substructure-specific bits are set from one molecule to the next, it is also not possible to encode the desired substructure as a bit signature that must be satisfied in molecular fingerprints in order for a match to be perceived. In other words, the bit signature associated with a specific atom connection path can vary across a collection of molecules containing that substructure because any bits in a hashed fingerprint are in principle influenced by the presence or absence of other substructures present in the molecule. One reasonable scheme would thus be to prefilter a large compound collection for species whose fingerprints contain bit signatures similar (though not necessarily identical) to the target substructure, and then perform a more rigorous graph theoretical screening on the subset via algorithms tailored for subgraph detection (e.g., as per Ullman [19]). The diminished value of the fingerprint prefilter is encouraging innovation, however. Recently implementations of purely graph theoretical substructure retrieval systems such as SUBCOMP [20] have emerged. The even bolder concept of encoding structural information and corresponding queries directly within the SQL environment of a database has been proposed, demonstrating reasonable preliminary promise [21]. So, a number of algorithms are available to address the substructure search task. If one provides such an algorithm as a routine (e.g., *Function substructure[Substructure, Table]*) that can be called from within SQL code, then the second query **2.** above can be expressed as:

```
SELECT * FROM Compound
WHERE substructure(piperazine, Compound)
AND CompoundID in (SELECT DISTINCT B1.CompoundID FROM Bioactivity B1,
Bioactivity B2
WHERE B1.CompoundID=B2.CompoundID AND B1.BioactivityID<>B2.BioactivityID AND
B1.desc NOT LIKE '%target A%' AND B2.desc LIKE '%target B%' AND
B2.outcome='inactive')
```

Similarity based search (for retrieval of analogs to a given compound such as requested in **3.**) is perversely either the simplest and most complicated retrieval task, depending on individual perspective. It can be considered the simplest because there is a very straightforward and widely applied protocol for pursuing the task, but is the most complicated because the task is ill-defined, and the standard protocol often achieves mediocre results in the specific applications for which it is really intended. The most common protocol by which similarity searches are performed entails comparison of molecular fingerprints between the target compound of interest (*A*) and each of those (*B*) stored in the database, via the following measure, called the Tanimoto coefficient:

$$Tc(A, B) = \frac{(\text{bits common to } A, B)}{(\text{total bits in } A) + (\text{total bits in } B) - (\text{bits common to } A, B)}$$

This simple and elegant relationship provides a clear way of quantifying relative molecular similarity, all the way from those pairs ($Tc(A, B) = 1$) that are *identical* (at least to within an uncertainty range arising from the imperfect correspondence between molecular structure and fingerprint representation), down to those that are wholly dissimilar ($Tc(A, B) = 0$). The advantage to this paradigm is that it provides an intuitive and computationally simple method for deriving some quantification of the relative similarity of a pair of molecules. The first key disadvantage of this simple formalism is the aforementioned fact that inaccurate similarity assessment will arise from imperfect fingerprint representation of molecular structures of this relationship. The second problem is that it does not explicitly account for the number of nonconserved bits between two molecules. Alternative representations that include recognition of the number of explicit bit mismatches include Baroni-Urbani/Buser, Pearson, Stiles and Dennis coefficients [22]. However all of these representations suffer from a fundamental limitation: they all blindly enumerate the presence or absence of chemical substructure features in molecules without recognizing that **a**) the presence or absence of some substructures are much more important in determining the disposition of molecules toward some specific properties than others (e.g., the presence or absence of alcohol groups has much more bearing on a molecule's aqueous solubility than does the question of whether the molecule has an ethyl group instead of an allyl), and **b**) some chemically very different substructures may actually lead to similar properties (e.g., methyl and chloro substituents are often effectively interchangeable in protein receptor environments). In other words, simple similarity measures do not take context into consideration. This issue has been circumvented to a degree through statistical analysis that suggests that pairs of compounds with high fingerprint Tanimoto coefficients are likely to exhibit similar bioactivity profiles or physicochemical properties. For example, $Tc(A, B)$ 0.85 is often considered to be a reasonable lower limit for confident projections of functional similarity, however a study by Martin et al. revealed from a study of bioactivity results from 115 biochemically diverse assays that only 30% of compounds with Tc values greater than 0.85 relative to a known hit were found to also be bioactive at levels equal to or greater than the known hit. [23]. This is not incredibly surprising since the study of pharmacology is rife with examples of very minor structural modifications that eliminate bioactivity toward a specific target relative to known hits. As well, numerous chemicals exhibit similar attributes and functionality despite having Tanimoto coefficients well below 0.85, the perception of which is the basis for lead-hopping studies whereby information about one family of compounds with some sort of desirable property can be used to intuit distinct families of compounds with similar attributes.

In order to achieve context-sensitive similarity measures, one must find a representation that can be modified adaptively to recognize which features of a molecule are important for determining whether a given molecule does or does not have a specific attribute of interest (i.e., active or inactive against a given target; soluble or insoluble in water; etc.). This is typically accomplished by **a**) assembling a data set that specifies the relative disposition of a collection of molecules toward the assay or other measurement of interest, **b**) providing or calculating simple independent molecular descriptors (e.g., known physicochemical parameters, substructure bits, connectivity indices, shape parameters, etc.), and **c**) applying machine learning or correlative techniques to determine which of the independent descriptors (or combinations of descriptors) in b) tend to reliably distinguish between molecules that satisfy a specified criterion for the measurement in a). This paradigm is extensively applied in pharmacological research and forms the basis for quantitative

structure activity relationship (QSAR) modeling, which is widely used for extrapolating initial screening data toward a broader understanding of the SAR trends from which to design compounds with improved attributes. Such techniques that have been reviewed numerous times before and need not be reexamined in detail here, however the concept of using machine learning techniques to develop context-sensitive compound search protocols is relatively new. The most successful schemes to date use graph theory to encode molecules, then employ frequent subgraph identification methods [24] to identify substructures whose presence or absence in a molecule might be relevant to distinguishing between molecules with desirable (e.g., bioactive) vs. undesirable (i.e., inactive) properties. These frequent subgraphs can then be subjected to feature selection techniques that can rapidly assess whether the presence or absence of a given subgraph has any effect on the likelihood that a molecule satisfies a given property of interest. Analog queries that gauge similarity among molecules based on such context-sensitive properties should thus be more likely to retrieve compounds that are truly similar in ways that are important to the person conducting the search.

A number of specific graph mining methods have been objectively compared by Wang et al. [25] for their relative accuracy in retrieving molecules with similar activity, where accuracy is defined as whether queries based on the structure of an active compound will retrieve similar compounds that are also active, while queries on inactives will retrieve analogous inactives. For their evaluation, Wang et al. surveyed the performance of five different molecular representations: Daylight Fingerprints [7], Closure-tree graphs (C-tree) [26], frequent subgraph indices (gIndex) [27], Wavelet aligned frequent subgraphs (WA) [28], and hashed indices of wavelet aligned graphs (G-hash) [25] with respect to analog retrieval from within an assay for inhibitors of focal adhesion kinase 1 (FADK 1) reported AID810 in PubChem [29], the Jorisson/Gilson data sets (bioactivity data on five distinct protein targets) [30], BindingDB (seven assays selected randomly) [31] and the NCI/NIH AIDS Antiviral Screen data set [32]. Context insensitive searches based on standard Tanimoto similarities over fingerprints (average retrieval accuracy level of 52.92% over all screens; precision values ranging from 27.80 – 76.57% across the different screens; recall from 27.40 – 75.97%) and C-tree (mean accuracy of 51.64%; precision from 29.36 – 62.63%; recall from 46.70 – 72.16%) achieve rather poor retrieval performance in the context of finding functionally similar species. However, the latter three methods establish a degree of context-sensitivity within the search process, which gIndex achieves by paring down representative the collection of potential subgraphs via feature selection, WA attains by measuring similarity via a novel kernel that accounts for property distributions within a small subset of molecules (between 100 – 500), distributed approximately evenly between actives and inactives, and G-hash acquires using both feature selection and an adaptive kernel. This produces a substantial improvement in retrieval performance, with WA attaining a mean accuracy of 66.23% (precision: 56.90 – 98.93%; recall: 58.06 – 79.64%), while gIndex (mean accuracy: 77.83 %; precision: 82.00 – 99.00%; recall: 67.82 – 91.38%) and a 30-feature G-hash protocol (mean accuracy: 77.81%; precision: 77.00 – 96.93%; recall: 64.22 – 86.83%) are the most robust.

While the community generally acknowledges that fingerprint-based similarity searches generally do not yield the most functionally relevant search hits, it is widely considered to be a highly general and computationally efficient protocol that will undoubtedly continue to be the standard protocol for compound retrieval from large chemical databases. The context-sensitive graph-based methods lack the generality of fingerprints since the search protocol must be retrained each time one finds a new assay or property dataset within which one might want to query for analogs. However, analysis by Wang et al. suggests the preprocessing time required to prepare a new data set for G-hashed-based searching (i.e., encoding frequent subgraphs, feature extraction and kernel construction) can be performed

computationally about two orders of magnitude faster than original fingerprint generation (about one second to train a dataset of 40,000 compounds) and requires only a modest amount of disk space (8 MB) [25], which means that many different context-sensitive representations can be saved on a server for future searches. Furthermore, once the preprocessing is complete, an actual G-hash search can be performed at speeds (e.g., ~0.006 seconds for 40,000 compounds) that are between 6–8 times faster than a comparable fingerprint query (and over 100 times faster than gIndex) [25], which suggests that G-hash may provide a clear computational advantage for large database searches.

So from these various suitable analog querying algorithms, one may choose to implement a specific method (or multiple method options) as function calls (e.g., *Function similarity[Compound, Table]*) within the SQL code. Thus query **3** may be expressed as:

```
SELECT * FROM Compound
WHERE similarity(J, Compound) > 0.85
AND CompoundID in (SELECT distinct B1.CompoundID FROM Bioactivity B1,
Bioactivity B2 WHERE
B1.CompoundID=B2.CompoundID AND B1.BioactivityID<>B2.BioactivityID AND
B1.desc NOT
LIKE '%target A%' AND B2.desc LIKE '%target B%' and B2.outcome='inactive')
```

In order to process the subsets of results retrieved from previous searches to provide a query for question **4**, it is first necessary to create a view:

```
Create view vw1 as
(
SELECT CompoundID, userid FROM Compound
WHERE CompoundID IN (SELECT distinct B1.CompoundID FROM Bioactivity B1,
Bioactivity B2
WHERE B1.CompoundID=B2.CompoundID AND B1.BioactivityID<>B2.BioactivityID
AND B1.desc LIKE '%target A%' AND B1.outcome='Active' AND B1.data> XA
AND B2.desc LIKE '%target B%' AND B2.outcome='Inactive')
)
Union
(
SELECT CompoundID, userid FROM Compound
WHERE substructure(piperazine, Compound)
AND CompoundID in (SELECT distinct B1.CompoundID FROM Bioactivity B1,
Bioactivity B2 WHERE
B1.CompoundID=B2.CompoundID AND B1.BioactivityID<>B2.BioactivityID AND
B1.desc NOT
LIKE '%target A%' AND B2.desc LIKE '%target B%' AND B2.outcome='inactive')
)
Union
(
SELECT CompoundID, userid FROM Compound
WHERE similarity(J, Compound) > 0.85
AND CompoundID in (SELECT distinct B1.CompoundID FROM Bioactivity B1,
Bioactivity B2 WHERE
```



```
B1.CompoundID=B2.CompoundID AND B1.BioactivityID<>B2.BioactivityID AND
B1.desc NOT
LIKE '%target A%' AND B2.desc LIKE '%target B%' and B2.outcome='inactive')
)
```

This view *vw1* can now be used to assemble query **4** as follows:

```
SELECT d.compoundID, d.LotNumber, d.Barcode, d.Structure FROM vw1 c, compound
d, User u,
Plate p
WHERE c.userID = u.userID AND u.chemistname='M' AND
c.compoundId=d.compoundId AND
c.userid=d.userid d.plateID = p.plateID AND p.toxic='nontoxic' AND p.date >
'31-Jan-10' AND
p.quantity>1
SELECT c.compoundID, c.LotNumber, c.Barcode, c.Structure FROM Compound c,
User u, Plate p
WHERE c.userID = u.userID AND u.chemistname='M' AND c.plateID = p.plateID AND
p.toxic='nontoxic' AND p.date > '31-Jan-10' AND p.quantity>1
```

Thus, the development of chemical databases for drug discovery purposes employs an interesting challenge of blending very generic and very specialized data representations: standard information retrieval protocols that differ little from those that one would use for any biomedical or corporate database on one hand with sophisticated chemical informatics representations and retrieval strategies on the other.

Chemical Cartridges

While it is possible for a skilled database developer to devise all of the schemas and specialized functions necessary to embed chemical awareness into a database environment, the customization process can be greatly facilitated by use of cartridge technology. While the various forms of SQL (in particular Oracle) have a diverse range of native data types, traditionally those data types have not been particularly well adapted to chemical information, thus a cartridge is a software package designed to extend the database environment by implementing pre-defined data structures, relationships, query types and special data operation functions in a way that influences how the database interprets, stores, retrieves and indexes the domain-specific data. This minimizes the need for individual developers to re-implement features and data specifications that are used commonly throughout a specific discipline such as chemical informatics and chemical biology, and simultaneously encourages applications to exploit some of the more sophisticated and powerful features available within the database environment. Data cartridges may also define a set of APIs (Application Programming Interfaces) or procedures that can be used for domain-specific actions such as inserting or updating chemical structures, performing substructure or similarity searches, and developing customized applications. A number of cartridges exist that have been developed specifically for drug discovery applications, of which some well-known examples introduced through the years include the JChem cartridge from ChemAxon [33], Accord from Accelrys [34], DayCart from Daylight [35], MDL/Direct RCS [36], Auspyx from Tripos [37], and the CambridgeSoft Cartridge [38].

Among the various services performed by cartridges, specialized indexing is one of the most important augmentation for improving the efficiency of a database. Indexing data enables

much faster data retrieval for queries submitted to large data sets. By default, a database would handle indexing via native data types, but specific methods in cartridges dictate (based on established common usage patterns within the community) how data should be stored, retrieved and manipulated in the tables and indexes. For example, the JChem cartridge uses its own implementation for indexing chemical structure data as follows:

```
CREATE INDEX jc_idx ON jchemtest(structure_col) INDEXTYPE IS jc_idxtype;
```

where `jc_idxtype` is the encoded index type (as devised by the cartridge vendors) implemented for performing index manipulations. This encoded structural representation framework can effect great time savings on the part of local database developers, who would otherwise have to identify, develop and specify numerous external function calls to specify how the molecular structural information and any associated metadata should be converted from its native format (e.g., SDF files, SMILES strings, cdxml files, etc.) into a form that can be stored, manipulated and retrieved from a database environment in the most efficient manner. For comparison purposes, the same function is achieved via the CambridgeSoft Cartridge as follows:

```
CREATE INDEX ix ON struc_table(structures) INDEXTYPE IS  
CsCartridge.MoleculeIndexType;
```

where `CsCartridge.MoleculeIndexType` is the corresponding index type.

Cartridges also render the database interface more intuitively accessible to chemists, via predefined SQL query operators make query syntax more transparent to non-developers. For example, the JChem cartridge enables one to formulate a substructure query based directly on an intuitive molecular representation such as a SMILES string, without resorting to external function calls, e.g.:

```
select id from plaintable where jc_contains(structure,  
'CN1C=NC2=C1C(=O)N(C)C(=O)N2C') = 1;
```

where `jc_contains` is a SQL operator developed by ChemAxon to perform the substructure searching. Similarly CambridgeSoft has an SQL query operator `MoleculeContains` that enables queries that simultaneously search for substructures while attaching a diverse range of different property conditions, such as:

```
SELECT mol FROM struc_table WHERE CsCartridge.MoleculeContains(mol,  
'c1ccccc1', '', 'MASSMIN=200, MASSMAX=500')=1
```

which queries a table `struc_table` for molecules containing a phenyl group and having a molecular weight between 200 and 500 daltons.

Practical Considerations

In order for a database to be a useful tool for real research activities, one must invest some thought into basic issues such as the hardware and operating system to use for hosting the server, ensuring that the database framework is installed in a manner that will enable robust

and efficient database operation, establishing protocols to ensure that the data (plus any effort in designing and customizing the server) is safeguarded from system failure or corruption through rigorous backup, and implementing procedures to track system performance in order to spot potential problems before they become critical and to identify aspects for improvement.

Hardware and OS considerations

A key initial planning step should entail analyzing the infrastructure requirements of the specific software (including database environment, plus any additional services to be implemented for front-end analysis) and developing a hardware specification (e.g., number and speed of CPUs, amount of memory and storage, etc.) that somewhat exceeds the currently anticipated requirements as dictated by the scope of the application (i.e., number of compounds to be stored, approximate volume of associated data, volume of concurrent users, etc.). The plan should also determine what conditions of Service Level Agreement are appropriate for availability of the application, which may dictate the use of High Availability solutions for the OS and/or database level to minimize the downtime. For example, Sun provides Sun Clusterware for Solaris operating systems, and Oracle has Oracle Real Application Clusters (RAC) as solutions for optimizing system availability. Some flexibility in choice of operating system and database software can be made based on budgetary considerations and on technical support provided by vendor of any domain specific software that one intends to install on the software. In general, Solaris, Linux and Windows operating systems are common, viable choices for the server, due to general considerations of stability and availability of specific software optimized for those platforms.

Database installation and testing

In many cases it is possible to obtain a server in which the hardware, OS and database environment comes as a prepackaged unit. For the conservative or uninitiated developer, this is an advisable choice since the total package is generally assembled in a form that, over the consensus of numerous prior customers, is likely to be fairly robust and efficient for a reasonably general variety of applications. For those developers who are planning to install a specific chemical cartridge, it is likely that the cartridge vendor has assembled specific recommendations either on complete-package server solutions appropriate for that particular cartridge, or else a prescribed range of different recipes for constructing a reasonable server. Note that since many cartridges are suitable only for a fairly limited range of specific database environments, it is advisable that one at least pay close heed to those specifications. Fortunately, those adventuresome spirits who do proceed with their own database installation typically find that if they have made intelligent choices in server specification, the out-of-box installation process is straightforward. The next step would be to create a database instance to address the specific application that the developer plans to address. Typically this involves specifying the system resources that will be available to the database (e.g., how much memory the database instance would use) and parameters that affect the performance of the database instance. Although these parameters will likely vary for each specific database implementation, the vendor may have recommendations on values that are appropriate for their software product. It is generally advisable to have some sample data available for prepopulating a test version of the database that can then be subjected to a stress test/performance analysis (with a realistic user load on the application) to verify that the database specification is reasonable, and identify initialization parameters that can be adjusted for better application performance.

Backup and Recovery

A key operational issue to address is the availability and recovery of a database in case of a hardware failures or human errors. A sound backup strategy would enable database recovery under such circumstances. Most database environments support some sort of backup protocol (e.g., Oracle provides a tool called RMAN for this purpose). Once the protocol is in place, there is still a responsibility on the part of a database administrator to ensure that frequent backups of the database are indeed performed and that the archivelog files that accrue between these backups are also properly backed up. In case the database needs to be recovered to a certain point in time in the past, a full backup can be restored and archivelogs can be applied to recover the exact system status at the desired time point. Logical backups of the database-like exports can also be taken periodically, but they only provide a snapshot of all the tables in the database as of the time the export started. This does not unambiguously guarantee the possibility of restoring to an exact state present at a specific time point, but may still be useful for restoring a table whose contents are static.

Table Statistics

usage information should be gathered on the tables on a regular basis, since the resulting statistics can provide the basis for optimizing the execution plan, which is the sequence of data operations that the underlying SQL statement performs, including whether to access the data via index scans or full table scans, the order in which tables are accessed, the kind of joins (hash join/nested loop) used when the SQL statement joins two tables on common column(s). In some database environments such as Oracle 10g [39] (and later), statistics are automatically gathered on tables. In Oracle, tables in which a certain adjustable percentage of data (default value of 10%) has changed since the last analysis are selected as candidates for analysis. Some chemistry cartridges implement specific protocols to control how statistics should be collected.

Summary and Conclusions

The research underlying preclinical drug discovery is a very information intensive process that can be greatly expedited by convenient access to information that has previously been acquired by the community. While the processes by which such information can be productively mined in order to pave the way toward new advances is left for other articles to address, we have outlined some basic considerations that we believe will help the uninitiated in their efforts to develop or refine data storage and retrieval schemes for their own research efforts, thus contributing to the knowledge pyramid upon which future discoveries will be based.

Progress in database technologies have been advanced by many different disciplines in ways that may be easily exploited by information technologists as they develop data management schemes for drug discovery. Chemical informaticians have played a key role in this progress through the encoding of chemical cartridges that simplify and standardize the storage and utilization of molecular information. By other measures, however, the adoption of powerful new chemical informatics technology has been tepid. Despite the explosion of available chemicals and bioactivity data that has produced a wealth of applicable information, many drug discovery scientists still pursue structure-activity relationship analysis as a pencil and paper exercise, and attempt lead hopping either by intuition or trial-and-error. Perhaps this is because the databases that support these scientists still rely on old mediocre chemical informatics technologies that are known to yield mediocre performance. Given major recent successes in formulating molecular representations that support much more efficient and intelligent database queries (especially in the area of chemical similarity searches that are potentially so important for analoging and novel lead identification), one can imagine the

great benefits that can be derived from a marriage between the tremendous power of existing databases and the great intelligence being encoded into new chemical informatics methods. This more efficient and effective way of growing and utilizing the knowledge pyramid may be a key ingredient in an emerging recipe for success in the extraordinarily competitive and exacting challenge of discovering new leads with the potential to survive through to the pharmaceutical marketplace.

Acknowledgments

We gratefully acknowledge funding from NIH awards P50 GM069663-06, U54 MH005031-02, and P20 RR016475.

References

1. Weininger D. J Chem Inf Comput Sci. 1988; 28:31.
2. Ash S, Cline MA, Homer RW, Hurst T, Smith GB. J Chem Inf Comput Sci. 1997; 37:71.
3. ChemBioDraw 12.0. CambridgeSoft, Inc; Cambridge MA: 2010.
4. ChemSketch 12.0. ACD Labs, Inc; Toronto ON: 2010.
5. Marvin 5.4. ChemAxon, Inc; Budapest, Hungary: 2011.
6. Accelrys Draw 4.0. Accelrys, Inc; San Diego CA: 2010.
7. Daylight Fingerprints. <http://www.daylight.com/dayhtml/doc/theory/theory.finger.html>
8. Clark, RD.; Dorfman, R.; Heritage, TW. Abstracts of Papers. 223rd ACS National Meeting; Orlando, FL. 2002. p. COMP-203
9. Rzepa HS, Murray-Rust P, Whitaker B. Chem Soc Revs. 1997; 26:1.
10. Dalby A, Nourse JG, Hounshell WD, Gushurst AKI, Grier DL, Leland BA, Laufer J. J Chem Inf Comput Sci. 1992; 32:244.
11. Wheeler DL, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, Geer LY, Helmberg W, Kapustin Y, Kenton DL, Khovayko O, Lipman DJ, Madden TL, Maglott DR, Ostell J, Pruitt KD, Schuler GD, Schriml LM, Sequeira E, Sherry ST, Sirotkin K, Souvorov A, Starchenko G, Suzek TO, Tatusov R, Tatusova TA, Wagner L, Yaschenko E. Nucl. acids res. 2006; 34:D173. [PubMed: 16381840]
12. OpenBabel v. 2.30. GNU Free Documentation License 1.2. 2010. http://openbabel.org/wiki/Main_Page
13. Pedretti A, Villa L, Vistoli G. J Mol Graph Model. 2002; 21:47. [PubMed: 12413030]
14. Ihlenfeldt WD, Takahashi Y, Abe H, Sasaki S. J Chem Inf Comput Sci. 1994; 34:109.
15. UNITY 8.1. The Tripos Associates (Certara); St. Louis, MO: 2010.
16. Concord 8.1. The Tripos Associates (Certara); St. Louis, MO: 2010.
17. Gasteiger J, Rudolph C, Sadowski J. Tetrahedron Computer Methodology. 1990; 3:537.
18. Balaban A. Chemical Graphs: Looking Back and Glimpsing Ahead. J Chem Inf Comput Sci. 1995; 35:339–350.
19. Ullmann JR. J Assoc Comput Mach. 1976; 23:31–42.
20. Hattori, Masahiro; Tanaka, Nobuya; Kanehisa, Minoru; Goto, Susumu. SIMCOMP/SUBCOMP: chemical structure search servers for network analyses. Nucleic Acids Research. 2010; 38:W652–656. [PubMed: 20460463]
21. Golovin A, Henrick K. Chemical Substructure Search in SQL. J Chem Inf Model. 2009; 49:22–27. [PubMed: 19072559]
22. Haranczyk M, Holliday. Comparison of Similarity Coefficients for Clustering and Compound Selection. J Chem Inf Model. 2008; 48:498–508. [PubMed: 18293953]
23. Martin YC, Kofron JL, Traphagen LM. Do structurally similar molecules have similar biological activity? J Med Chem. 2002 Sep 12; 45(19):4350–8. [PubMed: 12213076]
24. Inokuchi, A.; Washio, T.; Motoda, H. An apriori-based algorithm for mining frequent substructures from graph data. Proc. 2000 European Symp. Principle of Data Mining and Knowledge Discovery (PKDD'00); Lyon, France. Sept. 1998; p. 13-23.

25. Wang X, Huan J, Smalter A, Lushington GH. Application of kernel functions for accurate similarity search in large chemical databases. BMC Bioinformatics. 2010; 11:S8. [PubMed: 20438655]
26. He, H.; Singh, AK. Closure-tree: an index structure for graph queries. Proc. International Conference on Data Engineering'06 (ICDE); 2006.
27. Yan, X.; Yu, PS.; Han, J. SIGMOD. 2004. Graph indexing: a frequent structure-based approach.
28. Smalter, A.; Huan, J.; Lushington, G. Graph Wavelet Alignment Kernels for Drug Virtual Screening. Proceedings of the 7th Annual International Conference on Computational Systems Bioinformatics; 2008.
29. PubChem Bioassay AID 810. http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=810&loc=ea_ras
30. Jorissen R, Gilson M. Virtual Screening of Molecular Databases Using a Support Vector Machine. J Chem Inf Model. 2005; 45(3):549–561. [PubMed: 15921445]
31. Liu T, Lin Y, Wen X, Jorissen RN, Gilson M. BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. Nucleic Acids Research. 2007; 35:D198–D201. [PubMed: 17145705]
32. NCI/NIH AIDS Antiviral Screen data set. 1999. <http://dtp.nci.nih.gov>
33. JChem Cartridge 5.4. ChemAxon, Inc; Budapest, Hungary: 2010.
34. Hayward J, Buchan I. Current Opinion Drug Disc Dev. 2000; 3:306. Accord Chemistry Cartridge - 6.0.2. Accelrys, Inc; San Diego, CA: 2010.
35. DayCart. Chemical Informatics Systems, Ltd; Aliso Viejo, CA: 2007.
36. MDL/Direct RCS. Accelrys, Inc; San Diego, CA: 2003.
37. Akella, L.; Mandl, M.; Homer, Webster. Abstracts of Papers. 229th ACS National Meeting; San Diego, CA. 2005. p. CINF-113Auspyx 1.7. The Tripos Associates (Certara); St. Louis, MO: 2008.
38. Cambridgesoft Oracle Cartridge 11.0.1. CambridgeSoft, Inc; Cambridge MA: 2011.
39. Oracle 10g. Oracle, Inc; Redwood Shores, CA: 2003. <http://www.oracle.com/technetwork/indexes/documentation/index.html>

Figure 1.
Drug Development Knowledge Pyramid.

Figure 2.
Use case diagram

Fig. 3.

Class diagrams. Numbers on the graph vertices refer to the number of specific instances that can be represented within a certain field, i.e., 0..* indicates that any number of values from zero to many may be stored, while 1 means that only a single value should be stored.

Figure 4.

Entity-Relationship (ER) Model, providing an abstract and conceptual representation of data, containing entity, relationship, and attributes. PK refers to the Primary Key. FK represents foreign keys. Numbers on the graph vertices refer to the number of specific instances that can be represented within a certain field, i.e., 0..* indicates that any number of values from zero to many may be stored, while 1 means that only a single value should be stored.

Figure 5.
Normalized ER model. PK refers to the Primary Key. FK represents foreign keys.