DOE/ER/61923-1

Database Transformations for Biological Applications

Final Report (July 1, 1994 to June 30, 1998)

C. Overton (PI), S.B. Davidson*, P. Buneman, and V. Tannen (co-PIs)
Dept. of Genetics and Dept. of Computer and Information Science,
Center for Bioinformatics
476 Curie Boulevard
University of Pennsylvania
Philadelphia, PA 19104-6145

*Phone: (215) 573-4411, *Fax:* (215) 573-3111,
Email: {susan,peter,sharker,coverton,val}@cis.upenn.edu

NOTICE: This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed or represents that its use would not infringe privately-owned rights

Prepared for THE U.S. DEPARTMENT OF ENERGY, AGREEMENT NO. DE-FG02-94-ER-61923

Abstract

The goal of this project was to develop tools to facilitate data transformations between heterogeneous data sources found throughout biomedical applications. Such transformations are necessary when sharing data between different groups working on related problems as well as when querying data spread over different databases, files and software analysis packages.

We summarize progress made during the term of this grant in the development of the Kleisli query system. Kleisli implements a high level query language called the Collection Programming anguage (CPL) and contains drivers to access many types of databases found throughout the genomic community, including: relational databases (Oracle and Sybase); ASN.1; BLAST and FASTA; Shore (and potentially other object-oriented databases); EcoCyc (a Lisp-based metabolic pathways system); SRS; Medline; MMDB; OPM/CTL; US and IBM patent databases. The query system is based on a complex model of data, which provides a natural encoding of data in this domain, and uses a number of optimization strategies and sophisticated parallel execution strategies to improve the performace of queries. The system has been used for applications with the Center for Bioinformatics at the University of Pennsylvania, and within projects at SmithKline Beecham. It also forms the basis of the Tambis system developed at the University of Mancheste, UK.

We then describe a re-implementation of Kleisli called K2. K2 is implemented in Java (Kleisli was implemented in the rapid prototyping language ML), and allows the overlay on an ontology to aid integration. The report closes with some preliminary observations on the use of XML for data integration.

DOE Patent Clearance Granted
MP Dvorscak 4-11-01
Mark P. Dvorscak Date
(630) 252-2393
E-mail: mark.dvorscak@ch.doe.gov
Office of Intellectual Property Law
DOE Chicago Operations Office

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

1. Introduction

The process of building a new database relevant to some field of study in biology involves extracting, transforming, integrating and cleansing data from multiple external data sources, as well as adding new material and annotations. For example, EpoDB [1] is a database created at the University of Pennsylvania Center for Bioinformatics www.cbil.upenn.edu, which was designed to study gene regulation during differentiation and development of vertebrate red blood cells. In building EpoDB, data relevant to red blood cells were *extracted* from Genbank, Swissprot, TRRD (transcriptional regulation data) and GERD (expression levels data) and *transformed* into a relational form. The data was then *cleansed* of errors using a semi-automated approach. Cleansed data was then *integrated*, and additional information or *annotations* were entered manually to the integrated data.

Due to the cleansing and value added to the extracted data, databases such as EpoDB are commonly referred to as curated data warehouses, or in database jargon, as *materialized* views. In contrast, in an *unmaterialized* view, the data is not stored, but is extracted on demand. The specification of the view describes how to translate queries against the view into queries against the underlying distributed data sources. The CPL queries illustrated at http://agave.humgen.upenn.edu/cpl/cplhome.html are examples of a solution using unmaterialized views.

The advantage of a materialized solution is that system performance will, in general, be much better than is possible in an on-demand environment. First of all, query optimization can be performed locally at the data warehouse (assuming the system used for warehousing is a good one); second, inter-data source communication time will be eliminated. System reliability should also be significantly higher since there are fewer dependencies on network connectivity and the individual data sources (again, assuming that the system in which the materialized solution is stored is reliable). It is also less costly to implement any inter-data source constraints that have been determined in the integration [2]. The disadvantages of a materialized solution include the initial cost of constructing the warehouse, and the cost of maintaining currency of the data. As updates are made in the underlying data sources, the materialized warehouse becomes out of date. Keeping the materialized view current involves translating the updates to the underlying data sources into updates on the warehouse, which may involve a complex query over the warehouse as well as source data if the integrated schema is complex.

Despite these differences, both solutions share the need to extract, transform and integrate data from multiple external data sources, as illustrated in Figure 1. At the bottom of the figure are the data sources of interest, and at the top are applications which require access to an integrated view of the data (e.g. the queries of interest submitted to EpoDB). The middle area surrounded by dotted lines indicates that some form of data transformation and integration must be performed to enable top-level data access. EpoDB is an example of a data warehouse solution, in which the transformed and integrated data is physically stored in an Oracle database.

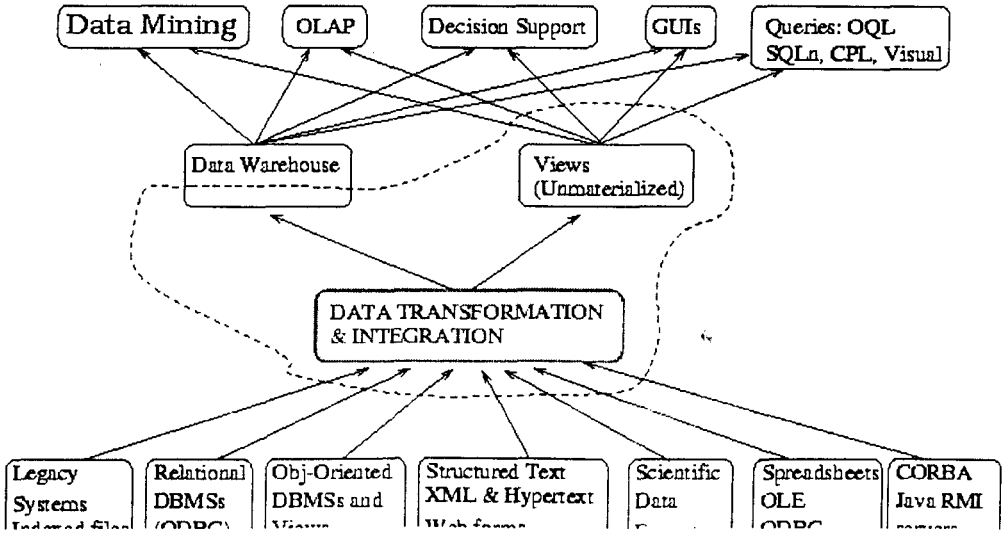

Figure 1: Data Warehousing (Materialized View) versus On-Demand Integration (Unmaterialized View)

While solutions exist for data transformation and integration when the underlying data sources are all relational, they do not work in the context of biomedical data sources due to their variety and complexity. For example, the "database of molecular biology databases" www.infobiogen.fr/services/dbcat/ contains information about databases of general interest in molecular biology and genetics. Of the more than 400 databases listed, less than 10 appear to use commercial database technology.[1] Over 50% give an ftp site from which source data is available in some structured form, and most provide some sort of web interface. A cursory examination of the ftp sites shows that quite a number of these databases use some variation of the EMBL format; others use ASN.1 or an ACeDB format.

Although relational database technology is well developed, and provides a number of features not generally provided by systems based on data formats (such as security, support for multiple concurrent users, recoverability, a "complete" query language, and the ability to embed queries in general purpose programming languages such as C and Java), formats such as EMBL and ACeDB are appealing in the biomedical community for a number of reasons. First, biomedical data are complex and not easy to represent in a relational DBMS; typical structures include sequential data (lists) and deeply nested record structures. As an example, a natural translation of a Swissprot entry to a relational form results in the entry being split over about 15 tables. Second, formatted files are easily accessed from languages such as Fortran, C, Java and Perl, and a number of useful software programs exist that work with these files. Third, the files and associated retrieval packages are available for a variety of platforms and are much less costly than commercial systems. It is therefore unlikely that existing biomedical data sources will migrate to relational systems so that existing commercial tools can be used for transformation and integration.

In this report, we briefly describe how the Kleisli system can be used for data transformation and integration within biomedical applications. Kleisli is based on a complex value model of data, and has a well-defined query language (the **C**ollection[2] **P**rogramming **L**anguage, CPL) for which powerful optimization techniques have been developed. The appeal of a complex value model of data is that it captures most biomedical data formats — EMBL, ASN.1, ACeDB, etc. It is therefore easy to translate from biomedical data into this form, as it is for a variety of commercial data models (relational, object-oriented, etc). We then discuss the new generation of Kleisli, called K2. We close by observing how semi-structured data, most notably the data format XML, may be "on the horizon" for data exchange, transformation and integration.

2. Kleisli and CPL

An example of a CPL type called "Genbank_entry" is given in Figure 2. This models a simplified Genbank entry, and uses the following notation: $\{\tau\}$ denotes a set, $\{|\tau|\}$ a bag, $\{||\tau||\}$ a list, $[l_1: \tau_1, \ldots, l_n: \tau_n]$ a record, and $<l_1: \tau_1, \ldots, l_n: \tau_n>$ a variant (tagged union). Note the use of a list of records within the References field, indicating that the order of references (and again of authors' names within each reference) is important. Also note the use of a variant or "tagged union" type within the Features record field, indicating that a feature is either a Source or Gene type. Values in CPL are explicitly constructed as follows: $[a_1= e_1, \ldots,$

[1] The low percentage of biomedical databases that use commercial technology is actually not surprising. It is estimated that as much of 90% of the world's data is not in a commercial database, but rather in data formats developed for the transmission and/or archiving of data and "applications" such as on-line dictionaries and BLAST.

[2] The "collection" types in CPL are set, bag (set with duplicates) and list (bag with order).

Figure 2: Sample CPL type

$a_n=e_n$] for records, giving values e_i of the appropriate type for each of the attributes; <a = e> for variants, giving a value of the appropriate type for label a; and $\{e_1, \ldots, e_n\}$ for sets.

```
Genbank_entry = [ ID: string,
                  AccNumber: string,
                  Description: string,
                  SeqLength: int,
                  Source: string,
                  References: {|| [Title: string, Authors: {|| string ||} ] ||},
                  Sequence: string,
                  Features: {|| < Source: [From: int, To: int, Complement: bool, Organism: string],
                                  Gene: [From: int, To: int, Complement: bool, Gene: string] > ||} ]
```

Queries in CPL are expressed using *comprehensions*. As an example, suppose we have a set of values of type Genbank_entry called gbs and wish to extract the ID and AccNumber of all sequences whose source is "Bacillus subtilis". We would write this as the following comprehension:

```
{ [ID=e.ID, AccNumber=e.AccNumber] | \e ← gbs, e.Source="Bacillus subtilis"}
```

The variable e is successively bound (indicated by the backslash, "\e") to each element of gbs (indicated by the generator "←"). For each e, if e.Source="Bacillus subtilis" then the value [ID=e.ID, AccNumber=e.AccNumber] is constructed; all such values are then combined in a set to form the final result.

Another important concept of CPL is *pattern matching*. Instead of binding an element of a collection to a variable name using the expression "\e", it is possible to specify complex variable bindings and conditions using patterns. For example, we could rewrite the query above using patterns as:

```
{ [ID=i, AccNumber=a] | [ID=\i, AccNumber= \a, Source="Bacillus subtilis", ...] ← gbs}
```

The use of the ellipsis "..." indicates that other fields are present in the record type of an element of gbs.

A final example shows how the gene features of Genbank entries can be restructured (flattened) into a relational format, so that each gene feature is stored separately with the ID of the entry in which it appears:

```
{ [ID=i, From=s, To=e, Complement=c, Gene=g] |
            [ID = \i, Feature= \f, ...] ← gbs, <Gene ▫ [From=\s, To=\e, Complement=\c, Gene=\g]> ← f}
```

Note that the result type is {[ID: string, From: int, To: int, Complement: bool, Gene: string]}. In addition to using pattern matching to bind variables such as i to the value of a field, the pattern "Gene=" is used to extract only those features which are genes.

CPL has been implemented in a data transformation and integration system called Kleisli. The system is based on (1) an extensible architecture for implementing the CPL query primitives; (2) optimization techniques that extend many known techniques from relational databases and derive new techniques for complex data types; and (3) a collection of generic drivers for external data sources. The extensible architecture allows new data types and operations to be easily added to the system. Optimizations allow time and resource consuming queries to be rewritten into equivalent "cheaper" queries, and determine good ways of executing particular operators. By "generic drivers" we mean that once a driver for a particular type

of source (e.g., the relational database system Oracle) has been developed, it can be used for any database of that type (e.g., the Chr22DB database, which is implemented in Oracle). The function of a driver is to communicate queries to the underlying data source and translate the results of queries into CPL format.

As an illustration of how Kleisli can be used to integrate heterogeneous databases, suppose that we have a BLAST package and an SRS version of Swissprot. The type of a Swissprot entry is given in CPL as:

```
Swissprot_entry = [ID: string, AccNumber: string, Description: string, GeneName: string,
                   Organism: string, Comments: {string}, SeqLength: int, Sequence: string]
```

Entry points into each of these data sources are modeled in Kleisli as *functions*. For example, the function BLAST takes as input the desired database against which to perform comparisons and a nucleotide sequence, and returns a set of hits in Genbank format (i.e. the hits will have type Genbank_entry). The function Swissprot_Gene_Org models access to Swissprot: It takes as input a gene name and source organism and returns a set of entries of type Swissprot_entry. We can then use these entry points in CPL queries to perform integration of the underlying data sources.

Suppose we wish to integrate Genbank and Swissprot as follows: "Given a nucleotide sequence NUCLEOTIDE_SEQUENCE, find all similar Genbank entries using BLAST. For each gene feature that has a corresponding Swissprot entry, return the Genbank accession number, the gene feature fields, and the ID and comments (annotations) from the Swissprot entry."

The CPL query (which we will name "Genbank-Swissprot") to perform this integration is as follows:

```
{ [AccNumber ▫ gb.AccNumber, From = f, To = t, Complement = c,
   Gene ▫ gn, SwissID ▫ s.ID, SwissComments = s.Comments] |
                                        \gb ← BLAST("genbank", NUCLEOTIDE_SEQUENCE),
                                        <Gene = [From = \f, To = \t, Complement ▫ \c, Gene = \gn]> ← gb.Features,
                                        \s ← Swissprot_Gene_Org(gn, gb.Source) }
```

Genbank-Swissprot produces as output a set of records with fields AccNumber, From, To, Complement, Gene, SwissID and SwissComments. The input sequence NUCLEOTIDE_SEQUENCE is passed to BLAST with the specification to use Genbank. For each hit gb (which has type Genbank_entry), we extract all fields of features that are genes. The gene name and source fields are then used to access Swissprot to obtain the Swissprot ID and comments.

These examples illustrate only a small part of the expressive power of CPL. A more detailed description of the language is given in [3], and more extensive discussions of data integration using Kleisli can be found in [4, 5, 6].

3. K2, ODMG, and K2MDL

Another approach to the integration of information sources that was begun under funding for this grant is the K2 system [7]. This system has interfaces based on the ODMG standard [8] and extensions thereof. In particular, integrated schemas are expressed in ODL (the data definition language) extended with variant types, while queries against these schemas are expressed in an extension of OQL (the query language) with variant operations. For example, the schema of an integrated view analogous to Genbank-Swissprot would be represented in ODL as follows:

```
class Genbank-Swissprot
 (extent Genbank-Swissprot_extent)
 {attribute string AccNumber;
  attribute int From;
  attribute int To;
  attribute bool Complement;
  attribute string Gene;
  attribute string SwissID;
  attribute set<string> SwissComments;}
```

Figure 3: ODL Schema for Genbank-Swissprot View

K2 also uses ODL to represent the data sources to be integrated. It turns out that most biological data sources can be described as "dictionaries" returning complex values. For example:

```
Swissprot_Gene_Org :  dictionary<struct{string GeneName, string SourceOrganism}, set<Swissprot_entry>>
```

A dictionary is a finite function. Each dictionary consists of a domain of keys (e.g., GeneName and SourceOrganism) and an association that maps each of the keys in the domain into a value (e.g. a set of Swissprot entries). To describe the integration, K2 uses a new language, K2MDL, which combines the syntax of ODL and OQL to specify data transformations from multiple sources to one target. The key to making ODL, OQL, and K2MDL work well together is the expressiveness of the internal framework of K2, which is based on complex values and dictionaries. ODL classes with extents are represented internally as dictionaries with abstract keys (the object identities). This framework opens the door to interesting optimizations that make this approach feasible.

As with Kleisli, a tremendous enhancement in productivity is gained by being able to express complicated integrations in a few lines of K2MDL code as opposed to several orders of magnitude more lines of, for example, Perl or C++ code.

4. Semi-Structured Data and XML

As a universal data exchange format, XML [9] may well supplant existing formats such as ASN.1 for biological data. Having a universal format for data exchange will surely simplify some of the lower-level driver technology that is part of the Kleisli and K2 systems. An open question is whether it will do more than this. Since this approach may well become popular in the next several years, we summarize some pros and cons of using XML as a basis for data integration.

XML is an example of a *semistructured* data format [10], which means that each data element component of the data) carries its own description. Unlike ASN.1, there is no separate schema to constrain the structure of the data. What this means in practice is that it will be possible to make ad-hoc changes to the structure of individual components (such as additional fields, missing elements etc.) and collections of these elements may be heterogeneous. ACeDB [11] is an interesting example of what might be taken as semistructured data. Although it has a schema, the schema imposes only weak constraints on the data (it can accommodate missing data, for example) and the type system used by ACeDB — that of an edge-labeled graph — is remarkably close to that adopted for semistructured data.

As an example, an XML version of an instance of a Genbank entry (whose type is given in Figure 2) is shown in Figure 3. Note that the type is carried in the labels.

Because it can accommodate schema changes (there is no schema) and annotations, semistructured data may seem to be a solution to biological data integration problems where the schemas evolve with the progress of experimental techniques and scientific discovery. In the XML example, one could add a new field to an <entry> element or one could have a repeated <Source> field. Moreover, data integration may be performed simply by ignoring the structure and using one of the query languages developed for semistructured data or XML [12, 13, 14, 15]. We believe this approach should be treated with a great deal of caution. Biological data has a very rich structure. We should not ignore that structure in our applications just because that structure evolves. The semistructured approach has the following serious dangers:

- Queries that would normally fail (i.e. they would not execute) because of a type error will now succeed. They will probably return the "empty" answer.
- Users who misunderstand the implicit structure of the data will, as a consequence of the previous point, be further confused by the results of queries.
- Optimization techniques that are essential to efficient data integration are much more difficult to obtain with semistructured data.

In short, the use of semistructured formalisms and query languages, while they may simplify certain aspects of data integration will not eliminate the need for biologists to understand their data.

However the ability to incorporate some aspects of semistructured data into our integration systems is clearly useful for dealing with evolution and for browsing. One would like, for example, to add a field to some structure without having to change a schema and all the applications that depend on it. Equally one would like to ask questions of the database that return some structural information (e.g. "Where in the database is 'ubiquitin'?"). The internal structure of Kleisli/K2 will support this. Unlike many object-oriented and relational systems the internal representation of data in Kleisli and K2 is *dynamic*. This means that it is possible to add semistructured features to our existing query language or to support new query languages.

```
<entry GID="g2280496ID">
<AccNumber> AB005554 </AccNumber>
<Description> Bacillus subtilis genomic DNA... </Description>
<SeqLength> 36448 </SeqLength>
<Source> Bacillus subtilis </Source>
<References>
  <Reference>
    <Title> Organization and transcription of the gluconate operon... </Title>
    <Authors> <Author> Fujita,Y. </Author> <Author> Fujita,T. </Author> <Author> Miwa,Y. </Author>...
    </Authors>
  </Reference>...
</References>
<Sequence> ccaaaagcag... </Sequence>
<Features>
 <Gene>
  <From> 178 </From>
  <To> 1584 </To>
  <Complement> true </Complement>
  <Gene-string> gntZ </Gene-string>
 </Gene>
 <Source> ... </Source>...
</Features>
</entry>
```

Apart from providing a low-level data format, XML will not do more for us until there is an agreed schema technology. At the time of writing there is a bewildering variety of proposals for adding structure to XML and no standard even for the simple task of representing relational data in XML has yet been defined. It is to be hoped that some form of type system for XML will soon gain widespread acceptance.

List of Papers and Software

1. "A Data Transformation System for Biological Data Sources," P. Buneman, S.B. Davidson, K. Hart, C. Overton and L. Wong. *Proceedings of the 21st VLDB*, Sept. 1995.
2. "Morphing Sparsely Populated Data, " S. B. Davidson, C. Hara, A. Kosky and C. Overton. Meeting report, *Meeting on Interconnecting Molecular Biology Databases*, Cambridge England, July 1995.
3. "A Logic Grammar Based Approach to Transforming Data Beween Files and Biological Databases," S. Davidson, W. Fan and C. Overton. Meeting report, *Meeting on Interconnecting Molecular Biology Databases*, Cambridge England, July 1995.
4. "Challenges in Integrating Biological Data Sources," S. Davidson, C. Overton, and P. Buneman. *Journal of Computational Biology* Winter 1995 (1995).
5. "A Query Language and Optimization Techniques for Unstructured Data," P. Buneman, S. Davidson, G. Hillebrand and D. Suciu. *Proceedings of ACM SIGMOD International Conference on Management of Data* (May 1996).
6. "BioKleisli: A Digital Library for Biomedical Researchers," S. Davidson, C. Overton, V. Tannen and L. Wong. *Journal of Digital Libraries*, Nov 1997.
7. "Querying an Object-Oriented Database Using CPL," S. Davidson, C. Hara, and L. Popa. *Proceedings of the Brazilian Symposium on Databases* (1997).
8. "WOL: A Language for Database Transformations and Constraints," S. Davidson and A. Kosky. *Proceedings of the International Conference of Data Engineering* (1997).
9. "Semantics of Database Transformations," A. Kosky, S. Davidson and P. Buneman. In *Semantics of Databases*, edited by L. Libkin and B. Thalheim (Springer LNCS1358) (1998).
10. "BioKleisli," P. Buneman, J. Crabtree, S. Davidson, V. Tannen and L. Wong. Bioinformatics, edited by S. Letovsky (Kluwer Academic Publishers) (1998).
11. "Processing Updates on Complex Value Databases," H. Liefke and S. Davidson. *Information Resource Management Association International Conference* (1999).
12. "Specifying Updates in Biomedical Databases," H. Liefke and S. Davidson. *SSDBM'99* (1999).
13. "Efficient View Maintenance in XML Data Warehouses," H. Liefke and S. Davidson. Technical Report MS-CIS-99-27 (1999).

In addition, details about specific integration systems can be found as follows:

1. The K2 prototype is available at http://www.cbil.upenn.edu/K2*/k2web. This page contains information about the system, as well as demos of queries.
2. Information about Morphase can be found at http://www.cis.upenn.edu/~db/morphase/

Kleisli has been supplanted by K2 and is no longer available online. However, it continues to function in data integration projects within SmithKline Beecham, as well as forming the basis of the Tambis system at the University of Manchester, UK.

References

[1] C.J. Stoeckert, F. Salas, B. Brunk, and G.C. Overton "EpoDB: a prototype database for the analysis of genes expressed during vertebrate erythropoiesis". *Nucleic Acids Research*, 27(1), January 1999.

[1] M. Rusinkiewicz, A. Sheth, and G. Karabatis. "Specifying interdatabase dependencies in a multidatabase environment." *IEEE Computer*, December 1991.

[1] Peter Buneman, Leonid Libkin, Dan Suciu, Val Tannen, and Limsoon Wong. "Comprehension syntax." *SIGMOD Record*, 23(1): 87-96, March 1994.

[1] P. Buneman, S.B. Davidson, K. Hart, C. Overton, and L. Wong. "A data transformation system for biological data sources." *Proceedings of VLDB*, Sept 1995.

[1] P. Buneman, J. Crabtree, S.B. Davidson, C. Overton, V. Tannen, and L. Wong. "BioKleisli". In *Bioinformatics*, S. Letovsky, editor. Kluwer Academic Publishers, 1998.

[1] Susan Davidson, Christian Overton, Val Tannen, and Limsoon Wong. "Biokleisli: A digital library for biomedical researchers." *International Journal of Digital Libraries*, 1(1), April 1997.

[1] V. Tannen, J. Crabtree and S. Harker. "The K2 Information Integration System." Available at http://db.cis.upenn.edu.

[1] R. G. G. Cattell and D. K. Barry. "The Object Database Standard: ODMG 2.0", Morgan Kaufmann Publishers, 1997.

[1] T. Bray, J. Paoli, and C.M. Sperberg-McQueen. Extensible markup language (XML) 1.0. http://www.w3.org/TR/REC-xml

[1] Serge Abiteboul, Peter Buneman and Dan Suciu. Data on the Web: from Relations to Semistructured Data and XML. Morgan Kaufmann Publishers, 1999.

[1] Jean Thierry-Mieg and Richard Durbin, "ACeDB --- A C. elegans Database: Syntactic Definitions for the AceDB Data Base Manager." http://www.sanger.ac.uk/Software/Acedb/Acedocs/syntax.html.

[1] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu. "A query language and optimization techniques for unstructured data." *Proceedings of ACM-SIGMOD International Conference on Management of Data*, May 1996.

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom and J. Wiener. "The Lorel Query Language for Semistructured Data." *International Journal on Digital Libraries*, 1(1), April 1997, pp. 68-88.

[1] Stephen Deach, "Extensible Stylesheet Language (XSL) Specification." http://www.w3.org/TR/WD-xsl/

[1] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "XML-QL: A query language for XML." http://www.w3.org/TR/NOTE-xml-ql.

References

[1] C.J. Stoeckert, F. Salas, B. Brunk, and G.C. Overton "EpoDB: a prototype database for the analysis of genes expressed during vertebrate erythropoiesis". *Nucleic Acids Research*, 27(1), January 1999.

[2] M. Rusinkiewicz, A. Sheth, and G. Karabatis. "Specifying interdatabase dependencies in a multidatabase environment." *IEEE Computer*, December 1991.

[3] Peter Buneman, Leonid Libkin, Dan Suciu, Val Tannen, and Limsoon Wong. "Comprehension syntax." *SIGMOD Record*, 23(1): 87-96, March 1994.

[4] P. Buneman, S.B. Davidson, K. Hart, C. Overton, and L. Wong. "A data transformation system for biological data sources." *Proceedings of VLDB*, Sept 1995.

[5] P. Buneman, J. Crabtree, S.B. Davidson, C. Overton, V. Tannen, and L. Wong. "BioKleisli". In *Bioinformatics*, S. Letovsky, editor. Kluwer Academic Publishers, 1998.

[6] Susan Davidson, Christian Overton, Val Tannen, and Limsoon Wong. "Biokleisli: A digital library for biomedical researchers." *International Journal of Digital Libraries*, 1(1), April 1997.

[7] V. Tannen, J. Crabtree and S. Harker. "The K2 Information Integration System." Available at http://db.cis.upenn.edu.

[8] R. G. G. Cattell and D. K. Barry. "The Object Database Standard: ODMG 2.0", Morgan Kaufmann Publishers, 1997.

[9] T. Bray, J. Paoli, and C.M. Sperberg-McQueen. Extensible markup language (XML) 1.0. http://www.w3.org/TR/REC-xml

[10] Serge Abiteboul, Peter Buneman and Dan Suciu. Data on the Web: from Relations to Semistructured Data and XML. Morgan Kaufmann Publishers, 1999.

[11] Jean Thierry-Mieg and Richard Durbin, "ACeDB --- A C. elegans Database: Syntactic Definitions for the AceDB Data Base Manager." http://www.sanger.ac.uk/Software/Acedb/Acedocs/syntax.html.

[12] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu. "A query language and optimization techniques for unstructured data." *Proceedings of ACM-SIGMOD International Conference on Management of Data*, May 1996.

[13] S. Abiteboul, D. Quass, J. McHugh, J. Widom and J. Wiener. "The Lorel Query Language for Semistructured Data." *International Journal on Digital Libraries*, 1(1), April 1997, pp. 68-88.

[14] Stephen Deach, "Extensible Stylesheet Language (XSL) Specification." http://www.w3.org/TR/WD-xsl/

[15] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "XML-QL: A query language for XML." http://www.w3.org/TR/NOTE-xml-ql.