BioXRT: a novel platform for developing online biological databases based on the Cross-Referenced Tables model

Junjun Zhang[1], Gavin E. Duggan[1], Razi Khaja[1] and Stephen W. Scherer[1,2,*]

[1] Department of Genetics and Genomic Biology, The Hospital for Sick Children, Toronto, Ontario, Canada, M5G 1X8.

[2] Department of Molecular and Medical Genetics, University of Toronto, Toronto, Ontario, Canada, M5S 1A8.

*Corresponding author.

E-mail addresses:

Junjun Zhang:	jzhang@sickkids.ca
Gavin E. Duggan:	gavin@genet.sickkids.on.ca
Razi Khaja:	rkhaja@sickkids.ca
Stephen W. Scherer:	swscherer@sickkids.ca

ABSTRACT

BioXRT defines a data model termed Cross-Referenced Tables (XRT) and a software implementation to serve as a generic tool for quickly developing online biological databases. XRT is a text file based data modeling tool, its high flexibility and extensibility allow one to model and integrate dynamic biological datasets with relative ease. Open architecture and content-independence make BioXRT broadly applicable and highly adaptable to handle database expansion of unforeseen complex data without the need for software redevelopment.

1 RATIONALE

The rapid expansion of molecular biological knowledge through reduction in computing costs, spread of internet access, and the emergence of high-throughput genomic technologies has led to a rapid growth of electronically available molecular biology databases. For examples, for collections of locus-specific mutations alone, there were 262 databases as of 2002 [1]; and the 2005 updated *Nucleic Acids Research* online Molecular Biology Database Collection included 719 databases, an increase of 171 over the previous year, and this listing was far from exhaustive [2]. Online databases provide many advantages, such as wide-accessibility, advanced querying, fast retrieval and persistent referencing. More and more people in the biology research community (as well as in the financing bodies) now appreciate the importance of open databases in spreading knowledge [3]. However, for most research groups, setting up their own database of any significant size or complexity is arduous. Even when finished, a database needs to be updated regularly, and new data has to be parsed, indexed and integrated. So, despite the high desirability of an *in-house* database, the cost of developing and maintaining it can be prohibitive for a small research group. If a generic and easy to use database development platform could be used without sacrificing any significant functionality, it would substantially shorten the development cycle and reduce the cost of maintenance, and many more research groups would be able to set up their own online databases.

Although the content and the architecture among biological databases are quite different, many of they do share a common cycle of tasks including: (i) collecting and curating data from different sources such as public databases, scientific literature and internal laboratory results; (ii) integrating this information using an appropriate model; (iii) loading data into a relational database; and (iv) providing a web-based interface for users to query and browse the data in a read-only fashion. When updating with new data, they just go through the cycle over again. These common tasks make it practically feasible to build and maintain a biology database using a generic platform. By avoiding *in-house* development, a generic approach can prevent unnecessary duplication of effort.

Just like the Generic Genome Browser (GBrowse) [4], an open source project which is now gaining its popularity, can be easily adapted as the visualization tool for any location data, such as genome annotations. Inspired by GBrowse and based on a novel data modeling system, we developed BioXRT that aims to provide a light-weight generic solution for housing and publishing biological data. BioXRT consists of three major components: a data modeling methodology, a generalized database schema, and web-based data accessing and presentation facilities. BioXRT is available for download on the worldwide web [5].

2 SYSTEMS AND METHODS

2.1 Design objectives

As a generic platform, BioXRT was designed with the following basic features in mind: (i) ***broad applicability***; (ii) ***capability to handle complex relationships***, so that heterogeneous data can be easily integrated; (iii) ***high extensibility***, so new data types can be integrated later on without database redevelopment; (iv) ***relative ease of use***. In the rest of this paper, we will illustrate how these features are achieved and how the common tasks mentioned earlier are supported in BioXRT.

2.2 The Cross-Referenced Tables data model

A data model is a description of the data for a particular subject area, how they are defined and organized, and how they relate to one another. It includes the data items and their relationship. Taking the large diversity and fast emerging pace of biological data into account, a broadly applicable and extensible data model is essential for a generic approach to build biological databases. With this in mind, we developed a data modeling system termed Cross-Referenced Tables (XRT).

For simplicity, XRT model uses tab-delimited flat files (i.e. text tables) as basic modeling unit to keep data items. A text table structured by a field/value convention is the most natural format and is commonly used in many public biological data sources. It is capable of storing arbitrary data

items by simply adding new fields, and is generally applicable to any textual information. Additionally, no special tool is needed to prepare or parse a text table. However, it also has some caveats, such as lack of referencing and constraints, and difficulty in modeling complex data with a single table. To overcome these limitations, we applied several rules to the text tables, basically, injecting mechanism to handle relationship among data items.

XRT is a simple file schema, which encapsulates data in an object hierarchy with arbitrary attributes and relationships. It organizes data into different classes according to its biological meaning (eg. gene, Gene Ontology term and OMIM entry). Each class has as many attributes as necessary to describe the properties of its elements, and its attributes can be settled in one or more XRT tables. An XRT table is a tab-delimited flat file. The first line specifies the attribute names, while the following lines contain the actual attribute values for elements with each element having a unique identifier (ID, primary key in database term). Special attributes called P_ID for parent ID and C_ID for child ID keep track of references between data elements (in database terms, these relationships are known as foreign keys). The relationship can be one to one, one to many, many to one, or many to many. XRT class name is defined as the string before the first dot (.) of the XRT table file name, for example, the class name of XRT table *Transcript.main.xrt* is *Transcript*. Online documentation on detailed XRT specification is available at http://projects.tcag.ca/bioxrt/xrt_spec.html.

An example XRT model of the gene-centric data is shown in Figure 1a. As you can see, XRT model is very similar to the Entity-Relationship Model (ERM) on which relational database management systems (RDBMSs) are based; classes, attributes, data elements, unique identifiers and P_IDs/C_IDs are roughly equivalent to database tables, fields, records, primary keys and foreign keys respectively. This similarity ensures XRT would be, like RDBMSs, generally applicable to model potentially any kinds of data. More importantly, comparing to the relational model, there are two major differences of XRT: (i) Attributes are allowed to have multiple values (values are separated by "&;", such as the attribute *Source Seq* in Figure 1a), while only single value is accepted for a field of one record in an RDBMS, which will have to create a child table to keep the multi-valued field.

Multi-valued attributes P_ID and C_ID, also make it possible to model many to many relationship, which is often required, such as one gene could cause different diseases and one disease could be caused by many genes. (ii) Data in one XRT class can be settled in multiple tables, just make sure except the ID, no other attributes overlap among the tables. The benefits of this are that new table (with new attributes) can be added to an existing class later on without touching any existing table(s), and different tables (even for the same class) can be generated and maintained separately as long as appropriate referencing is kept. This feature allows database expansion of new data and facilitates integration of data from scattered sources. As shown in Figure 1a, the *Transcript* class encompasses two tables, *Transcript.main.xrt* and *Transcript.express.xrt*, with their data originated from internal curation and UCSC genome bioinformatics site [6] respectively. With these distinctive features, XRT allows users to model their data in a more natural way, thus it's easy to understand and use. Three more XRT model examples are provided online at http://projects.tcag.ca/bioxrt/xrt_spec.html. Actually XRT can be adapted to any descriptive data, including but not limited to biological datasets.

[**Figure 1**]

2.3 Generalized database structure for XRT data storing

To take advantages of the industry-standard database management system, file-based XRT data is loaded into an RDBMS, and a generalized database structure was designed to achieve maximum flexibility and unified data accessing and manipulation. Figure 1 shows the conversion of XRT tables from "horizontal" format to "vertical" format. Subsequently, the vertical tables can be merged into a single table (Figure 1b), such unified vertical table significantly simplifies the physical database schema design. Since no information is lost during the conversion and merging, XRT data in this physical schema can be easily converted back to its normal format. And more importantly, integrated conceptual schema (users' perception of the XRT modeled data, as illustrated in Figure 1c)

can be constructed from the physical schema using a style of database query known as a Cross Tab or Pivot query. In fact, to store data in the vertical format (Figure 1b), only four database tables are needed and their structures are shown in Figure 2. A similar database schema was used in a recent paper discussing molecular biological data integration [7], while in XRT, with the additional class layer, data is much easier to organize and manipulate. Actually such data structures are both derived from the Entity-Attribute-Value (EAV) model, which has been used in many biomedical applications with their data highly heterogeneous and sparse [8].

[**Figure 2**]

Storing XRT data in an EAV structure provides many advantages: (i) Data in EAV model can be accessed through a unified and content-independent way, upon which highly robust data retrieving application program interface (API) can be built. (ii) An arbitrary number of XRT classes (types of biological data) and their attributes can be added, and changes in classes and attributes can be made at any time without restructuring the database, consequently, no rebuilding of the data accessing API is needed. This level of flexibility is the feature most often requested in a biological database due to the large diversity of data types and the frequent emergence of new data types. As a result, data model needs to be dynamic to reflect the fast evolving biological knowledge. (iii) Contradictory to the conventional design, attributes with no value (NULL value in database term) will not waste database space, as we don't keep them in the EAV representation (such as the attribute *amygdala* in Figure 1a). This feature allows us to model certain type of data with a very wide XRT table, i.e. table with a very large number of attributes, without worrying about that some or many of the attributes would have no value in some data elements (rows/tuples). (iv) No administrative efforts are required to maintain the database structure. (v) From a technical standpoint, querying an EAV database for search terms is straightforward, as only one table has to be searched. Since the majority of the biological data

consists of free text, such as descriptions and comments, full text indexing can be used to provide more efficient query.

It's well known that the simplicity and compactness of EAV representation may be offset by a potential performance penalty compared to the equivalent conventional design. For example, the simple AND, OR and NOT operations on conventional data must be translated into the less efficient set operations of Intersection, Union and Difference respectively, which may have to be accomplished through multi-step SQL queries with intermediate results kept in temporary table(s). As well, some computing would be needed to construct the conceptual schema from the physical schema. However, given the powerful hardware and sophisticated database server we have nowadays, the performance impact may not be noticeable by the end user when the database is relatively small. Such as the LocusLink demonstration database (accessible at http://projects.tcag.ca/bioxrt/locuslink) modeled in XRT with about 11.7 million records in the *gdata* table, most of the Ad Hoc queries (user-specified query criteria to get the data of interest) took less than five seconds (given the limit of maximum 500 records per page). And the promised high performance of keyword search has been confirmed in the demonstration database, searching database for RefSeq accessions, gene symbols, or PubMed IDs, etc., usually would be responded within one second.

3 IMPLEMENTATION

While having data modeled in XRT, we developed the BioXRT platform to provide XRT data storage and web presentation. It is implemented in Perl, and is built exclusively upon open source components such as MySQL and BioPerl [9]. These choices reflect the easiest configuration to install, however, the schema and configuration are applicable to any permutation of platform factors and support will be provided to labs choosing to implement it in a different setup. An overview of the BioXRT platform is shown in Figure 3.

[**Figure 3**]

3.1 Data preparation, storage, and accessing API

To build a particular online database using BioXRT, first of all, we need to model the data in XRT, i.e. define classes and their relationships, such as the model shown in Figure 1. Then, data from either internal results or external sources is converted into XRT tables. A Perl script named "bulk_load_xrt.pl" can later transform all XRT tables to the vertical format, and load them into the database and build the requisite indices. For the default implementation, the MySQL database management system was used to host XRT data because of its open source status, its widespread use in the bioinformatics community, and its superior performance in *read-mostly* environments. Any SQL92 compliant database engine could be used with relative ease.

Access to the database at a low level is accomplished via a standard connection such as the Perl DBI, with an XRT-specific API which translates the data requests into appropriate SQL queries, and then returns the results or transforms the results into objects for further manipulation. For examples, querying all available classes, searching for a data entry with a certain ID. The XRT API can be easily embedded in user-developed software as an object, which provides more flexibility to manipulate the data. However, we expect most researchers will choose to publish their data immediately using the standard presentation tools included with BioXRT platform.

3.2 Data query interface and presentation tools

In order to provide an efficient, user-friendly and widely accessible interface to an XRT database, we have implemented a web application called TBrowse. The browser accesses the XRT database and converts results into HTML tables. Such output tables are pre-defined in a highly customizable configuration file. For a particular output table, main class, the focus of the output table, is specified first. And then columns are defined with the attributes of the main class and/or its related classes. In this way data can be integrated from different classes into one final output table. Figure 4

shows how the output table in Figure 1 can be defined. Several options can be customized to the output table (e.g. table title, column headers, and hyperlinks). For additional details about the table configuration, an online tutorial is provided at http://projects.tcag.ca/bioxrt/tutorial. In addition to browsing pre-defined tables, TBrowse also functions as a data retrieval tool, users can perform keyword searches, select columns to show and filter records on certain column(s) to obtain their data of interest. Actually, the result filtering is a simple version of Ad Hoc query tool, which provides a more powerful means to do complex queries by allowing users freely build their own searching criteria, i.e. combinations of column constraints. Currently the Ad Hoc query in TBrowse is very simple, better support in newer version is planned. Output of TBrowse can be exported and downloaded in several formats: tab-delimited flat file, XML and Microsoft Excel file. Besides the interactive web interface, URL-based access to the XRT database is also supported in TBrowse.

[**Figure 4**]

Due to the simplicity of a two dimension table, TBrowse is not entirely ideal in displaying data of complex structure. Another web application called XView was implemented, which can recursively handle (theoretically) unlimited levels of XRT relationship in a hierarchical structure. XView presents data in an easy-to-understand hierarchical tree reflecting the logical relationship of XRT data items, similar to TBrowse, the tree structure is defined in a user-managed configuration file. In order to highlight the capacity of XView to display complex data relationship, we re-implemented the NCBI's LocusLink database with BioXRT, and employed XView as the presentation tools. First, the entire LocusLink data (in file LL_tmpl.gz, downloaded from ftp://ftp.ncbi.nih.gov/refseq/LocusLink/) was modeled in XRT, i.e., parser was developed to convert LocusLink data into XRT model. Additionally, gene expression data from UCSC (in files gnfAtlas2.txt.gz and knownToGnfAtlas2.txt.gz at http://hgdownload.cse.ucsc.edu/goldenPath/hg16/database/) was integrated. In total, the resulting

XRT model consisted of 19 XRT tables representing 15 classes in three levels. With this data, a demonstration database was built and can be accessed at http://projects.tcag.ca/bioxrt/locuslink. The result was encouraging, this sample database demonstrates that the XRT model and XView presentation engine can easily handle this relatively large dataset and present the data in a well-organized fashion expediently.

3.3 BioXRT sample and proof of concept databases

Besides the sample database of LocusLink mentioned above, BioXRT has also been successfully applied in several of our online projects in a wide range including: the Human Chromosome 7 Annotation Project [10,11], the Genome Segmental Duplication Project [12,13], the Autism Chromosome Rearrangement Database [14,15], the Database of Genomic Variants [16,17], the Genomic Clone Database [18], and the Lafora Progressive Myoclonus Epilepsy Mutation Database [19,20]. Within the chromosome 7 database, BioXRT is the primary harness for gene-centric data that are derived from diverse sources. There are currently 21 XRT tables representing 18 classes, each table can be maintained individually, even by different curators, this working mode matches the distributed nature of the data sources. When new data needs to be integrated, it is simply converted into XRT format while referencing existing data correctly, and the configuration file is updated. After being uploading, the new data gets integrated automatically, no database structure needs to be changed, and no program has to be modified.

4 DISCUSSION

4.1 Modeling biological data with XRT

With the BioXRT platform available, to set up an online biological database becomes a lot easier; no worry about database schema design, and developing programs for data query and presentation. The only thing users have to do is to model their data in XRT, which is like a simplified

version of the relational database schema design, since only the logical design phase is involved, and no normalization or other physical design concerns are needed.

The most distinctive feature of online biological databases that greatly affects data modeling is that they focus on knowledge spreading, data presentation, ultimately data is provided in a read-only fashion, and in this sense they are like data warehouses regardless how big the size is. This makes biological data quite different from the operational data, for instance, in a *transaction-richened* banking database where updating happens extensively. So a good design should fully take advantage of the read-only characteristic. One major step of a relational database design is normalization which usually will create more tables in order to reduce data redundancy and improve efficiency of data integrity enforcement. But if the database is built to be read-only, data integrity won't be broken once data has been loaded. And some data redundancy shouldn't matter too much, again because data is read-only; no updating of the redundant data copies is needed. So normalization that sometimes is tedious is not required when modeling data in XRT, major effort can then be focused on the biological meaning and logical relationship of the data itself, i.e., the conceptual level. For example, in a relational model design, the *Gene* table in Figure 1, A should be normalized so that the *Type* field is kept in an additional table, since there are very limited *types* of gene. While modeling data in XRT, such moralization is unnecessary, and sometimes is even troublesome due to the increased complexity. When well designed, XRT model can be a naturally denormalized schema, which is often pursued in data warehouse design attempting to simplify the database schema, and improve the understandability and query performance [21].

Basically, every biological database has a subject or focus, XRT modeling starts with identifying this subject. Information (attributes) that directly associated with the subject would go to the main XRT class, and then from the main class we can drill down to other level classes. For example, in the Genomic Clone Database [18], *Clone* is the main class, and its attributes may include ID, clone name, alias, plate number, row and column etc. One clone must be belong to one library, and one library contains many clones, then information about a clone library, such as name, type,

developer etc., should be included in the *Library* class, which should be the parent class of the *Clone* class. Sequence data of a clone can be kept in the *Genbank_clone* and *End_seq* classes, which are child classes of the *Clone*. Other information, such as, genes, genetic markers, FISH mapping etc., can be added as well.

Biological data is rarely static due to the fast pace of new data emergence, change is unavoidable no matter which modeling tool has been used; quite efforts are needed for data re-modeling. The advantages of XRT's simplicity stand out while handling data model changes, which actually was the initial motivation of the BioXRT project, modification (adding, changing and deleting) of the XRT classes and/or their attributes can be easily done through the updating of XRT tables. More importantly, due to the content-independency of the BioXRT platform, no effort is needed for database or program re-engineering to accommodate the updated XRT model. As thus, XRT model is highly flexible and broadly applicable, and the reusability of the BioXRT platform is maximized.

In the XRT model, data type is loosely controlled; currently everything is kept as text (as a benefit of the auto data type conversion provided in MySQL, numeric comparison is supported for text fields in an SQL statement). As a result, some data may not be optimized for better accessing, for example, the location of a genomic feature, which usually includes start and end coordinates of its parent feature (e.g., chromosomes). Without special handle, querying features located in a specific sub-region of their parent feature would be less efficient. One feasible solution is to predefine some build-in data types, and optimize their storing and accessing design accordingly. In the case of location data, the optimizing algorithm used in GBrowse [4] can be easily adapted. Importantly, all these can be done almost transparently to the end users, i.e., only very minor change is needed while modeling data with XRT – specifying data type for each attribute.

4.2 Unique identifiers for XRT data entries

Cross-referencing of XRT tables is the foundation of data integration exercises, and as with any unique key, the identifier of data entries should be unique to facilitate cross-referencing. There are two types of identifiers – internal identifiers which are purely used for internal data referencing among related classes, and then "published" but still internal referencing identifiers which can also be tracked by the outside world. Such externally accessible identifiers should be kept stable and database-wide unique so that biologists can track a particular biological object over time, and stable identifiers also facilitate inter-database integration over the internet [22]. *Internal-use-only* identifier can be arbitrarily assigned (still has to be unique at least within its class, and starting with a "_" is suggested). For the "published" ones, we strongly recommend using systematic identifiers consisting of one or more letters of prefix for each XRT class plus a fixed-length of numbers (eg. GA0001 for a gene, T01206 for a transcript, and DP00458 for a duplicon pair). This approach is similar to that used by GenBank and other large public databases. Some external data may already have identifiers, for example, NCBI's LocusLink. While this external data being converted into XRT, identifiers should be reassigned complying with your current nomenclature, such as the one suggested above, then _LL0000107 would be a good internal identifier for locus 107.

4.3 Comparison with other systems

There are some other biological database systems provide support for functionality that resembles some aspects of BioXRT. The most popular among these are BioMart at EBI [23], and Chado, the modular schema of GMOD [24]. BioMart, formerly known as EnsMart [25], was designed to be a simple, federated query system for large biological datasets. The architecture of BioXRT and BioMart is very similar, data from different sources is gathered and transformed at a domain-specific staging area (modeling data with XRT in the case of BioXRT), and then domain-independent facilities are provided for data storing and accessing. Such domain-independency makes both systems extensible and adaptive to a broad range of biological datasets. The database design of BioMart was

based on the dimensional model, specifically the star schema, which has been proved to be very successful in terms of structural simplicity and query performance in many commercial data warehouse implementations. Consequently, the key feature of BioMart is high performance of large-scale data querying. On the other hand, opposite to BioXRT, BioMart has very limited capability to model data with complex relationship, for example the LocusLink data. In BioMart, every query against the dimensionally modeled data can only join one central table with one or few dimensional tables, and then generates the output in a tabular format; this is very similar to BioXRT's TBrowse which constructs the integrated output from a chosen main class and its related classes. In both cases, only two levels of relationship are involved, which is not sufficient for complex data with more levels of relationship, for example in a gene-centric database, one gene can have many splicing forms of transcripts, each transcript encodes one protein, a protein may have many domains with each domain annotated with multiple functions. Additionally, such complex data structure would be hardly illustrated in a two-dimensional tabular format. XView, provided in BioXRT as discussed earlier, is an elegant solution with the capability to present the whole picture of the complex dataset. While for BioMart, it would have to build multiple marts with each one focusing on the object at a certain level, which is a tedious work, and even implemented, it can only provide non-integrated sub-area snapshots of the whole dataset.

The Generic Model Organism Project (GMOD) is a collaborative project to develop reusable components suitable for creating new community databases of biology. Chado is the modularized data schema of GMOD, with each schema module specially designed to handle data from a specific domain, such as sequence, expression, ontology, publication and general feature etc. Since the modules are predefined, the flexibility is limited; extension or modification would be expensive, with the exception of the general module which is extensible in a certain degree. Moreover, as data from a new biological domain (e.g. protein interactions) to be included, new data module along with its manipulating programs have to be designed and implemented, meanwhile special care has to be taken

to ensure the interoperability with the existing modules. Contradictorily, with its open structure, BioXRT can be easily extended to accommodate new data types.

4.4 BioXRT and biological database integration

BioXRT provides several features to facilitate data integration as described earlier; however, it's not attempting to be a warehousing solution for large-scale biological data integration. Instead, BioXRT mainly focuses on a universal data structure; through it integrated data can be accommodated and accessed. As a data modeling tool, XRT is not that different from XML, ANS.1 or ERM, it's hard to believe that many issues biological data integration has to deal with, such as reconciling discrepancies among databases, bio-entity name inconsistency, ontology and semantics etc., can be solved by simply choosing a *more proper* modeling tool. Honestly, without a series of standardizations (including nomenclatures and ontologies), these kinds of problem can never be handled automatically. Before achieving some sort of standardization, whichever modeling tool you choose, a lot work has to be done manually by data modelers and curators with a high level of domain-specific knowledge.

One major problem of biological data integration is the large amount of databases with almost each one having its own data structure and interface. If a universal database platform with unified accessing interface, such as BioXRT, could be used widely to implement the mid-sized domain-specific databases, then the web service-based federation approach to integrate these data sources would become a lot easier [22].

4.5 Maintenance of BioXRT databases

BioXRT was developed to be a generic data presentation platform for integrated biological data modeled in XRT; it works similar as data warehousing databases in terms of creating and maintaining procedures. Maintenance of BioXRT databases is performed at the XRT table preparing

stage (*staging area* in data warehouse term). Any time updating (modification, insertion, and deletion) is needed, an appropriate change should be made to the related XRT table(s). At the time we decide to release a new version of the database which contains the updates have been made since the last version, we load all the XRT tables into a new BioXRT database, and reconfigure few settings, then the new version would become live. If preferred, XRT tables for the previous version can be archived.

Typically, there are two types of XRT table categorized by the way they are created and maintained. The first type is XRT tables generated by user-written programs. For example, specific parsers are needed to transform data from external sources into XRT tables, and ensure the cross-referencing. Since XRT model is easy to understand, and no special tool is needed to generate a XRT table (flat file), such parsers should be relatively easy to write. To keep this type of XRT tables updated, parser(s) should be run routinely, and in case parser stops working due to the structure change of the source data, it should be modified accordingly.

The other type of XRT tables is primarily human-maintained. For example, gathering data from scientific literature, which normally can't be substituted with a computer program. In this circumstance, tools may be needed to facilitate the curation, a spreadsheet software (e.g., MS Excel) should be good enough for simple cases. Better maintenance tools are under development.

4.6 Future work

BioXRT is an ongoing project, and is still at its early stage. We are looking to enhance it with additional features. Items to be implemented in the near future include (i) optimizing/adding the ability to handle some special data types, e.g., location data (coordinates), expression data and images; (ii) plug-in modules providing additional flexibility to visualize data in a user-defined manner (graphics, for example, would be a better way than numerical listings to present gene expression data, in this case, an external module can be plugged in to generate the graphics on the

fly); (iii) ability to generate default configuration automatically, and a user-friendly configuration editor; (iv) a maintenance tool providing a graphic user interface to maintain user-selected XRT tables, and enforcement of the cross-references; (v) better support for Ad Hoc query in TBrowse; (vi) performance and scalability improvement, one effective way would be data table (*gdata* in Figure 2) partitioning in the EAV structure, and non-EAV persistent implementation of XRT data model is also under consideration.

BioXRT is applicable to any textual information, and can integrate heterogeneous data sources. As such, it can serve as either a leaf node in a collaborative data network, or as an engine for tying together disparate sources. Our long-term plan is to implement BioXRT as a SOAP server (similar to a DAS server [26]) which provides a *program-friendly* interface to facilitate data integration among distributed BioXRT databases. Moreover, in order to automate data integration, we will develop tools for exporting BioXRT to BioMOBY, an open source project aims to provide an architecture for the discovery and distribution of biological data through web services [27].

5 CONCLUSIONS

BioXRT is novel in several ways. First, to our knowledge, it's the first effort trying to provide a general-purpose, quick and re-useable solution for developing online biological databases. Next, the XRT data modeling system is specially designed, and suitable for biological data and other descriptive data. Then, the generalized database schema design and the high configurability make the whole system content-indepentent. Finally, BioXRT is extremely flexible and gracefully extensible to accommodate change, which makes developing an online biological database much more manageable even for small development teams; large database can be built in a phase-by-phase manner, changing of the existing data and integrating new data can be easily done over time. Overall, BioXRT provides a measure of simplicity and ease of use suitable for niche applications in the research community. In particular, BioXRT is an excellent platform for small and medium size labs with *in-house* results they wish to share online in conjunction with, and correlated to, data from other public sources. The

system's simple setup allows a database to be brought online quickly, and its flexible schema can manage database expansion of unforeseen complex data without the need for database or software redevelopment.

We believe the light-weight approach presented here is an attractive solution for biological data sharing. This open source initiative was developed with two missions. One, to allow biologists the ability to quickly bring their research data online, where data is widely accessible throughout the world. And secondly, to invite outside developers the opportunity to contribute their own ideas and requirements to enhance BioXRT's ability to accomplish biological goals.

ACKNOWLEDGMENTS

The authors would like to thank Weimin Zhu and Charles Lee of the European Bioinformatics Institute, and Joseph Cheung, Jeffery MacDonald and Cheng Qian of The Centre for Applied Genomics for their valuable comments and kind assistances. The work is supported by Genome Canada, the McLaughlin Centre for Molecular Medicine, and the Hospital for Sick Children Foundation. S.W.S. is an investigator of the Canadian Institutes of Health Research and an International Scholar of the Howard Hughes Medical Institute.

REFERENCES

1. Claustres M, Horaitis O, Vanevski M, Cotton RGH: **Time for a Unified System of Mutation Description and Reporting: A Review of Locus-Specific Mutation Databases.** *Genome Research* 2002, **12**:680-688.

2. Galperin MY: **The Molecular Biology Database Collection: 2005 update.** *Nucleic Acids Research* 2005, **33**:D5-D24.

3. Ball CA, Sherlock G, Brazma A: **Funding high-throughput data sharing.** *Nature biotechnology* 2004, **22**:1179-1183.

4. Stein LD, Mugall C, Shu S, Caudy M, Mangone M, Day A, Nickerson E, Stajich JE, Harris TW, Arva A, Lewis S: **The Generic Genome Browser: A Building Block for a Model Organism System Database.** *Genome Research* 2002, **12**:1599-1610.

5. **BioXRT** [http://projects.tcag.ca/bioxrt]

6. **UCSC Genome Bioinformatics** [http://genome.ucsc.edu]

7. Philippi S: **Light-weight integration of molecular biological databases.** *Bioinformatics*, 2004, **20**:51-57.

8. Marenco L, Tosches N, Crasto C, Shepherd G, Miller PL, Nadkarni PM: **Achieving evolvable Web-database bioscience applications using the EAV/CR framework: recent advances.** *J Am Med Inform Assoc* 2003, **10**:444-453.

9. **BioPerl** [http://www.bioperl.org]

10. Scherer SW, Cheung J, MacDonald JR, Osborne LR, Nakabayashi K, Herbrick JA, Carson AR, Parker-Katiraee L, Skaug J, Khaja R *et al*.: **Human chromosome 7: DNA sequence and biology.** *Science* 2003, **300**:767-772.

11. **The Chromosome 7 Annotation Project** [http://www.chr7.org]

12. Cheung J, Estivill X, Khaja R, MacDonald JR, Lau K, Tsui LC, Scherer SW: **Genome-wide detection of segmental duplications and potential assembly errors in the human genome sequence.** *Genome Biology* 2003, **4**:R25

13. **Human Genome Segmental Duplication Database** [http://projects.tcag.ca/humandup]

14. Xu J, Zwaigenbaum L, Szatmari P, Scherer SW: **Molecular Cytogenetics of Autism.** *Current Genomics* 2004, **5**:347-364.

15. **The Autism Chromosome Rearrangement Database** [http://projects.tcag.ca/autism]

16. Iafrate AJ, Feuk L, Rivera MN, Listewnik ML, Donahoe PK, Qi Y, Scherer SW, Lee C. **Detection of large-scale variation in the human genome**. *Nat Genet* 2004, **36(9)**:949-951.

17. **The Database of Genomics Variants** [http://projects.tcag.ca/variation]

18. **The Genomic Clone Database** [http://projects.tcag.ca/gcd]

19. Ianzano L, Zhang J, Chan EM, Zhao XC, Lohi H, Scherer SW, Minassian BA. **Lafora progressive myoclonus epilepsy mutation database-EPM2A and NHLRC1 (EMP2B) genes.** *Human Mutation*, 2005, **26(4)**:397.

20. **The Lafora Progressive Myoclonus Epilepsy Mutation and Polymorphism Database** [http://projects.tcag.ca/lafora]

21. Kimball R, Ross M: **XXXX XXX XXX.** In *The data warehouse toolkit: the complete guide to dimensional modeling. Chapter ?.* 2nd edition. New York: publisher; 2002:??-??.

22. Stein LD **Integrating Biological Databases.** *Nature Reviews* 2003, **4**:337-345.

23. **BioMart** [http://www.biomart.org]

24. **GMOD Modular Schema – Chado** [http://www.gmod.org/schema]

25. Kasprzyk A, Keefe D, Smedley D, London D, Spooner W, Melsopp C, Hammond M, Rocca-Serra P, Cox T, Birney E: **EnsMart: A Generic System for Fast and Flexible Access to Biological Data.** *Genome Research* 2004, **14**:160-169.

26. Dowell RD, Jokerst RM, Day A, Eddy SR, Stein L: **The distributed annotation system.** *BMC Bioinformatics* 2001, **2**:7-13.

27. Wilkinson MD, Links M: **BioMOBY: an open source biological web services proposal.** *Brief Bioinform* 2002, **3**:331-341.

FIGURE LEGENDS

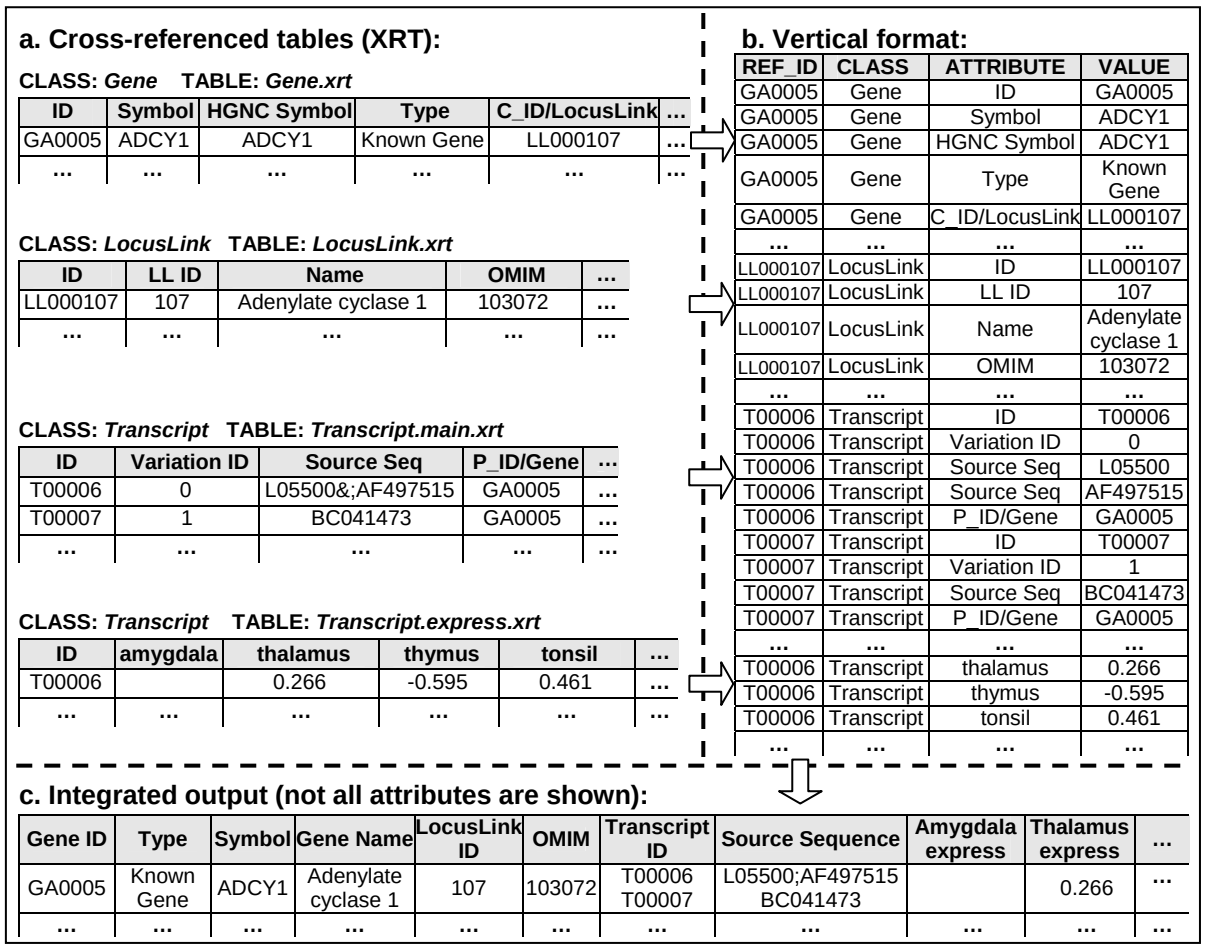
Figure 1: XRT example and its format transformation. (a) Four cross-referenced tables, XRTs; (b) The vertical format of the original XRTs; (c) Integrated output of the source XRT tables. This figure illustrates that XRT tables can be converted into a unified vertical format, data in this vertical format can later be converted back to a human readable table which integrates the original XRT tables. The physical database schema was designed basing on the vertical format.

Figure 2: XRT database schema. *gdata* is the central table which contains all attribute/value pairs; *gclass*, *gattribute*, and *grelationship* store the classes, attributes, and tracks the relations between data items in gdata, respectively.

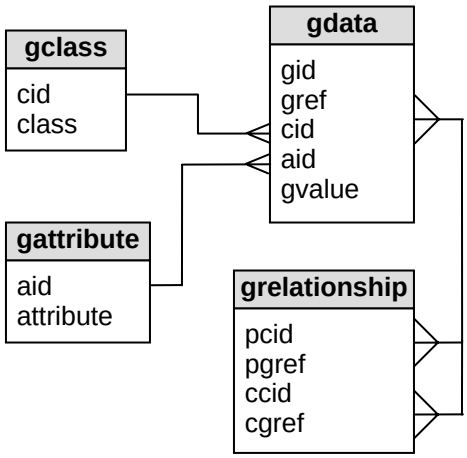
Figure 3: Overview of the BioXRT platform.

Figure 4: Defining an integrated output table in a TBrowse configuration file. In this example, the main class is *Gene*, and ten columns are derived from three related classes.

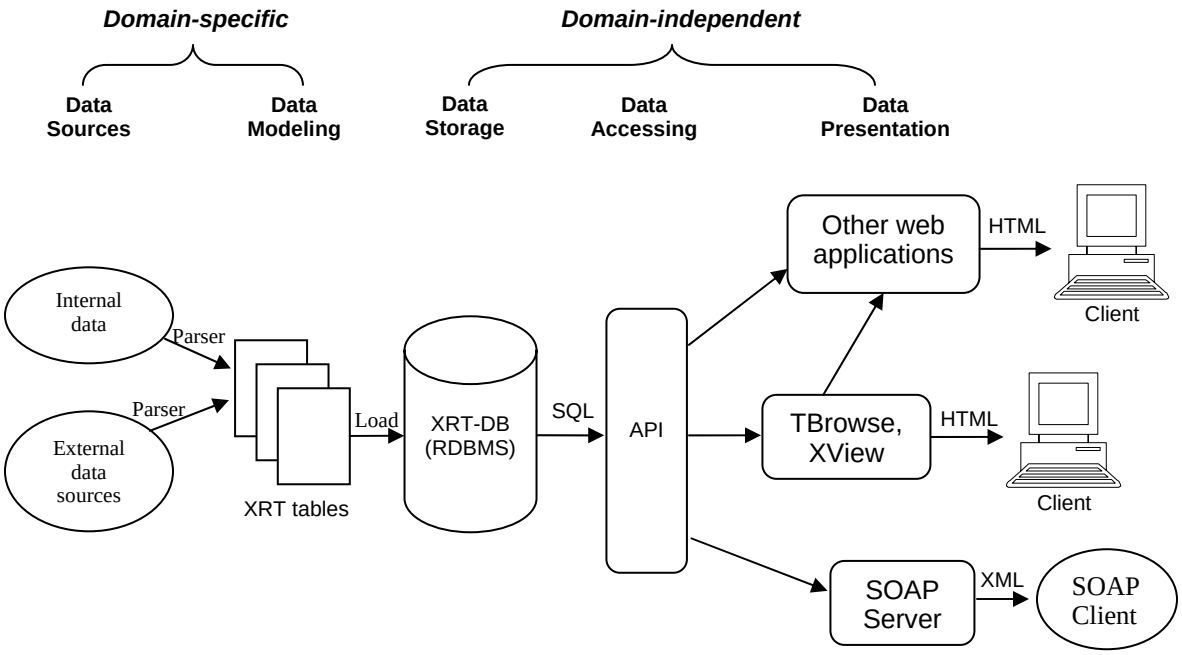
[Figure 1]


a. Cross-referenced tables (XRT):
CLASS: Gene TABLE: Gene.xrt
ID | Symbol | HGNC Symbol | Type | C_ID/LocusLink | ...
GA0005 | ADCY1 | ADCY1 | Known Gene | LL000107 | ...
CLASS: LocusLink TABLE: LocusLink.xrt
ID | LL ID | Name | OMIM | ...
LL000107 | 107 | Adenylate cyclase 1 | 103072 | ...
CLASS: Transcript TABLE: Transcript.main.xrt
ID | Variation ID | Source Seq | P_ID/Gene | ...
T00006 | 0 | L05500&;AF497515 | GA0005 | ...
T00007 | 1 | BC041473 | GA0005 | ...
CLASS: Transcript TABLE: Transcript.express.xrt
ID | amygdala | thalamus | thymus | tonsil | ...
T00006 | | 0.266 | -0.595 | 0.461 | ...
b. Vertical format:
REF_ID | CLASS | ATTRIBUTE | VALUE
GA0005 | Gene | ID | GA0005
GA0005 | Gene | Symbol | ADCY1
GA0005 | Gene | HGNC Symbol | ADCY1
GA0005 | Gene | Type | Known Gene
GA0005 | Gene | C_ID/LocusLink | LL000107
LL000107 | LocusLink | ID | LL000107
LL000107 | LocusLink | LL ID | 107
LL000107 | LocusLink | Name | Adenylate cyclase 1
LL000107 | LocusLink | OMIM | 103072
T00006 | Transcript | ID | T00006
T00006 | Transcript | Variation ID | 0
T00006 | Transcript | Source Seq | L05500
T00006 | Transcript | Source Seq | AF497515
T00006 | Transcript | P_ID/Gene | GA0005
T00007 | Transcript | ID | T00007
T00007 | Transcript | Variation ID | 1
T00007 | Transcript | Source Seq | BC041473
T00007 | Transcript | P_ID/Gene | GA0005
T00006 | Transcript | thalamus | 0.266
T00006 | Transcript | thymus | -0.595
T00006 | Transcript | tonsil | 0.461
c. Integrated output (not all attributes are shown):
Gene ID | Type | Symbol | Gene Name | LocusLink ID | OMIM | Transcript ID | Source Sequence | Amygdala express | Thalamus express | ...
GA0005 | Known Gene | ADCY1 | Adenylate cyclase 1 | 107 | 103072 | T00006 T00007 | L05500;AF497515 BC041473 | | 0.266 | ...

[Figure 2]


gclass
cid
class
gdata
gid
gref
cid
aid
gvalue
gattribute
aid
attribute
grelationship
pcid
pgref
ccid
cgref

[Figure 3]


Domain-specific
Domain-independent
Data Sources
Data Modeling
Data Storage
Data Accessing
Data Presentation
Internal data
External data sources
Parser
Parser
XRT tables
Load
XRT-DB (RDBMS)
SQL
API
Other web applications
HTML
Client
TBrowse, XView
HTML
Client
SOAP Server
XML
SOAP Client

[Figure 4]

```
[TCAG_Gene]
view_id    = V0001
title      = TCAG annotated genes on chromosome 7
main_class = Gene
Column1.   = 0::ID::Gene ID::/cgi-bin/geneview?id=*
Column2.   = 0::Type
Column3.   = 0::Symbol
Column4.   = LocusLink::Name::Gene Name
Column5.   = LocusLink::LL ID::LocusLink ID
Column6.   = LocusLink::OMIM
Column7.   = Transcript::ID::Transcript ID
Column8.   = Transcript::Source Seq::Source Sequence
Column9.   = Transcript::amygdala::Amygdala express
Column10.  = Transcript::thalamus::Thalamus express
```