ETLs for importing NCBI Entrez Gene, miRBase, mirCancer and microRNA into a bioinformatics graph database

A. Messina

Rapporto Tecnico N.:
RT-ICAR-PA-15-08

Dicembre 2015

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

ETLs for importing NCBI Entrez Gene, miRBase, mirCancer and microRNA into a bioinformatics graph database

A. Messina[1]

Rapporto Tecnico N.:
RT-ICAR-PA-15-08

Dicembre 2015

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo, Viale delle Scienze edificio 11, 90128 Palermo.

Index

1 Introduction

This work is the first of a series of technical report documenting the performed activities to build a big bioinformatics database.

Current available bioinformatics databases provide huge amounts of different biological entities such as genes, proteins, diseases, microRNA, annotations, literature references. But in many case studies, a bioinformatician often needs more than one type of resource in order to full analyze his data.

The bioinformatics database object of this work will allow the integration of different types of data sources, so that it is possible to perform bioinformatics analysis using only one comprehensive system.

The integrated database will be structured as a NoSQL graph database, based on the OrientDB platform, exploiting this way the advantages of that technology in terms of scalability and efficiency with regards to traditional SQL database.

The technical report is organized as follow: Section 2 presents a brief overview on the noSQL engine OrientDB; Section 3 presents the general structure of the developed ETLs; Sections from 4 to 7 report the specific ETLs implementations for the bioinformatics databases actually imported.

2 OrientDB

2.1 Introduction

OrientDB [1] is an open source NoSQL database management system (DMBS) developed in Java by Orient Technologies LTD. It collects features of document databases and graph databases, including object orientation. In graph mode, referenced relationships are like edges, accessible as first-class objects with a start vertex, end vertex, and properties. This interesting feature let us represent a relational model in document-graph model maintaining the relationships.

2.2 Brief features overview

The OrientDB engine supports Graph, Document, Key-Value, and Object models, so it is possible to use OrientDB as a replacement for a product in any of these categories.

However, the main feature of OrientDB is its ability to act as a true Multi-Model DBMS by combining all the features of the four models into one. Therefore, the database engine is built to support all four models, as shown in Figure 1. This is also the main difference with other Multi-Model DBMSs, since they implement an additional layer with an API that performs additional models.

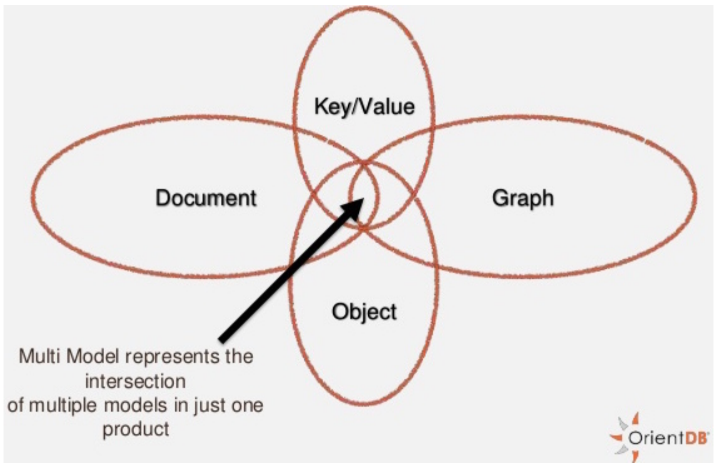

Key/Value
Document
Graph
Object
Multi Model represents the intersection of multiple models in just one product
OrientDB

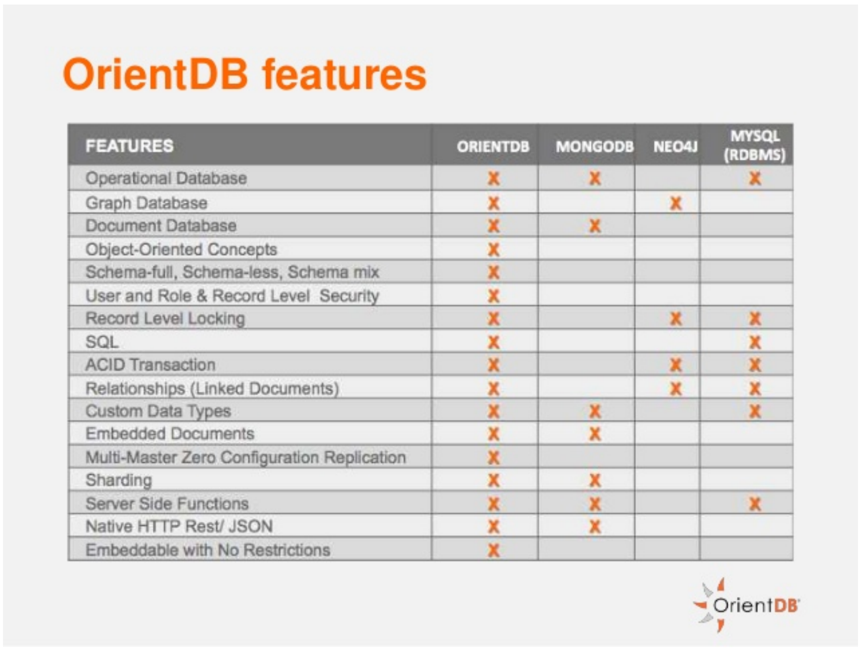
Figure 1 - OrientDB as a multi model noSQL database (courtesy of Orient Technologies LTD)

OrientDB is extremely versatile, as it includes features from relational databases, object oriented engines, document databases and, of course, graph models.

It is also capable of storing and serving records as JSON documents and performs very well thanks to its indexing algorithm, named MVRB-Tree. It provides a wide variety of different and powerful features: for example, it has an object- oriented model to exposing a REST interface; it is able to traverse a graph of thousands records, at any level of depth, in milliseconds; it provides to the developers a toolset and a variety of functionalities that they can never take advantage of with other database management system.

Finally, OrientDB Server is a customizable platform to build powerful server component and applications. Since the OrientDB server contains an integrated Web Server, it is possible to create server side applications without the need to have a J2EE and Servlet container.

By extending the server, it is possible to benefit of the best performance because you don't have many layers but the database and the application reside on the same JVM without the cost of the network and serialization of requests. Furthermore, it is possible to package your application together with the OrientDB server to distribute just a ZIP file containing the entire Application, Web server and Database.



OrientDB features

FEATURES	ORIENTDB	MONGODB	NEO4J	MYSQL (RDBMS)
Operational Database	X	X		X
Graph Database	X		X	
Document Database	X	X		
Object-Oriented Concepts	X			
Schema-full, Schema-less, Schema mix	X			
User and Role & Record Level Security	X			
Record Level Locking	X		X	X
SQL	X			X
ACID Transaction	X		X	X
Relationships (Linked Documents)	X		X	X
Custom Data Types	X	X		X
Embedded Documents	X	X		
Multi-Master Zero Configuration Replication	X			
Sharding	X	X		
Server Side Functions	X	X		X
Native HTTP Rest/ JSON	X	X		
Embeddable with No Restrictions	X			

OrientDB

Figure 2 - OrientDB features (courtesy of Orient Technologies LTD)

3 General ETLs Template

3.1 Introduction

All the developed ETLs are coded following the same general model, which basically supposes a subdivision in three parts:

- Initialization part
- Main-loop part
- Finalization part

Some details on the operations performed in these parts are exposed in the next sections.

3.2 Initialization part

As suggested by its name, all the preliminary initialization steps are performed in that code section.

It always starts with the connection to the database. Given that in this phase the database is going to be populated and in order to minimize writing times, a non-transactional connection is created. This means that every non idempotent operation against the graph database is atomic. It is the suggested way to massive import graph into OrientDB.

```
OrientGraphFactory graphFactory = new OrientGraphFactory(dbUrl);
OrientGraphNoTx graph = graphFactory.getNoTx();
graph.declareIntent(new OIntentMassiveInsert());
```

Depending on the bio entities to import, some basic objects are created in the database:

- one or more vertices types, with *graph.createVertexType(...)*
- one or more edges types, with *graph.createEdgeType(...)*
- one or more indexes, with *graph.createKeyIndex(...)*

Finally, a reader of the source file is created, usually with:

```
BufferedReader reader = new BufferedReader(new FileReader(fileName));
```

3.3 Main loop part

The main loop part is generally constituted by a *while* loop.

For textual tab-delimited source files, it runs until all the lines are read. In order to extract the columns values, every single line is then splitted into pieces, stored in a string array, which will contain the properties of a new vertex or a new edge:

```
while ((line = reader.readLine()) != null) {
    String datavalue[] = line.split("\t");

    …

}
```

The ETL section of the next chapters will present a detailed exposition of the operations performed in this part.

3.4 Finalization part

The last part of an ETL simply contains the necessary code to free the used resources:

```
reader.close();

graph.declareIntent(null);
graph.shutdown();

graphFactory.close();
```

4 NCBI Entrez Gene

4.1 Introduction

The NCBI Entrez Gene [1] is a searchable database of genes, focusing on genomes that have been completely sequenced and that have an active research community to contribute gene-specific data. Information includes nomenclature, chromosomal localization, gene products and their attributes (e.g., protein interactions), associated markers, phenotypes, interactions, and links to citations, sequences, variation details, maps, expression reports, homologs, protein domain content, and external databases.

4.2 Data source

The NCBI Entrez Gene database is available for download from the address ftp://ftp.ncbi.nih.gov/gene/ and it is splitted in several compressed files updated daily within the DATA directory. In this work, just the file gene_info.gz is considered; it is a textual tab-delimited file, containing a gene per line, and its structure (columns meaning) is reported below:

tax_id	the unique identifier provided by NCBI Taxonomy for the species or strain/isolate
GeneID	the unique identifier for a gene
Symbol	the default symbol for the gene
LocusTag	the LocusTag value
Synonyms	bar-delimited set of unofficial symbols for the gene
dbXrefs	bar-delimited set of identifiers in other databases for this gene. The unit of the set is database:value. Note that HGNC and MGI include 'HGNC' and 'MGI', respectively, in the value part of their identifier. Consequently, dbXrefs for these databases will appear like: HGNC:HGNC:1100 This would be interpreted as database='HGNC', value='HGNC:1100' Example for MGI: MGI:MGI:104537 This would be interpreted as database='MGI', value='MGI:104537'
chromosome	the chromosome on which this gene is placed. For mitochondrial genomes, the value 'MT' is used.
map location	the map location for this gene
description	a descriptive name for this gene
type of gene	the type assigned to the gene according to the list of options provided in http://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/lxr/source/src/objects/entrezgene/entrezgene.asn
Symbol from nomenclature authority	when not '-', indicates that this symbol is from a nomenclature authority

Full name from nomenclature authority	when not '-', indicates that this full name is from a nomenclature authority
Nomenclature status	when not '-', indicates the status of the name from the nomenclature authority (O for official, I for interim)
Other designations	pipe-delimited set of some alternate descriptions that have been assigned to a GeneID. '-' indicates none is being reported.
Modification date	the last date a gene record was updated, in YYYYMMDD format

4.3 ETL

In this work, just the genes related to the human species are considered. By comparing the value of the first column to the value '*9606*', all the non-human genes can be immediately skipped. Each other columns values are then assigned to strings, which will contain the values of the related properties of the new vertex:

```
while ((line = reader.readLine()) != null) {
    String datavalue[] = line.split("\t");

    String taxId = datavalue[0];
    if (!taxId.equals("9606"))
        continue;

    String geneId = datavalue[1];
    //String symbol = datavalue[2];
    String locusTag = datavalue[3];
    //String synonyms = datavalue[4];
    //String dbXrefs = datavalue[5];
    String chromosome = datavalue[6];
    String mapLocation = datavalue[7];
    String description = datavalue[8];
    String geneType = datavalue[9];
    String nomenclatureAuthSymbol = datavalue[10];
    String nomenclatureAuthFName = datavalue[11];
    String nomenclatureStatus = datavalue[12];
    String otherDesignations = datavalue[13];
    //String modificationDate = datavalue[14];

    graph.addVertex("class:gene",
        "geneId", geneId,
        //"taxId", taxId,
        //"symbol", symbol,
        "locusTag", locusTag,
        //"synonyms", synonyms,
        //"dbXrefs", dbXrefs,
        "chromosome", chromosome,
        "mapLocation", mapLocation,
        "description", description,
        "type", geneType,
        "nomenclatureAuthoritySymbol", nomenclatureAuthSymbol,
        "nomenclatureAuthorityFullName", nomenclatureAuthFName,
        "nomenclatureStatus", nomenclatureStatus,
        "otherDesignations", otherDesignations
        //"modificationDate", modificationDate
    );
}
```

5 miRBase

5.1 Introduction

The miRBase database [2] is a searchable database of published miRNA sequences and annotation. Each entry in the miRBase Sequence database represents a predicted hairpin portion of a miRNA transcript (termed *mir* in the database), with information on the location and sequence of the mature miRNA sequence (termed *miR*). Both hairpin and mature sequences are available for searching and browsing, and entries can also be retrieved by name, keyword, references and annotation.

5.2 Data source

The file at the URL ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.dat.gz contains all the current published miRNA data in EMBL format [3]. The newer available database is the version 21 and it was released on June 2014.

5.2.1 The EMBL format

The entries in a database in EMBL format are structured so as to be usable by human readers as well as by computer programs. The explanations, descriptions, classifications and other comments are in ordinary English, and the symbols and formatting employed for the base sequences themselves have been chosen for readability. Wherever possible, symbols familiar to molecular biologists have been used. At the same time, the structure is systematic enough to allow computer programs easily to read, identify, and manipulate the various types of data included.

Each entry in the database is composed of lines. Different types of lines, each with its own format, are used to record the various types of data which make up the entry. In general, fixed format items have been kept to a minimum, and a more syntax-oriented structure adopted for the lines.

Each line begins with a two-character line code, which indicates the type of information contained in the line. The currently used line types, along with their respective line codes, are listed below:

```
ID - identification             (begins each entry; 1 per entry)
AC - accession number           (>=1 per entry)
```

```
PR - project identifier         (0 or 1 per entry)
DT - date                       (2 per entry)
DE - description                (>=1 per entry)
KW - keyword                    (>=1 per entry)
OS - organism species           (>=1 per entry)
OC - organism classification    (>=1 per entry)
OG - organelle                  (0 or 1 per entry)
RN - reference number           (>=1 per entry)
RC - reference comment          (>=0 per entry)
RP - reference positions        (>=1 per entry)
RX - reference cross-reference  (>=0 per entry)
RG - reference group            (>=0 per entry)
RA - reference author(s)        (>=0 per entry)
RT - reference title            (>=1 per entry)
RL - reference location         (>=1 per entry)
DR - database cross-reference   (>=0 per entry)
CC - comments or notes          (>=0 per entry)
AH - assembly header            (0 or 1 per entry)
AS - assembly information       (0 or >=1 per entry)
FH - feature table header       (2 per entry)
FT - feature table data         (>=2 per entry)
XX - spacer line                (many per entry)
SQ - sequence header            (1 per entry)
CO - contig/construct line      (0 or >=1 per entry)
bb - (blanks) sequence data     (>=1 per entry)
// - termination line           (ends each entry; 1 per entry)
```

5.2.2 The ID Line

The ID (IDentification) line is always the first line of an entry. The format of the ID line is:

```
ID   <1>; SV <2>; <3>; <4>; <5>; <6>; <7> BP.
```

The tokens represent:

1. Primary accession number
2. Sequence version number
3. Topology: 'circular' or 'linear'
4. Molecule type
5. Data class
6. Taxonomic division
7. Sequence length

5.2.3 The AC Line

The AC (ACcession number) line lists the accession numbers associated with the entry. Each accession number, or range of accession numbers, is terminated by a semicolon. Where necessary, more than one AC line is used. Consecutive secondary accession numbers in ENA flatfiles are shown in the form of inclusive accession number ranges. Accession numbers are the primary means of identifying sequences providing a stable way of identifying entries from release to release. An accession number, however, always remains in the accession number list of the latest version of the entry in which it first appeared. Accession numbers allow unambiguous citation of database entries. Researchers who wish to cite entries in their publications should always cite the first accession number in the list (the "primary" accession number) to ensure that readers can find the relevant data in a subsequent release. Readers wishing to find the data thus cited must look at all the accession numbers in each entry's list.

5.2.4 The PR Line

The PR (PRoject) line shows the International Nucleotide Sequence Database Collaboration (INSDC) Project Identifier that has been assigned to the entry. Full details of INSDC Project are available at http://www.ebi.ac.uk/ena/about/page.php?page=project_guidelines

5.2.5 The DT Line

The DT (DaTe) line shows when an entry first appeared in the database and when it was last updated. Each entry contains two DT lines, formatted as follows:

```
DT   DD-MON-YYYY (Rel. #, Created)
DT   DD-MON-YYYY (Rel. #, Last updated, Version #)
```

The date supplied on each DT line indicates when the entry was created or last updated; that will usually also be the date when the new or modified entry became publicly visible via the EBI network servers. The release number indicates the first quarterly release made after the entry was created or last updated. The version number appears only on the "Last updated" DT line.

5.2.6 The DE Line

The DE (Description) lines contain general descriptive information about the sequence stored. This may include the designations of genes for which the sequence codes, the region of the genome from which it is derived, or other information which helps to identify the sequence. The format for a DE line is:

```
DE   description
```

The description is given in ordinary English and is free-format. Often, more than one DE line is required; when this is the case, the text is divided only between words. The first DE line generally contains a brief description, which can stand alone for cataloguing purposes.

5.2.7 The KW Line

The KW (KeyWord) lines provide information which can be used to generate cross-reference indexes of the sequence entries based on functional, structural, or other categories deemed important. The format for a KW line is:

```
KW   keyword[; keyword ...].
```

More than one keyword may be listed on each KW line; the keywords are separated by semicolons, and the last keyword is followed by a full stop. Keywords may consist of more than one word, and they may contain embedded blanks and stops. A keyword is never split between lines.

The keywords are ordered alphabetically; the ordering implies no hierarchy of importance or function. If an entry has no keywords assigned to it, it will contain a single KW line like this:

```
KW   .
```

5.2.8 The OS Line

The OS (Organism Species) line specifies the preferred scientific name of the organism which was the source of the stored sequence. In most cases this is done by giving the Latin genus and species designations, followed (in parentheses) by the preferred common name in English where known. The format is:

```
OS   Genus species (name)
```

5.2.9 The OC Line

The OC (Organism Classification) lines contain the taxonomic classification of the source organism. The classification is listed top-down as nodes in a taxonomic tree in which the most general grouping is given first. The classification may be distributed over several OC lines, but nodes are not split or hyphenated between lines. The individual items are separated by semicolons and the list is terminated by a full stop. The format for the OC line is:

```
OC   Node[; Node...].
```

5.2.10 The OG Line

The OG (OrGanelle) linetype indicates the sub-cellular location of non-nuclear sequences. It is only present in entries containing non-nuclear sequences and appears after the last OC line in such entries. The OG line contains one data item (title cased) from the controlled list of the organelle qualifier definition or a plasmid name.

5.2.11 The Reference (RN, RC, RP, RX, RG, RA, RT, RL) Lines

These lines comprise the literature citations within the database. The citations provide access to the papers from which the data has been abstracted. The reference lines for a given citation occur in a block, and are always in the order RN, RC, RP, RX, RG, RA, RT, RL. Within each such reference block the RN line occurs once, the RC, RP and RX lines occur zero or more times, and the RA, RT, RL lines each occur one or more times. If several references are given, there will be a reference block for each.

The RN Line

The RN (Reference Number) line gives a unique number to each reference citation within an entry. This number is used to designate the reference in comments and in the feature table. The format of the RN line is:

```
RN   [n]
```

The RC Line

The RC (Reference Comment) linetype is an optional linetype which appears if the reference has a comment. The comment is in English and as many RC lines as are required to display the comment will appear. They are formatted thus:

```
RC   comment
```

The RP Line

The RP (Reference Position) linetype is an optional linetype which appears if one or more contiguous base spans of the presented sequence can be attributed to the reference in question. As many RP lines as are required to display the base span(s) will appear. The base span(s) indicate which part(s) of the sequence are covered by the reference. Note that the numbering scheme is for the sequence as presented in the database entry (i.e. from 5' to 3' starting at 1), not the scheme used by the authors in the reference should the two differ. The RP line is formatted thus:

```
RP   i-j[, k-l...]
```

The RX Line

The RX (reference cross-reference) linetype is an optional linetype which contains a cross-reference to an external citation or abstract resource. For example, if a journal citation exists in the PUBMED database, there will be an RX line pointing to the relevant PUBMED identifier. The format of the RX line is as follows:

```
RX  resource_identifier; identifier.
```

The first item on the RX line, the resource identifier, is the abbreviated name of the data collection to which reference is made. The current set of cross-referenced resources is:

```
Resource ID    Fullname
-----------    ------------------------------------
PUBMED         PUBMED bibliographic database (NLM)
DOI            Digital Object Identifier (International DOI Foundation)
AGRICOLA      US National Agriculture Library (NAL) of the US Department
               of Agriculture (USDA)
```

The second item on the RX line, the identifier, is a pointer to the entry in the external resource to which reference is being made. The data item used as the primary identifier depends on the resource being referenced.

The RG Line

The RG (Reference Group) lines list the working groups/consortia that produced the record. RG line is mainly used in submission reference blocks, but could also be used in paper reference if the working group is cited as an author in the paper.

The RA Line

The RA (Reference Author) lines list the authors of the paper (or other work) cited. All of the authors are included, and are listed in the order given in the paper. The names are listed surname first followed by a blank followed by initial(s) with stops. Occasionally the initials may not be known, in which case the surname alone will be listed. The author names are separated by commas and terminated by a semicolon; they are not split between lines. As many RA lines as necessary are included for each reference.

The RT Line

The RT (Reference Title) lines give the title of the paper (or other work) as exactly as is possible given the limitations of computer character sets. Note that the form used is that which would be used in a citation rather than that displayed at the top of the published paper. The title is enclosed in double quotes, and may be continued over several lines as necessary. The title lines are terminated by a semicolon.

The RL Line

The RL (Reference Location) line contains the conventional citation information for the reference. In general, the RL lines alone are sufficient to find the paper in question. They include the journal, volume number, page range and year for each paper. Journal names are abbreviated according to existing ISO standards (International Standard Serial Number). The format for the location lines is:

```
RL   journal vol:pp-pp(year).
```

5.2.12 The DR Line

The DR (Database Cross-reference) line cross-references other databases which contain information related to the entry in which the DR line appears. The format of the DR line is as follows:

```
DR   database_identifier; primary_identifier; secondary_identifier.
```

The first item on the DR line, the database identifier, is the abbreviated name of the data collection to which reference is made. The second item on the DR line, the primary identifier, is a pointer to the entry in the external database to which reference is being made. The third item on the DR line is the secondary identifier, if available, from the referenced database.

5.2.13 The AH Line (in TPA and TSA records only)

Third Party Annotation (TPA) and Transcriptome Shotgun Assembly (TSA) records may include information on the composition of their sequences to show which spans originated from which contributing primary sequences. The AH (Assembly Header) line provides column headings for the assembly information. The lines contain no data and may be ignored by computer programs. The AH line format is:

```
      AH    LOCAL_SPAN      PRIMARY_IDENTIFIER      PRIMARY_SPAN      COMP
```

5.2.14 The AS Line (in TPA and TSA records)

The AS (ASsembly Information) lines provide information on the composition of a TPA or TSA sequence. These lines include information on local sequence spans (those spans seen in the sequence of the entry showing the AS lines) plus identifiers and base spans of contributing primary sequences (for ENA primary entries only):

```
a) LOCAL_SPAN                base span on local sequence shown in entry
b) PRIMARY_IDENTIFIER        acc.version of contributing ENA sequence(s)
                             or trace identifier for ENA read(s)
c) PRIMARY_SPAN              base span on contributing ENA primary
                             sequence or not_available for ENA read(s)
d) COMP                     'c' is used to indicate that contributing sequence
                             originates from complementary strand in primary
                             entry
```

5.2.15 The CO Line (in CON records only)

Con(structed) sequences in the CON data classes represent complete chromosomes, genomes and other long sequences constructed from segment entries. CON data class entries do not contain sequence data per se, but rather the assembly information on all accession.versions and sequence locations relevant to building the constructed sequence. The assembly information is represented in the CO lines.

5.2.16 The FH Line

The FH (Feature Header) lines are present only to improve readability of an entry when it is printed or displayed on a terminal screen. The lines contain no data and may be ignored by computer programs. The format of these lines is always the same:

```
FH   Key             Location/Qualifiers
FH
```

The first line provides column headings for the feature table, and the second line serves as a spacer. If an entry contains no feature table (i.e. no FT lines), the FH lines will not appear.

5.2.17 The FT Line

The FT (Feature Table) lines provide a mechanism for the annotation of the sequence data. Regions or sites in the sequence which are of interest are listed in the table. In general, the features in the feature table represent signals or other characteristics reported in the cited references.

5.2.18 The SQ Line

The SQ (SeQuence header) line marks the beginning of the sequence data and gives a summary of its content. Bases other than A, C, G and T are grouped together as "other". The word "Sequence" is present solely as a marker for readability.

5.2.19 The Sequence Data Line

The sequence data line has a line code consisting of two blanks. The sequence is written 60 bases per line, in groups of 10 bases separated by a blank character, beginning at position 6 of the line. The direction listed is always 5' to 3', and wherever possible the non-coding strand (homologous to the message) has been stored. Columns 73-80 of each sequence line contain base numbers for easier reading and quick location of regions of interest. The numbers are right justified and indicate the number of the last base on each line.

5.2.20 The CC Line

CC lines are free text comments about the entry, and may be used to convey any sort of information thought to be useful that is unsuitable for inclusion in other line types.

5.2.21 The XX Line

The XX (spacer) line contains no data or comments. Its purpose is to make an entry easier to read on a page or terminal screen by setting off the various types of information in appropriate groupings. XX is used instead of blank lines to avoid confusion with the sequence data lines. The XX lines can always be ignored by computer programs.

5.2.22 The // Line

The // (terminator) line also contains no data or comments. It designates the end of an entry.

5.3 ETL

The processing of biological data in EMBL format has been developed using BioJava [4], an open-source framework that enables rapid bioinformatics application development in the Java programming language. BioJava contains powerful analysis and statistical routines, tools for parsing common file formats and packages for manipulating sequences and 3D structures.

The ETL's structure is similar to the one used to read textual tab-delimited files, but now the main loop iterates through the sequences found in the source file.

For each miRNA, the ETL extracts all the data and, by the reading of the features, it also extracts all the related mature miRNAs. In this way, it gives values to two class of vertices, *miRNA* and *miRNAmature*, and creates edges of type *precursorOf* between them.

```
…

BufferedReader br = new BufferedReader(new FileReader(fileName));
Namespace ns = RichObjectFactory.getDefaultNamespace();
RichSequenceIterator seqs = RichSequence.IOTools.readEMBLRNA(br,
ns);

while (seqs.hasNext()) {
       RichSequence entry = seqs.nextRichSequence();

       String accession = entry.getAccession();
       String name = entry.getName();
       String description = entry.getDescription();
       Vector<String> dbReferences = new Vector<String>();
       Vector<String> comments = new Vector<String>();

       for (Comment comment : entry.getComments()) {
              String cmt = comment.getComment().replaceAll("\n", "
");
            comments.add(cmt);
        }
        String comment = "";
        if (comments.size() > 0)
            comment = comments.get(0);

        for (RankedCrossRef docRef : entry.getRankedCrossRefs()) {
            String reference = docRef.getCrossRef().getDbname() +
" " + docRef.getCrossRef().getAccession();
            dbReferences.add(reference);
        }

        String sequence = entry.getInternalSymbolList().seqString();

        Vertex mirna = graph.addVertex("class:miRNA",
                    "accession", accession,
                    "name", name,
                    "description", description,
                    "comment", comment,
                    //"dbRefs", dbReferences,
                    "sequence", sequence
                    );

        Iterator<Feature> itf = entry.getFeatureSet().iterator();

        while (itf.hasNext()) {
            Feature f = itf.next();

            String location = f.getLocation().toString();
            String subSequence =
sequence.substring(f.getLocation().getMin()-1,
f.getLocation().getMax());

            Vertex mature = graph.addVertex("class:miRNAmature",
                                "location", location,
                                "sequence", subSequence
                                );

            Map<Object, ?> map = f.getAnnotation().asMap();
            Set<Object> keys = map.keySet();
            for (Object key : keys) {
                String keyString = key.toString();
                String value = (String) map.get(key);
                mature.setProperty(keyString.substring(
                        keyString.lastIndexOf(":")+1), value);
            }

            mirna.addEdge("precursorOf", mature);
        }

}

…
```

6 mirCancer

6.1 Introduction

miRCancer [3] provides comprehensive collection of microRNA (miRNA) expression profiles in various human cancers which are automatically extracted from published literatures in PubMed. It utilizes text mining techniques for information collection. Manual revision is applied after auto-extraction to provide 100% precision. User can search the database by miRNA and/or cancer names in the miRCancer Search page. The website also provides two sequence analysis tools: clustering and chi-square analysis which can perform analysis on all or selected pool of miRNA sequences.

6.2 Data source

The latest miRCancer database is usually available for download upon request. At time of writing, the latest was updated on Dec. 3rd, 2015. In this work, the version http://mircancer.ecu.edu/downloads/miRCancerSeptember2015.txt was used. It is a textual tab-delimited file, and the columns meaning is reported below:

mirId	the unique identifier for a miRNA
Cancer	the cancer name
Profile	the profile: up or down
PubMed Article	the title of the PubMed article

6.3 ETL

The ETL is quite simple, because the source file has a well-defined structure and the number of fields to read is very low.

For each new disease read from the file, a vertex of class *cancer* is created and then it is linked to the related miRNA by a new edge of type *cancer2mirna*.

```
…

while ((line = reader.readLine()) != null) {
    String datavalue[] = line.split("\t");

    String mirId = datavalue[0];
    String cancerName = datavalue[1];
```

```
     String cancerProfile = datavalue[2];
     //String pubmedTitle = datavalue[3];

     Vertex miRNA = null;
     Vertex cancer = null;

     Iterator<Vertex> it = graph.getVertices("miRNA.name",
mirId).iterator();
    if (it.hasNext()) {
        miRNA = it.next();

        it = graph.getVertices("cancer.name",
cancerName).iterator();
        if (it.hasNext()) {
              cancer = it.next();
              String profile = cancer.getProperty("profile");

              if (!cancerProfile.equals(profile)) {
                     if (it.hasNext())
                            cancer = it.next();
                     else {
                            cancer = graph.addVertex("class:cancer",
"name", cancerName, "profile", cancerProfile);
                     }
              }
        }
        else {
              cancer = graph.addVertex("class:cancer", "name",
cancerName, "profile", cancerProfile);
        }

        cancer.addEdge("cancer2mirna", miRNA);

    }
}

…
```

7 microRNA

7.1 Introduction

microRNA [4] is one of publicly available miRNA-target interactions database and contains both validated and predicted interactions. It reports about 2,000 human genes with miRNA target sites conserved in mammals and about 250 human genes conserved as targets between mammals and fish. The prediction algorithm optimizes sequence complementarity using position-specific rules and relies on strict requirements of interspecies conservation.

7.2 Data source

The latest database was released on August 2010 and, for the human species, is splitted in the following files:

http://cbio.mskcc.org/microrna_data/human_predictions_S_C_aug2010.txt.gz

http://cbio.mskcc.org/microrna_data/human_predictions_S_0_aug2010.txt.gz

http://cbio.mskcc.org/microrna_data/human_predictions_0_C_aug2010.txt.gz

http://cbio.mskcc.org/microrna_data/human_predictions_0_0_aug2010.txt.gz

Each of them contain predictions characterized by homogeneous mirSVR score and homogeneous miRNA type (conserved/non-conserved). Therefore, the meaning of *S_C*, *S_0*, *0_C*, and *0_0* in the filename is as follow:

S_C	good mirSVR score, conserved miRNA
S_0	good mirSVR score, non-conserved miRNA
0_C	non-good mirSVR score, conserved miRNA
0_0	non-good mirSVR score, non-conserved miRNA

7.3 ETL

The database files cited above are compressed tab-delimited textual files. Their parsing is quite easy, even if the lines contain a very high number of column.

Each line represents a prediction, that is an interaction between a gene and a miRNA mature. The ETL creates a new vertex of class interaction for each prediction; then that vertex is linked to the related gene and to the related miRNA.

```
…

while ((line = reader.readLine()) != null) {
       String datavalue[] = line.split("\t");

       String mirAccession = datavalue[0];
       //String mirName = datavalue[1];
       String geneId = datavalue[2];
       //String geneSymbol = datavalue[3];
       String transcriptId = datavalue[4];
       String extTranscriptId = datavalue[5];
       String mirAlignment = datavalue[6];
       String alignment = datavalue[7];
       String geneAlignment = datavalue[8];
       int mirStart = Integer.valueOf(datavalue[9]);
       int mirEnd = Integer.valueOf(datavalue[10]);
       int geneStart = Integer.valueOf(datavalue[11]);
       int geneEnd = Integer.valueOf(datavalue[12]);
       String genomeCoordinates = datavalue[13];
       double conservation = Double.valueOf(datavalue[14]);
       int alignScore = Integer.valueOf(datavalue[15]);
       int seedCat = Integer.valueOf(datavalue[16]);
       double energy = Double.valueOf(datavalue[17]);
       double mirSvrScore = Double.valueOf(datavalue[18]);

       Vertex miRNA = null;
       Vertex gene = null;

       Iterator<Vertex> it =
                     graph.getVertices("miRNAmature.accession",
                     mirAccession).iterator();
       if (it.hasNext()) {
              miRNA = it.next();

              it = graph.getVertices("gene.geneId",
                     geneId).iterator();
              if (it.hasNext())
                     gene = it.next();
       }

       if ((miRNA != null) && (gene != null)) {

              Vertex interaction =
                     graph.addVertex("class:interaction",
                            "transcriptId", transcriptId,
                            "extTranscriptId", extTranscriptId,
                            "mirAlignment", mirAlignment,
                            "alignment", alignment,
                            "geneAlignment", geneAlignment,
                            "mirStart", mirStart,
                            "mirEnd", mirEnd,
                            "geneStart", geneStart,
                            "geneEnd", geneEnd,
                            "genomeCoordinates", genomeCoordinates,
                            "conservation", conservation,
                            "alignScore", alignScore,
                            "seedCat", seedCat,
                            "energy", energy,
                            "mirSvrScore", mirSvrScore
                            );
```

```
                interaction.addEdge("interactingMiRNA", miRNA);
                interaction.addEdge("interactingGene", gene);

        }
}

…
```

8 References

[1] Orient Technologies LTD, "OrientDB Distributed Graph Database," [Online]. Available: http://orientdb.com. [Accessed 16 12 2015].

[2] G. D. Schuler, J. A. Epstein, H. Ohkawa and J. A. Kans, "Entrez: molecular biology database and retrieval system," in *Methods in enzymology*, vol. 266, 1996, pp. 141-162.

[3] A. Kozomara and S. Griffiths-Jones, "miRBase: integrating microRNA annotation and deep-sequencing data," in *Nucleic acids research*, vol. 39, 2011, pp. 152-157.

[4] European Bioinformatics Institute, "EMBL Outstation," [Online]. Available: ftp://ftp.ebi.ac.uk/pub/databases/embl/doc/usrman.txt. [Accessed 15 12 2015].

[5] R. C. G. Holland, T. A. Down, M. Pocock, A. Prlić, D. Huen, K. James, S. Foisy, A. Dräger, A. Yates, M. Heuer and M. J. Schreiber, "BioJava: an open-source framework for bioinformatics," *Bioinformatics,* vol. 24, no. 18, pp. 2096-2097, 15 Sep 2008.

[6] B. Xie, Q. Ding, H. Han and D. Wu, "miRCancer: a microRNA-cancer association database constructed by text mining on literature," *Bioinformatics,* vol. 29, no. 5, pp. 638-644, 2013.

[7] B. John, A. J. Enright, A. Aravin, T. Tuschi, C. Sander and D. S. Marks, "Human microRNA targets," *PLoS Biology,* vol. 2, no. 11, 2004.