TIDB2 - Implementation and Testing Object Extensions to Open Source RDBMS for Meteorological Data

João Simões[1], António Amorim[2] and Maria Monteiro[1]

[1] IM-Instituto de Meteorologia, Rua do Aeroporto, Lisboa

[2] SIM/IDL and Fac. Ciências da Universidade de Lisboa, Ed. C8, Campo Grande

Abstract

The present R&D effort to addresses a wide need for indexing scientific object data on time or other varibles using a relational database. While keeping an unified abstract interface to different RDBMS we have introduced a plugin arquitecture that not only loads MYSQL, POSTGRESQL and other at runtime but also extends the content of relational table entries to different scientific objects including Word Meteorological Organization data (like GRIB and BUFR).

The interface is aware of the object schema in orther to create the relational indexes automatically. In particular if a BUFR of a GRIB object is being stored in a relational table containing a column that matches one of its internal parameter names (and header data) not only the entire object is stored but also the specific value is made available for relational queries model.

With the appropriate plugin this system is also suitable for server replication and scalability.

1 Introduction and Motivation

Most of the existing free and commercial database APIs (Application Programming Interfaces)were developed originally to store data that is produced by the end user, like facturation, library or stocks databases and were later extended to support more demanding applications including multimedia data. In the case of scientific data is very common to have another source of data rather than people, often data comes from sensors or computer models that produce large quantities of data or complex datatypes (scientific objects) – this was one requirement of High Energy Physics where TIDB2 was initially developed, using the know-how of the ConditionsDB [1] ancestor project.

A Numerical Weather Prediction (NWP) sustainable work at a small meteorological service like Portuguese Instituto de Meteorologia – IM requires a flexible, but still efficient, data processing and management systems in order to allow the optimal use of the atmospheric available observations in quasi-real time as well as numerical prediction products. In this context, national data processing and management is only a part of the task since useful atmospheric data overtakes political boundaries and hence international exchange of data is a requirement for the NWP work.

At present, World Meteorological Organisation (WMO) is coordinating an effort to replace worldwide exchange data formats for observations from TAC (Traditional Alfanumeric Codes) to TDCF (Table Driven Code Forms) [2]. In particular BUFR (Binary Universal Form for the Representation of meteorological data), created already almost two decades ago in order to allow an efficient way of disseminating and archiving observations [3], is now being used on the circulation of observational data within the Global Telecommunications System (GTS) when universal BUFR decoders exist and are free to be used among the Meteorological community.

At IM, like in many other data processing and forecasting centres at national level (from the WMO Global Data Processing and Forecasting System), the option to organise quasi-real time observational data and numerical prediction products under temporary on-line archives was made already in 1993. Doing this, data integrity could be assured but still be accessed efficiently by different users at the same time to support different weather watch applications. FM 92 – GRIB[4]was the data form of field numerical prediction products and FM94 – BUFR [4] was the form for observational data. A suitable form for the GRIB database management system was then used – a "hierarchical" or tree-based file systems model [5] describing the structure of the information on a database where atomic GRIB objects were considered; moreover an indexed database management system was used following ECMWF schema for on-line database observations where a single file was used for all observations of the same type within 6 hours. To retrieve GRIB and BUFR data at the same time, just a single home made FORTRAN application, here called GB – Grib Bufr – could be used in run time mode. And this data management system was installed under a VAX/OpenVMS computer platform.

Nowadays, the VAX machine does not allow the upgrade of BUFR universal decoders by obvious reasons and full Portuguese system had to be considered under a UNIX or Linux computer platform. Besides, new era of computer and software facilities brought new requirements that should be had to this need, in particular, a visualization interface to easy allow the monitoring of the data arrival and archiving, a more flexible C interface to allow the maintenance of the home made GBFORTRAN applications in order to easily allow the migration of an all

world of weather watch applications running in operational mode since the last 10 years, a more flexible C++ application to allow the eventual creation of new facilities. Finally, FM 94 BUFR and FM 92 GRIB should be considered the atomic objects of such database. However, taking into account the recent acquisition of a discs based storage system at IM, tests should start to check the flexibility of this database to build a NWP historical archive of both BRUF and GRIB data and its versatility to allow queries of long series of data, using the facilities of a relational database.

TIDB2 was designed to satisfy all these needs, using a plugin architecture where the user can specify how TIDB2 will handle his own objects(for automatic "index" creation) and which RDBMS will be used to fit his needs of security and performance.

TIDB2 is a fully featured database interface with a simple intuitive C++ (shortly will available also as C and Fortran)interface that provides a small set of tools that can be integrated with "command tools" to store and retrieve data form databases.

A special effort was placed in keeping the API as simple and intuitive as possible, it is open source published under GPL license allowing everyone to improve and correct it.

2 General structure of TIDB2

The general structure of TIDB2 consists on a set of four main classes:

- **TIDB Connection**: the first class to be used: connects, drops and creates databases and provides methods that return tables and queries. This class also provides access to the plugin virtual class **TIDB Connection PLG** that is implemented as a plugin for each particular RDBMS or file based solution.
- **TIDB Table**: a set of rows, with methods to add and retrieve rows.
- **TIDBRow**: a set of fields, with methods to add and retrieve fields.
- **TIDBField**: a class that provides direct access do the data that is either be of a standard type or of an extended data type, the latter is handled by the virtual class **TIDB TypeEx** and implemented as a plugin.

The TIDB2 plugins are shared libraries loaded at runtime. They are used for database connections and for extended datatypes providing independence from RDBMS or any libraries and APIs used by the extended type handeler. The TIDB2 applications need only to be linked with the general **libtidb2** library. Connection objects are created with a *connection string* with the format:

```
protocol://server:database:username:password
```

For a given protocol the connection will load the appropriate plugin to load the required RDBMS server, calling plugin's virtual class. Till now there are implemented 5 connecion plugins to support MySQL, PostgreSQL, GRIBs and BUFRs on a filesystem and the source file plugin, that will be described later. A connection plugin to Oracle will be available shortly.

The *source file* plugin does not connect directly to any RDBMS but only calls several other connection plugins adding replication and scalability features. This plugin behaves somehow like the Debian's "apt-get" framework. Each client using TIDB2 keeps a list of available servers for each time period. Since we can have different servers for the same database over different time periods this mechanism provides scalability. Also, since we can have different source file lists on different clients pointing to different servers replication is also available.

One of the most important objects in TIDB2 is the transient table used both for retrieving result sets and for storing data in the databases. These tables can be built from scratch using a schema model row, containing the table field names and field types including the extended ones. The table is then filled with data rows and stored by the connection object. Another way of using the **TIDBTable** is to retrieve it from the database, append it with rows and using the store method. Any SQL query will be returned also as a **TIDBTable** however these tables may not in general be suitable for storing. For commodity any relational table inside the database server can be registered into TIDB2 and opened as a **TIDBTable**.

The TIDB2 data elements are managed as **TIDBFields**. They are composed by a name, a type and a value; they can contain any basic data type from boolean to string but also they can contain the abstract class **TIDBTypeEx** for extended types used with scientific objects.

3 Special Streamers and Types

Presently the available extended data plugins include pure BLOBs, arrays of basic types, ROOT objects [6] and World Meteorological Organization's BUFR and GRIB data.

One of the main features of TIDB2 is the special C++ streamer "<<" that fills TIDBRows and processes the scentific objects as needed. This operator automatically casts the data to the appropriate type. For example if we have the row object *MyRow* that contains three columns of the integer type the following code will behave correctly:

```
TIDBRow MyRow(table) << 1 << "2" << 3.0;
```

This row could be streamed into a **TIDBTable** and refilled again with new data.

Before introducing the way how complex data is handled by TIDB2, it is useful to explain what are the three general alternatives for extended data storage in databases:

- **Atomized** – complex data is splitted into all it's elements and stored using basic data types. This method is good for browsing data, however since we'll have a lot of data redundancy and many associations, a lot of disk space will be used and queries will become very unefficient.
- **BLOB storage** – complex data stored only as BLOBs, which is very unsuitable for seeking objects and makes it impossible to quickly find the most relevant data properties.

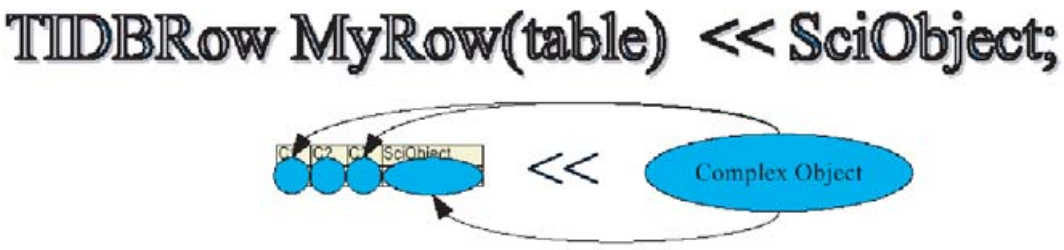

Fig. 1 How the special streamer works

- **Mixed mode TIDB2 approach** Both Tunable metadata and data are stored together in a configurable way and some relevant values are copied form the blob to help "index creation" and "quick searches".

Upon the **TIDBRow** creation the user specifies column names that matches names inside the complex object, while using the special C++ streamer "<<" the internal structure of complex objects is analised and the **TIDBRow** is automatically filled with the scientific object data. These columns can be used as "keys" and are tunable at the moment of the row creation.

4 The WMO BUFR and GIRB Support

Any GRIB/BUFR data inside a corresponding file could be loaded into a **TIDBField** – it just needs to be assigned to a filename, or used with the set method depending if the data is already in memory. After that **TIDBField** provides the necessary functionality to access all GRIB/BUFR data and headers. The user must keep in mind that BUFR tables should be stored in:

```
"/usr/local/etc/emos/bufrtables"
```

or it's location should be set on a environment variable.

Any BUFR/GRIB could be stored in a TIDBRow, making use of the "special streamer" that automates the index creation.

There is another way to access BUFR/GRIB data without using TIDB2 C++API, a set of small tools that can be integrated with *Metview* and "shell scipts" have been developed:

- **storegrib, storebufr** – to store data inside a database.
- **showgrib, showbufr** – to retrieve a set of GRIBs/BUFRs from a database (matching a weather centre, a time interval and a "SQL where clause").

This set of tools uses the environment variable **TIDBURL** that provides the required "connection string" to connect the database.

At the moment we are investigating a way to allow storing a link to a GRIB/BUFR that could be on a different system like **MARS** or some other data storage engine instead of storing the objects themselves.

5 KTIDBExplorer Browser

The **KTIDBExplorer** tool is a graphical interface to browse any TIDB2 database and constitutes a way to quickly browse all data in the databases. Accessing the data stored using the World Meteorological Organization formats can thus be achieved using a simple **QT**[8]/**KDE**[7] (one of most popular Linux window managers) interface.

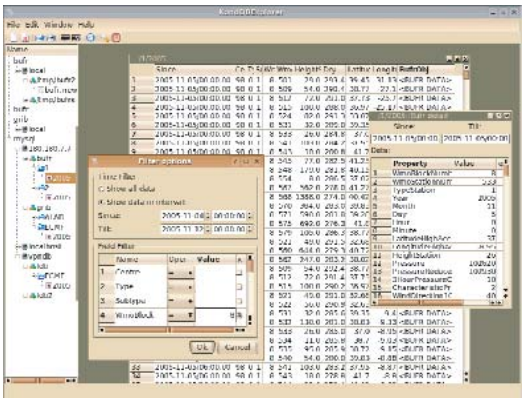

Fig. 2 KTIDBExplorer browsing BUFR data

6 Obtaining TIDB2.

The TIDB2 interface can be obtained by downloading the code from the CVS repository or can be tried live using the PAIPIX [9] live Linux distribution. The TIDB2 source code can be obtained using:

- cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/t-i-db login
- cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/t-i-db co -P tidb2

To download KTIDBExplorer from CVS:

- cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/t-i-db co -P ktidbexplorer

Tarballs can be found at:

https://sourceforge.net/project/showfiles.php?group id=117005

To contact me for help: Email to joao.simoes@meteo.pt

7 Conclusions and outlook

The plugin architecture is a very useful feature, any developer using TIDB2 API doesn't need to be aware of any RDBMS specific API since there is the TIDB2 layer that is common for all systems. Also after a connection plugin is available for a new RDBMS system all existing TIDB2 applications can start using the new plugin, making RDBMS migration a very simple task. This is also an advantage of the extended types, for example if we want to upgrade all applications using GRIB to GRIB2 data formats, we just need to have a new plugin that supports both. Since plugins are just implementations of virtual classes they are fairly easy to develop – just as the body of few functions needs to be coded.

Although the pair TIDB2<->MySQL have performed very well until now, it seems that after storing some hundreds of GigaByte of data into a table the performance may decrease, demanding some research on the "reference to BLOB" storage instead of storing BLOBs directly.

The way TIDB2 solves the problem of index creation saving metadata with the data is not new, however the ability to define at table creation time or at a maintenance task which metadata is stored is a very handy feature for schema migration.

KTIDBExplorer combined with TIDB2 is a very powerful tool, since all that inside a database could be graphically browsed.

The preferred way to get TIDB2 is the CVS, however a very simple way to try it is to run a DVD with **PAIPIX scientific live linux** [9].

References

[1] **Amorim A., N. Barros, D. Klose, J. Lima, C. Oliveira** and **L. Pedro**. Evaluation of the Relational Implementation of the Conditions Database Interface, CERN, Feb 2003

[2] WMO CBS Expert Team on Migration to Table Driven Code Forms Plan for Migration to Table Driven Code Forms, WMO, Switzerland, 2003 (link: http://www.wmo.ch/web/www/WMOCodes/MigrationTDCF/MigrationPlan.doc)

[3] Bynary Universal Form for Data Representation, FM 94 BUFR, Collected papers and specification, Several authors, ECMWF, UK, 1988

[4] Manual on Codes, Volume I.2, International Codes, Part B -Binary Codes, WMO no306, Switzerland (link: http://www.wmo.ch/web/www/WMOCodes/ManualCodes/WMO306vol-I2PartB.pdf), WMO, 2001

[5] **Ullman, J. D**. and **J. Widom**, A first course in database systems, Prentice-Hall International, Inc. USA, 1997

[6] **Simões J., A. Amorim** and **R. Neves.** TIDB, an investigation of the interface between object structures for scientific data and relational database features. ROOT 2005 workshop, CERN – Switzerland, Oct 2005

[7] Kdesktop Environment http://www.kde.org/.

[8] QT,a complete C++ application framework http://www.trolltech.com/.

[9] **Amorim A.** and **L. Amorim**, at this Workshop and references therein, ECMWF – Reading, Nov 2005. Making the ECMWF tools, including Metview, available in a restricted version of the PAIPIX scientific live linux (http://www.papipix.org).