Main Memory Data Processing - Generic data loader

Manuel Blechschmidt

Hasso-Plattner-Institut, 14482 Potsdam, Germany
Manuel.Blechschmidt@student.hpi.uni-potsdam.de

Abstract. This paper will explain which methods exist for the different steps to load arbitrary data into an arbitrary database. Further some of these abstraction frameworks will be compared by speed. This loading process is needed in a lot of situations. In the paper the author will use the example of a transfer from data of an OLTP system to an OLAP system and the example of a benchmark.
In general the process of generic data loading is abbreviated as ETL (Extract-Transform-Load) [3] which describes the different parts of the loading.

1 Introduction

In the last 30 years the world of enterprise computing divided into two major system. There are the Online Transaction Processing systems (OLTP) which are used in the common day-to-day operations in a normal company. They are used for example for creating customers, saving orders, and creating bills. These tasks contain a lot of small write operations and only a few reads. [3] Further the companies deploy big Online Analytics Processing systems (OLAP) to support the management during the strategic planing for the future. These software needs a lot of data and is used to create management reports for examples aggregating sales in a particular region.

Both system play an important role for competing in the market and to make the enterprise processes cheap and effective. The data which is used in the two scenarios is the same but because an OLTP and an OLAP system poses different characteristics in the querying every single system has an own copy of the data but are using different structures to save them to provide best performances for their workload.

Normally the data is entered during the sales, delivery, procurement and support cycle into the OLTP system and then on a regular basis it is extracted, transformed and loaded into the OLAP software. This process is called ETL. In huge systems the ETL process can take a long time and when the data is loaded further time is consumed to pre calculate the desired aggregations. The results are saved in so called cubes.

When using an OLAP system the manager has to know in advance what information he wants to know and then he has to wait until the ETL process and

the cube calculation has been done. Normally this is done in a batch job at night. Because human beings thing in an visual, iterative and evolutionary way [11], it should be possible to make experiments with the results of the OLAP system. At the moment the trend in OLAP systems goes to a new system architecture by using high performance databases which use column-based storage, in-memory processing and compression. [4]

Because new system use sometimes hybrid databases for not copying the data, new requirements arise.

The current problem is that the old benchmarks which were either designed for OLTP or for OLAP systems does not give reasonable results anymore. So a new benchmark, which includes both workloads for an OLTP and for an OLAP systems should, be designed. One part of this benchmark is the loading mechanism for the data. For convenience reasons different characteristics of this process should be measured and should be a small influence on the overall rating.

In this paper the author will present different approaches and frameworks to load data. One of these approaches could be used in the proposed benchmark [1].

Section 1 gave a brief overview about the change of enterprise systems. Section 2 will give a description of the extract and transformation process further it will present common problems and best practices. The architecture part 4 will described a architecture blueprint of a loader and explain some design decisions. In section 5 the performance of the implemented loader and two other implementations with other technologies will be compared. The paper continues in 6 what should be examined next and how the loader can be used in other projects. Then 7 shows other approaches for benchmarks and introduces some other approaches for loading. The paper will end with the conclusion in 8.

2 Extract and Transform

The next section will explain how data is stored in OLTP and other systems and which common methods are used to extract that data. Afterwards it will introduce some standardized and other common methods to transform this data into a loadable form for other applications.

Figure 1 shows the usual actors during an ETL process. The goal is most of the time to transport transactional data to an analytical database which can be afterwards used for reporting purposes. During this process the data has to pass through 3 phases. The different phases will be discussed in detail in the next paragraphs.

The extract process of the data is most of the time easy (Fig. 1 Extractor). The used systems support export formats or interfaces which provide the data in an accessible format. When this is not the case or the provided interfaces are to slow then the data can directly be read from disc in the native format of the application. If this format is well documented then the transformation afterwards is not challenging. Only in special cases like encrypted data for example for credit cards [5] or undocumented proprietary formats the extraction can be hard.

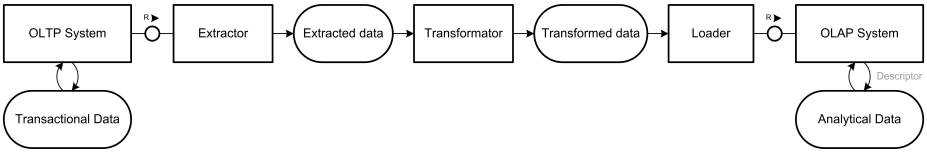

Fig. 1. Actors of usual ETL process.

There are a lot of consortiums and companies which define common exchange formats. The probably most famous at the moment is XML.[6,7] Further there exists CSV (comma separated values) formats, standardized and proprietary binary formats and multimedia codecs. Even when there are standards, sometimes the standards are competing against each other like Open Document Format and Open XML format or the formats are not common like the eCommerce standards xCBL or cXML. Most of the eCommerce systems do not use standardized formats as their native saving format. [10]

When the extraction is done and the data is accessible, it has now to be transformed to the new data schema (Fig. 1 Transformer). For XML exists the XSLT[8] language. At the moment all common XSLT processors[9] need a lot of memory. This is the case because the whole document has to be read into memory to be processed. Alternatives are programming APIs like SAX or StaX.

For CSV formats scripts or programs in any particular language can be used. In the implementation for this paper a PHP script which transformed the CSV data into SQL insert statements was used.

Besides the structural changes it is sometimes also necessary to change the representation. In the last years it was common to switch from legacy character sets like ANSI or ISO-8859-1to the universal Unicode charset. Nearly every transformation program has the ability to map these charsets. In unix environments the iconv program is famous.

2.1 Charset Mapping

The charset mapping is done with mapping tables. These mapping tables are provided by the vendors which created them based on the charset description. Further they have to provide sorting tables which explain how the different characters should be sorted. With a charset set change it could happen that also the sorting changes because different characters sets could probably sort the characters differently. Further there can be problems with case transformations because some characters like the german sz did not exist as uppercase until 2004. [12]

Sometimes new characters are invented like the euro sign €[22] further new versions for unicode are released. All these changes could make adjustments for the charset mapping necessary.

3 Load

The next section will explain common problems for data loading further it will show some common solutions for them. With data loading the process of sending function calls with payloads through the database interfaces.

During the loading process (Fig. 1 Loader) errors for certain rows can happen because of the huge amount of data this is common and will be ignored. The data will be used for reporting purposes and therefore small misses do not matter for the overall result. Common errors are missing escaping characters, characters which do not exist in the encoding or data which do not fit in the data type.

It can take a long time to load analytical data. A often used solution is to disable the ACID (atomic, consistent, isolated and durable) features of a database. Then the database can use a bigger memory buffers and do not have to make integrity checks immediately. This can make the process a lot faster. Most vendors provide special tools which extra command line switches to make that faster for example bcp (Bulk copy) for Sybase Adaptive Server Enterprise or the LOAD utility for DB2.

3.1 Type Mapping

It is necessary for certain operations like sorting, aggregation or arithmetic that semantic information like the type of the data is available. Some operations like compression can be more efficient when more information about the data is available. Some databases have functions which can analyze the data in a certain table and propose the types for the different columns [18]. Besides these auto detect functions a lot of abstraction layers have their own type system [19,21]. The type systems provide mappings to the SQL-92 data types. Some databases extend or change these types. For performance reasons it makes sense to implement these new data types.

In this paper the schema had to be translated from a proprietary format into SQL-92 datatypes. This was done by using generics types like VARCHAR or DOUBLE. This is not the most efficient way of saving the data but the author expects that this has not a big influence.

3.2 Error Mapping

Every database has an own error model. Own error messages and error codes are defined and often a localization for different countries is supplied. For the loading process and an architect of an database abstraction layer this can be quite complicated. The normal way of fixing this is providing an own error model with the abstraction layer and mapping all other error models to it. It makes sense to adapt to a convention like X/Open[23]. This can be a lot of work for example MySQL has about 600 error codes [24]. Besides the error codes Microsoft SQL server also have a severity model.

Most of the time the error models of the abstraction layers are a lot simpler then from the database vendors. For example QtSQL has 5 different error types.

JDBC tries to adapt to the X/Open SQLState conventions [23] but this has to be implemented by the JDBC driver for the specific database.

4 Architecture

This section will present an example architecture how a generic data loader could be designed. In picture 2 the main parts can be seen.

The config file consists the information which are needed to connect to the database, host, user, and password. Further it contains the name of the driver which has to be used. The schemas storage contains SQL DDL or loader specific files which define the table structure for the data. In this paper it was possible to use the same files for two databases. No special features like special table formats to enable transactions or foreign key constraints were used for the loader.

The data contains a bunch of files which contain the actual content for the database. In this paper normal SQL-INSERT statements were used which were extracted with a php script from CSV data which came from an ERP system.

The user interface has only a convenience function and should give the user confidence in the system. It can be used to configure the database and during the loading it can show a progress bar. Some validations check like connections checking or strength of password could be integrated in the user interface.

For this paper only a small form was designed but because of time it was not integrated into the prototypical implementations.

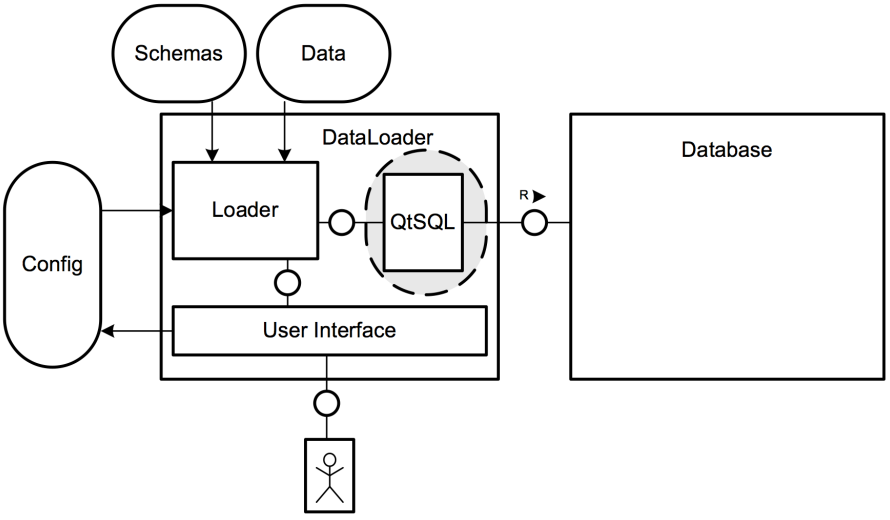

Fig. 2. Overall architecture of data loader.

5 Benchmark

The performance characteristics of the loading process of the data are focused in this paper. Three different loaders based on three different platform stacks have been implemented. All are using the same data schemas and are loading the same data. The tests are run against two database implementations.

In these benchmarks the three stacks are compared with each other based on performance. A bunch of SQL files have to be loaded into two different databases. The main load is created by a file with about 20.000 insert statements. The inserts are executed as they are. No prepared statements are used.

In previous measurements it was measured if the IO of loading the schemas into memory influences the results. It was found out that the impact is insignificant.

The databases implementation differ in their architecture. The first one runs in an own process which is connected via a socket the second one is an in process database which provides directly an API to access the data.

The time was taken with the time utility for unix systems. The benchmark was run on the following hardware

- MacBook Pro
- 2.53 Ghz Intel Core 2 Duo
- 4 GB 1067 MHz DDR3
- Mac OS X Snow Leopard 10.6

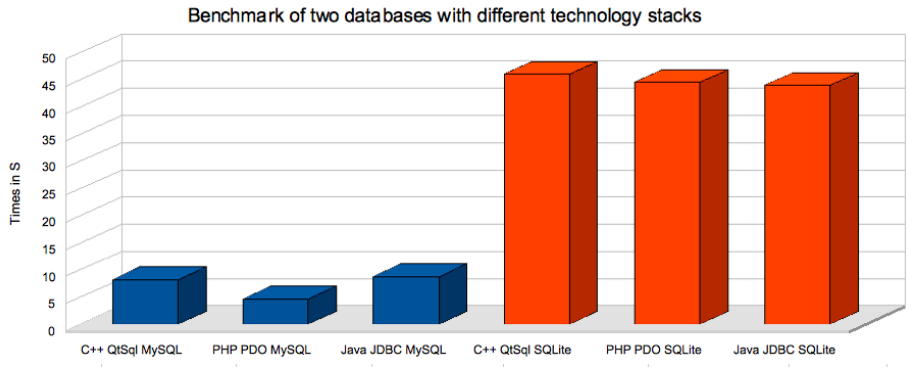

Fig. 3. Benchmark of different stacks.

The following versions of the software were used:

- PHP 5.3.0, mysqlnd 5.0.5, sqlite 3.6.15 (64 bit)

- Java(TM) SE Runtime Environment (build 1.6.0_15-b03-219), Java HotSpot(TM) 64-Bit Server VM (build 14.1-b02-90, mixed mode), MySQL Connector 5.1.8, sqlitejdbc v0.56
- Qt 4.5 (32 bit)

In 3 it can be seen that the speed between the different sqlite implementations does not differ a lot. It is obvious that sqlite is a lot slower like 5-10 times during inserts then mysql depending on the compared technology stack. The reason for this is that for every insert a flush is executed and the insert is written on disc.

It can be seen that for sqlite there is no big difference in the technology stacks but for mysql it can be seen that the PHP PDO MySQL implementation is nearly twice as fast as the QtSQL or the JDBC implementation.

6 Future Work

The loader has to be integrated into the cbtr benchmark suite. In this integration it can fulfill its primary purpose by pre populating a database with the necessary benchmark data. In the implementation introduced in this paper only two databases were tested which are not enough for a generic software component. So more databases have to be tested and an automated testing process for the component itself should be developed.

If a database which should be benchmarked is not supported by the standard QtSql modules, then a new driver has to be implemented [17].

For the performance study is should be investigated why the mysql php driver is so much faster then the JDBC and the QtSQL version further it should be studied how SQLite would perform if the benchmark would be rerun with the sync option disabled. A profiler for Java and C++ should be used to analyze the call stacks and figure out where the most time is consumed.

The benchmark should use more efficient datatypes and check if the datatype mapping has a big influence on performance.

Loading is only one particular task of an OLAP system. It should also be studied how other functions like querying or updating can be used in a generic way and which problems arise. The computer processors contain more and more cores it should be examined how the loading process can be paralyzed and how the data could be distributed to large scaling systems.

7 Related Work

Full applications suites exists which cover the whole business intelligence process [25]. Some of them like Pentaho also offer tools for data loading and integration. The open source Pentaho Data Integration tool is a completely visual tool for data transformation and integration. Also commercial vendors like Altova do research and offer sophisticated tools for data mapping and afterwards loading. The application is called MapForce and also offers a complete visual way to match schemas.

There are already a lot of other benchmarks which test database for specific workloads. The Transaction Processing Performance Council TPC is the best know organization for providing benchmarks for testing database performance. For a database vendor it is important to have a good score because it gives the customer the possibility to compare different products.

TCP provides many benchmarks for OLTP systems. The one which is important for the customer is called TCP-H [13].

The TCP-H benchmark is already pretty old and only focuses on the transactional behavior of the database it does not incorporate big analysis queries. During the benchmark the ACID principles atomic, consistent, isolated, and durable have always to be true.

Further there are a lot of specific benchmarks which test a system including other systems like PolePosition, which tests databases with O/R mappers [14]. There is also a benchmark developed by SAP called SAP Standard Application Benchmark [15]. This benchmark can be used to find the appropriate hardware sizing for an SAP system. Exploring the system behavior and monitoring bottlenecks is also possible.

8 Conclusion

In this paper the basic problems and solutions for an ETL process were mentioned. The basic architecture of a database abstraction layer was discussed based on a C++ example.

Further two other easy ways of loading data with different technology stacks into a relational database system were shown and examined. In the tests it can be seen that the technology stack has only in special circumstances an influence on the performance but the database system and configuration is much more important for the speed.

Even with abstraction layers it is still hard to use a generic approach to load data into a system. A good specification for the abstraction layer is necessary and it should be supported by a big suite of test cases. The implementor of the ETL process needs good knowledge about all systems which are involved. These people can be hard to find. [16] Further the system do change often.

References

1. Anja Bog, Hasso Plattner, Alexander Zeier: **A mixed transaction processing and operational reporting benchmark** (2009)
2. Digital Equipment Corporation: **Information Technology - Database Language SQL** ISO/IEC 907 (July 30, 1992) http://www.contrib.andrew.cmu.edu/ shadow/sql/sql1992.txt
3. Prof. Dr. Peter Gluchowski, Prof. Dr. Hans-Georg Kemper: **Aktuelle Konzepte und Entwicklungstrends - Quo Vadis Business Intelligence?** BI-Spektrum (January, 2006) http://www.sigs.de/publications/bi/2006/01/gluchowski_kemper_BIS_01_06.pdf

4. Surajit Chaudhuri, Umeshwar Dayal : **An overview of data warehousing and OLAP technology** ACM SIGMOD Record archive Volume 26 , Issue 1 (March, 1997) Pages: 65 - 74
5. Payment Card Industry (PCI): **Data Security Standard - Requirements and Security Assessment Procedures** (July, 2009) https://www.pcisecuritystandards.org/security_standards/download.html?id=pci_dss_v1-2.pdf
6. T. Bray, J. Paoli, C.M. Sperberg-McQueen: **Extensible Markup Language (XML) 1.0 W3C Recommendation** (February, 1998) http://www.w3.org/TR/REC-xml.
7. Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, Ian Horrocks: **The Semantic Web: The Roles of XML and RDF** IEEE Internet Computing, vol. 4, no. 5 (September/October, 2000) Pages: 63-74
8. James Clark: **XSL Transformations (XSLT) - Version 1.0** (November 16, 1999) http://www.w3.org/TR/xslt
9. Delpratt O'Neil, Rahman Naila, Raman Rajeev: **Engeneering Succinct DOM** (December 12, 2007)
10. Robert J. Kauffmann, Hamid Mohtadi: **Proprietary and Open Systems Adoption in E-Procurement: A Risk-Augmented Transaction Cost Perspective** Journal of Management Information Systems Volume 21, Number 1 (June, 2004) Pages: 137 - 166
11. Barbara Tversky: **What does drawing reveal about thinking** (1999) Pages 93 -101
12. Andreas Stötzner: **Vorschlag zur Kodierung eines versalen ß in Unicode** (November 10, 2004) http://std.dkuug.dk/jtc1/sc2/wg2/docs/n2888.pdf
13. Transaction Processing Performance Council: **TPC Benchmark H 2.8.0** http://www.tpc.org/tpch/spec/tpch2.8.0.pdf (September 11, 2008)
14. **PolePosition - the open source database benchmark** http://www.polepos.org/
15. SAP AG: **SAP Standard Application Benchmarks** http://www.sap.com/solutions/benchmark/index.epx
16. A. Reinberg, M. Hummel: **Steuert Deutschland langfristig auf einen Fachkräftemangel zu?** IAB Kurzbericht (2003) http://www.abimagazin.de/200406/material/iab_9.pdf
17. Nokia Corporation: **Qt Reference Documentation - SQL Database Drivers - How to Write Your Own Database Driver** http://doc.trolltech.com/4.5/sql-driver.html#development
18. MySQL AB: **21.3.1. PROCEDURE ANALYSE** http://dev.mysql.com/doc/refman/5.0/en/procedure-analyse.html
19. Sun Microsystems Inc.: **Getting Started with the JDBC API - Mapping Overview** (2001) http://java.sun.com/javase/6/docs/technotes/guides/jdbc/getstart/mapping.html#996858
20. Jim Melton, Alan R. Simon: **Understanding the new SQL: a complete guide** (1993) http://books.google.de/books?id=VCsIPDZVQAIC&printsec=frontcover&source=gbs_v2_summary_r&cad=0
21. Lorenzo Alberton, Lukas Smith: **MDB2 - Datatypes – An overview of datatype handling** (Mai 3, 2007) http://pear.php.net/manual/en/package.database.mdb2.datatypes.php
22. Lisa Moore: **The Unicode Standard®, Version 2.1** (September 4, 1998) http://www.unicode.org/unicode/reports/tr8/tr8-2.html

23. X/Open Company: **X/Open Sql and Rda (X/Open Cae Specification)** (February, 1995)
24. MySQL AB: **MySQL 5.1 Reference Manual 21.4.4.7. Mapping MySQL Error Numbers to SQLStates** http://dev.mysql.com/doc/refman/5.1/en/connector-j-reference-error-sqlstates.html
25. Darius Hedgebeth: **Data-driven decision making for the enterprise: an overview of business intelligence applications** VINE (2007) Volume: 37 Issue: 4 Page: 414 - 420