

# **Requirements Specification – NewSQL**

---

Author: Thomas Müller

Version: 0.6

Date: 2003-05-27

---

## Table of Contents

1	Revision History.....	3
2	Summary.....	4
3	Project Scope.....	4
3.1	Priorities.....	4
4	Development Platform.....	5
4.1	Software.....	5
4.2	Development Tools.....	5
4.3	Tested Operating System.....	5
4.4	Hardware.....	6
5	Use Cases.....	6
6	Comparison.....	6
7	Language Features.....	6
7.1	General Requirements.....	6
7.2	DDL (Data Definition Language).....	7
7.3	DQL (Data Query Language).....	7
7.4	DML (Data Manipulation Language).....	8
8	Survey.....	8
9	NewSQL to SQL Translator.....	8
9.1	Functional Requirements.....	8
9.2	Testing Requirements.....	9
10	Benchmark Application.....	9
10.1	Functional Requirements.....	9
10.2	Testing Requirements.....	10
11	Query Wizard.....	10
11.1	Functional Requirements.....	10
11.2	Testing Requirements.....	11
12	Demo Application.....	11
12.1	Functional Requirements.....	11
12.2	Testing Requirements.....	11
13	General Limitations.....	12
14	Publishing Style.....	12
15	Copyright.....	12
16	Popular Database.....	12
17	Project Organization.....	13
17.1	Involved Parties.....	13
17.2	Approach / Processes.....	14
17.3	Schedule.....	15
17.4	Milestones / Reviews.....	15
17.5	Working Hours.....	15
17.6	Meetings with the Tutors.....	16
18	Documentation.....	16
18.1	Language.....	16
18.2	File Format and Location.....	16
18.3	Product Documentation.....	17
18.4	Journal.....	17
18.5	Meeting Protocols.....	17
18.6	Project Report.....	17
19	Results.....	18
20	Rating.....	18
21	Appendix.....	18

---

21.1 Glossary / Links.....	18
----------------------------	----

## **Index of Tables**

Table 1 - Revision History.....	4
Table 2 - Priorities.....	5
Table 3 - Involved Parties.....	14
Table 4 - Milestones / Reviews.....	16
Table 5 - Meetings.....	17
Table 6 - Rating.....	19

## **Drawing Index**

Drawing 1 - Approach.....	15
---------------------------	----

## **Illustration Index**

Illustration 1 - Schedule.....	15
--------------------------------	----

## 1 Revision History

<b>Version</b>	<b>Date</b>	<b>Topic</b>	<b>Remarks</b>	<b>Who</b>
0.1	2003-05-01	Draft		tm
0.2	2003-05-10	Add details		tm
0.3	2003-05-12	Add planning		tm
0.4	2003-05-17	Schedule updated		tm
0.5	2003-05-19	Development Platform	Added requirements batch files, installer, build scripts	tm
0.6	2003-05-27	Development Platform	JDK 1.4 now a must; Ant scripts a must	tm
0.6	2003-05-27	Tested operating system	Windows XP is a must - other a may	tm
0.6	2003-05-27	Hardware	Windows compatible is a must - others a may	tm
0.6	2003-05-27	General Requirements	clairification on extensibility	tm
0.6	2003-05-27	DML	Added 'Upsert' as a may	tm
0.6	2003-05-27	Publishing Style	Corrected link	tm
0.6	2003-05-27	Copyright	Link to Apache license	tm
0.6	2003-05-27	Milestones	Corrected type: January 2004	tm
0.6	2003-05-27	Journal and Meeting Protocol	Combined both paragraphs	tm
0.6	2003-05-27	Rating	Modifications: Creativity now 2 (was 1), Implementation now 3 (was 2), Conformance now 3 (was 5)	tm
0.6	2003-05-27	Glossary	Added new keywords OQL,...	tm

Table 1 - Revision History

## 2 Summary

Currently SQL is the generally accepted, most commonly used database (query-) language. However SQL has some disadvantages:

- There is no real standard (each database has its own SQL dialect).
- There are too many syntax rules.
- Some functionality seem outdated (e.g. length restriction for text; this was probably inherited from COBOL – modern languages like Java do not know such restrictions).
- Null is not equivalent to null - this is not obvious for many developer.
- It is difficult to integrate SQL in Java.
- The SQL language is very old and the language was never revised, but extends in each case.

The target of this project is to define a replacement for SQL. The new language should be easier to learn, more elegant, consistent, and well defined. The new language should not be a subset or extension of SQL, and not an object database language.

The main targets of this project are to create a new database access language called NewSQL, a converter to allow applications written in NewSQL to work with current databases, as well as tools, examples and documentation promoting the usage of this language. The language as well as the tools are to be licensed as open source.

Actually, an open source project called NewSQL has already been started on November 2001, but has been stopped on July 2002 for various reasons. The current project may be a continuation of the old NewSQL open source project.

## 3 Project Scope

### 3.1 Priorities

In order to achieve the goals of the project, there are three priorities defined:

<b>Priority</b>	<b>Definition</b>
Must	This target must be reached.
Should	At least some of these targets should be reached.
May	It is not required to implement any of these requirements. This functionality is documented only to show in what direction further development could go.

Table 2 - Priorities

## 4 Development Platform

### 4.1 Software

The programming language for all software components is Java.

- Must: The applications must run with JDK 1.3.
- Must: It must be possible to compile with JDK 1.3.
- Must: Build and execute scripts must be included in the software. The format of the build scripts must be cross platform.
- Must: A description on how to install and run the software must be provided. An automatic installer is not required.
- Must: It should be possible to compile and run the applications with JDK 1.4. If the source code must be changed in order to compile with JDK 1.4 (for example, because some interfaces are extended in JDK 1.4 that are implemented in the NewSQL software), then the procedure to allow compilation with JDK 1.4 should be simple, and fully documented.
- Should: Platform independence should be achieved by using Apache Ant scripts and documentation how to run the Ant scripts. It is not required to provide batch files or other double-clickable files (for example, .exe files) or an installation.
- May: Existing software such as the Parser generator Antlr or JavaCC may be used. The JDBC driver may be based on existing JDBC drivers such as the open source driver from the project LDBC.
- May: Batch files for some operating system(s) (for example, Windows XP) may be provided. However it is not required to make the batch files cross-platform.
- May: An installer may be provided. This may be an installer for just one operating system (for example, just Windows), or a cross-platform installer.

### 4.2 Development Tools

It is not defined what development tools are used, however there are some restrictions on the usage of proprietary tools.

As this project is open source, other developers should be in the position to seamlessly continue development after the official project period of this project (that is, at the end of the official project period). To allow this, all components must be editable with standard tools, preferably open source tools.

### 4.3 Tested Operating System

Because currently Microsoft Windows is the most common operating system (probably even among open source developers), a current version of Windows (for example Windows XP) is used as the development platform.

- Must: The software must be tested on the Windows XP operating system.

- May: The software may be tested in other operating systems, for example Linux.

## 4.4 Hardware

There are no special requirements for the hardware. However, as Windows XP is the test platform, the hardware where the software works is:

- Must: Microsoft Windows compatible hardware.
- May: Any Linux compatible hardware.

## 5 Use Cases

There are three main use cases for NewSQL. They are very similar to the use cases of SQL:

- Running ad-hoc statements or running existing scripts against a database.
- Embedding NewSQL statements in an application.
- NewSQL statements are generated by an application such as an object–relation mapping library, or an application server.

## 6 Comparison / Documentation

To show the advantages of the new database access language, existing database access techniques must be compared with NewSQL. The main differences must be documented, in order to show the advantages (but also potential disadvantages) of NewSQL. The audience of this comparison is the average developer, in order to give him the list of reasons on why to use NewSQL (maybe combined with other techniques). The documentation can be viewed as a type of marketing material for NewSQL, while still being technically correct, not hiding problems. The documentation should not go to very low level technical details, but should stay simple and readable. The comparison must include:

- Must: SQL (Structured Query Language)
- Must: OQL (Object Query Language)
- Should: APIs: ODBC, JDBC
- May: Embedded SQL, SQLJ
- May: Object Relation mappings, JDO

## 7 Language Features

### 7.1 General Requirements

The new language should have the following features. The following list does not have strict priorities as these items are not functions in a strict sense but represent only a general direction of the solution:

- Simple to learn, if possible simpler than SQL
- Consistent
- Have a low number of grammar rules
- Be extensible in a way that can not break existing applications. For example, adding new functionality to the access right subsystem without adding a new keyword. This is similar to adding libraries to a general programming language.
- Solve all common data access requirements
- Easy to parse
- Modern
- Not have any hidden unexpected (or confusing) behavior (like NULL behavior in SQL)
- It should be possible to write a query wizard, to generate a statement in a step-by-step way. See also the Query Wizard Application. This requires that 'roots' come before the 'nodes' (example: the table name is before the column names; this is not always the case in SQL, for example the SELECT statement where the table name(s) come after the select list)
- Should be functionally compatible to SQL so that the current database APIs can be reused. Reason: it is less likely that developers accept a new language if they have to learn and use a new API as well.

### 7.2 DDL (Data Definition Language)

- Must: creating & dropping tables functionality
- Must: primary key support
- Must: data types integers, fixed point / floating numbers of some kind, strings, date/time/timestamp of some kind and binary must be supported
- Must: null values (where behavior does not need to match exactly the SQL behavior)
- Should: creating and dropping indexes
- Should: automatic surrogate keys (auto increment columns)
- Should: referential integrity
- May: adding / dropping columns from existing tables
- May: creating / dropping users, access rights

- May: stored procedures, functions
- May: upsert (combination of insert and update: insert a row if a row with the same primary key does not exist, otherwise update the existing row)

### **7.3 DQL (Data Query Language)**

- Must: select columns from a table
- Must: select with join
- Must: select with condition (where clause)
- Must: comparison (=, >, <, >=, <=)
- Must: ordering
- Should: grouping
- Should: formulas
- Should: select count, max, min, sum, average
- May: distinct, alias, union, outer joins, having (where clause after grouping)
- May: subqueries (remark: MySQL does not support this feature)
- May: navigation without explicit joins as in OQL. Should make joins unnecessary, for example: select order.client.name from order

### **7.4 DML (Data Manipulation Language)**

- Must: update
- Must: insert
- Must: delete
- Should: transaction support
- Should: built in functions, type casting
- May: updating across multiple tables
- May: insert from a select
- May: additional transaction control, savepoints, 3-phase-commit
- May: triggers, roles, schemas, cross joins, default values
- May: exception handling

## 8 Survey

The survey must take place. The reason for the survey is to find out what are the major problems of developers when working with databases, and an idea on what the developers would like the new language to look like. The survey should give the developers the choice between different grammar variants of the new language, and ask about knowledge on specific topics on database access.

- Must: The results of the survey must be documented
- Must: The survey must take place on the internet
- Should: It should also take place in a controlled environment (for example a class of the school ISBE)
- Approximate date: 2003-06-23; before the summer vacations

## 9 NewSQL to SQL Translator

### 9.1 Functional Requirements

Implementation of a NewSQL to SQL translator.

- Must: Design and implementation of a NewSQL to SQL converter.
- Must: The converter must correctly convert NewSQL statements to SQL to at least two popular databases.
- Should: The converter should be part of a working and more or less (\*) complete JDBC driver, so that it can be used in an application.
- May: Conception of a converter from SQL to NewSQL, or documentation on this topic.
- May: The converter may be based on LDBC or use. The NewSQL to SQL converter may share some of the code with the open source project LDBC.
- May: Additionally, the converter may be made available stand-alone so it can be ported to other languages or APIs.

(\*) The JDBC API is very large, and even the major database vendors do not provide complete implementations of the API. However most applications and tools only use a small subset of the JDBC API.

### 9.2 Testing Requirements

- Must: All main statements (create, drop, select, update, delete,...) must be tested.
- Must: 70% coverage of the hand written source code (in line of source code; if a parser generator is used, there may be some code that is never actually used for this grammar).
- Should: 85% coverage of the source code.
- May: More than 95% code coverage.

- Should: The benchmark application should be tested.
- Should: The demo application should be tested.
- May: A 'random statement generator' may be used to extensively test all variations of the grammar.

## 10 Benchmark Application

### 10.1 Functional Requirements

One of the major concerns for application developers is speed. If the new language is slow, for example because the translation from NewSQL to SQL is slow, or because the generated SQL code executes a lot slower than functionally equivalent SQL code, then the developers will probably not use the new language. To show the speed difference between using SQL as compared to NewSQL, a benchmark application must be written that compares the two techniques side by side. If possible, the benchmark should be similar to already used benchmarks in that area. The most popular database benchmark family is the TPC benchmarks. The documentation on these benchmarks is available at [www.tpc.org](http://www.tpc.org)

- Must: Implementation of a benchmark application in regular SQL and in NewSQL.
- Must: The benchmark must be at least as complex as TPC-A.
- Must: The test must be run against at least one existing popular database.
- Must: Documentation of the findings, including a graphical statistic.
- May: The benchmark may be an implementation of TPC-C instead of TPC-A. TPC-C is more complex and more commonly used than TPC-A.

### 10.2 Testing Requirements

- Must: The correctness of the benchmark application must be tested by manual inspection of the SQL statements executed against the database.
- Must: The test procedure used must be documented.
- May: 75% coverage of the application

## 11 Query Wizard

### 11.1 Functional Requirements

A query wizard must be implemented. This is an application similar to the Oracle SQLPlus Worksheet, but not necessarily with the same quality or functionality. The reason to create such an application is to show people in a simple and intuitive way (learning by doing) what NewSQL means, and how it works. This application should bridge the gap between learning NewSQL, "from the books" (by reading the documentation) and using it. This application should be very simple to use, and does not need to be feature complete.

- Must: The query wizard must be implemented in Java, using Swing, AWT or SWT.
- Must: It can come with a connection dialog to connect to a database, and / or it can connect to a fixed database (for example, the connection parameters stored in a properties file).
- Must: It must be possible to type in a NewSQL statement and / or create a NewSQL statement using a wizard.
- Must: It must be possible to execute a statement, and the result must be displayed in some form, at least as text (as in the Oracle SQLPlus Worksheet).
- Must: The application must work with at least two popular databases.
- Should: The result of the query should be displayed in another text field than the query that was used.
- Should: The wizard should allow to construct a statement using a similar mechanism than the auto-complete functionality of modern IDEs / Editors (JBuilder, Eclipse, JEdit): using a drop down list of choices, while the text is displayed and while it is still possible to type the query.
- May: As an alternative, the wizard could be a separate window, and not integrated into the editor itself.
- May: The result should be displayed in form of a grid if the result is rows from the database.
- May: If time permits, the wizard may provide the full functionality of the language.

### 11.2 Testing Requirements

- Must: The correct functioning of the query wizard must be tested manually with each SQL statement.
- Must: The test procedure used must be documented.
- May: 75% coverage of the application.
- May: A tester robot may be used to re-run the tests and verify the results.

## 12 Demo Application

### 12.1 Functional Requirements

As an additional example that proves it is easy to write or port application to NewSQL, a small example application should be written that uses NewSQL as the data access layer. The source code of the two applications may be then compared side-by-side. The code should be easy to understand (which may not be true for the Benchmark application and for the Query Wizard). This may be a new application, or an existing application that is available in source form.

- Must: Such an application must be written or ported.
- Must: The code must be well written and logical, and contain useful comments.
- Must: The application must be written in Java.
- Must: The application must work with at least two popular databases.
- Should: The application should have a graphical user interface (AWT, Swing or SWT).
- Should: All supported data types should be used.

### 12.2 Testing Requirements

- Must: The correct functioning of the demo application must be tested manually.
- Must: The test procedure used must be documented in order to re-run the test.
- Should: Each SQL statement generated should be analyzed for correctness.
- May: 75% coverage of the application.
- May: A tester robot may be used to re-run the tests and verify the results.

## 13 General Limitations

If one of the database has severe limitations that can not be worked around in a straight forward way (for example, limited identifier length, limited result set row length) then this limitations must be documented (when they are found). It is not required to find solutions for such problems. An example of this limitation is that in Oracle, an empty string is equal to null.

## 14 Publishing

The project is done in an open source style, and available on the internet. The project will be published on SourceForge ([newsql.sourceforge.net](http://newsql.sourceforge.net)). The old project NewSQL dated July 2002 may be used as a starting point.

## 15 Copyright

The documentation of this project is public domain. The software will be licensed under Apache Software License.

## 16 Popular Database

At least two popular databases must work with NewSQL. At least one of the following databases must be included:

- Oracle 9i ([www.oracle.com](http://www.oracle.com))
- MS SQL Server 2000 ([www.microsoft.com](http://www.microsoft.com))

In addition to that, one of the following should be included:

- MySQL ([www.mysql.com](http://www.mysql.com))
- IBM DB2 ([www.ibm.com](http://www.ibm.com))

Database that may be included additionally:

- PostgreSQL ([www.postgresql.org](http://www.postgresql.org))
- PointBase ([www.pointbase.com](http://www.pointbase.com))
- HSQLDB (former Hypersonic SQL) ([hsqldb.sourceforge.net](http://hsqldb.sourceforge.net))

## 17 Project Organization

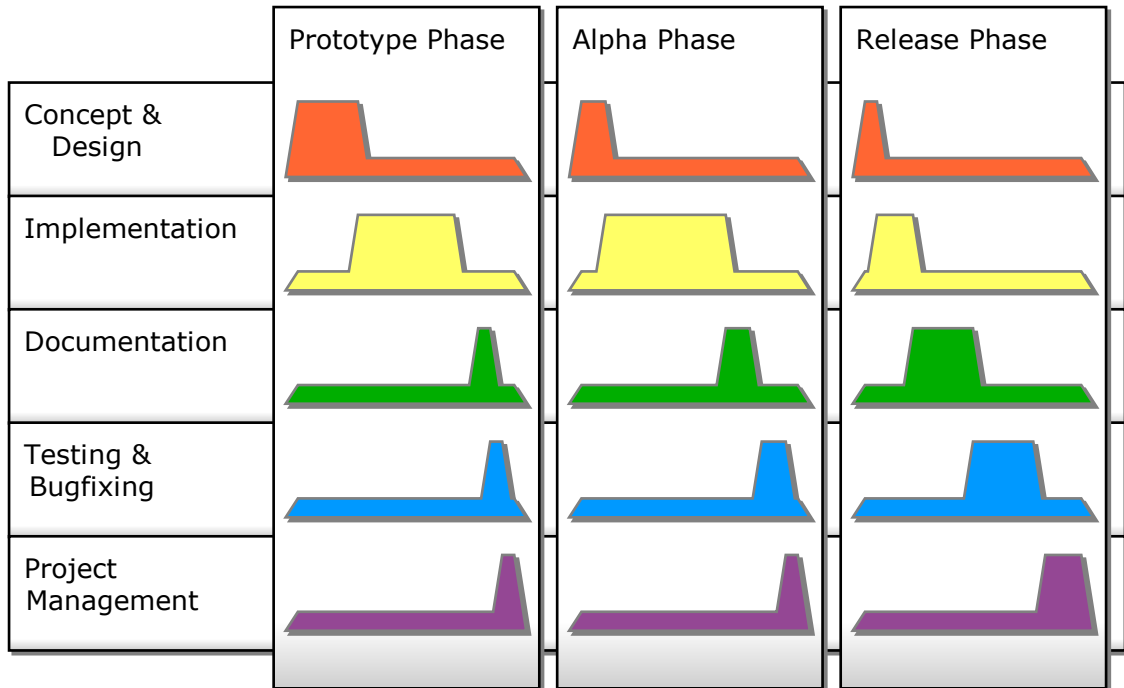
### 17.1 Involved Parties

<b>Name</b>	<b>E-Mail</b>	<b>Function</b>
Thomas Müller	<a href="mailto:thomas.mueller@swisscom.com">thomas.mueller@swisscom.com</a>	Graduate
Walter Eich	<a href="mailto:walter.eich@zuehlke.ch">walter.eich@zuehlke.ch</a>	Expert
Pierre Fierz	<a href="mailto:pierre.fierz@hta-be.bfh.ch">pierre.fierz@hta-be.bfh.ch</a>	Tutor
Dr. Jacques E. Boillat	<a href="mailto:jacques.boillat@hta-be.bfh.ch">jacques.boillat@hta-be.bfh.ch</a>	Tutor

*Table 3 - Involved Parties*

## 17.2 Approach / Processes

An iterative model will be used to develop the components. It is planned to use three iterations for this project:



*Drawing 1 - Approach*

### Prototype Phase

In the first iteration (prototype phase), a first version of the applications (JDBC driver, query wizard, benchmark, demo application) is implemented, even though the grammar of the language itself is not yet fully defined or may change. The results of the prototype phase are a set of working applications, while these applications still lack a lot of functionality. Only minimal testing and documentation will be made at this level.

### Alpha Phase

In the second iteration (alpha phase), the target is to achieve as many as possible of the main priorities. In this phase, all applications are brought to a stable and functional state, but may be buggy and lack testing and documentation. In a commercial product, this phase is called alpha phase.

### Release Phase

In the third iteration (release phase), some more functionality may be implemented. The testing is intensified, and the documentation is completed. This phase can be compared to the release of a commercial product.



## 17.6 Meetings with the Tutors

About once every three weeks a meeting with the tutors is scheduled. The content of the meeting is to discuss the current state of the project. The planned meeting dates may change, the current schedule is:

<b>Date</b>	<b>Meeting</b>
2003-05-27	Project Meeting
2003-06-17	Project Meeting & Prototype Review
2003-08-05	Project Meeting
2003-08-26	Project Meeting
2003-10-21	Project Meeting & Alpha Version Review
2003-11-11	Project Meeting
2003-12-02	Project Meeting
2004-01-06	Project Meeting; discuss remaining work
2004-01-27	Final Review Meeting

*Table 5 - Meetings*

## 18 Documentation

### 18.1 Language

All documents, as well as comments in the software, are to be written in English, in order to make the results accessible for other developers. Simple English should be used, because English is not the native language for a big part of the target audience (any kind of developer). As English is also not the native language of the author, the style of language will be lower than it would be if written in German, but this disadvantage is acceptable.

### 18.2 File Format and Location

The documents have to be published in HTML and / or PDF form. A HTML to PDF converter may be used to convert the HTML to PDF. All documentation must be published on the internet.

### 18.3 Product Documentation

The project documentation should give an easy to understand overview over the work. The new language must be carefully documented. The documentation must be detailed, while being as short as possible, so that learning NewSQL is made simple. The target audience of this documentation is the potential user of NewSQL, the developer. The contents of the documentation are:

- Must: All implemented functionality must be documented, grouped by functional groups (for example: data definition language, data manipulation language, data query language, transactions, null behavior).
- Must: The grammar must be documented. This may be done in text form (for example, BNF, or simplified BNF). The semantic must be documented.
- Must: Architecture, design and testing documentation of the NewSQL to SQL translator.
- Must: Survey results
- Must: Benchmark results
- Must: A documentation on how to install and run the software must be provided. It is not required to provide an installer (see also Development Platform / Software).
- Should: The grammar should be documented in a syntax diagram. In this case, the grammar documentation in text for is not required.
- Should: For each statement and the main variants of statements, examples should be given with explanation what the statements do.

### 18.4 Journal and Meeting Protocols

The journal must contain all important tasks and decisions. The journal reflects the history and current state of the project. It must be published online so that all involved parties can read it when required.

After each meeting between the author and the expert and/or tutors, a protocol must be written that reflects the discussed topics and decisions. The protocols will be integrated into the journal.

### 18.5 Project Report

The project report must contain:

- Must: Project progress
- Must: Achieved targets
- Must: Final report

## 19 Results

All software used or created as part of this project, as well as all documentation, must be burnt on CD. This must include the JDK, the development environment, and all documentation required to install, run and compile the software.

## 20 Rating

The rating scheme is defined as follows:

<b>Topic</b>	<b>Step</b>	<b>Weight</b>
<b>Preparation</b>	Organization and completeness of the requirements specification	3
<b>Realization</b>	Task schedule and time management	3
	Creativity, initiative, independence	2
	Choice and usage of (design-) methodology	4
	Implementation, robustness, programming style	3
	System tests (method, execution, report)	2
	Communication with the expert and tutors	1
<b>Result</b>	Conformance result / requirements specification	3
	General impression of the inspection	1
<b>Project report</b>	Correctness, completeness, comprehensibility	3
	Language, style, clarity	1
	Clear, meaningful summary	1

Table 6 - Rating

## 21 Appendix

### 21.1 Glossary / Links

#### **NewSQL**

<http://newsq.sourceforge.net>

#### **LDBC**

LDBC is an open source JDBC driver that converts standard SQL to vendor specific SQL statements and therefore allows accessing many different database types in exactly the same way. LDBC is written and maintained by the author of NewSQL.

<http://ldbc.sourceforge.net>

**TPC**

Transaction Processing Performance Council. The TPC's members include most of the largest computer system and database companies worldwide. Two major on-line transaction processing benchmarks are: TPC-A (11/1989 until 6/1995) and TPC-C (7/1992 until now).

<http://www.tpc.org>

**OQL**

OQL stands for Object Query Language and was developed by the ODMG (Object Data Management Group). However the web site of ODMG ([www.odmg.org](http://www.odmg.org)) is not available at the time of writing this document. Old versions of the web site may still be found using the internet archive ([web.archive.org](http://web.archive.org))

<http://web.archive.org>

**Ant**

Apache Ant is a Java-based build tool.

<http://ant.apache.org/>

**JDBC**

JDBC technology is an API that lets you access virtually any tabular data source from the Java programming language.

<http://java.sun.com/products/jdbc/>