

Survey – NewSQL

Author: Thomas Müller

Version: 1.0

Date: 2003-06-16

Table of Contents

1	Revision History.....	2
2	Summary.....	3
3	About the project NewSQL.....	4
3.1	The Language.....	4
3.2	The Translator.....	4
3.3	Related Project LDBC.....	4
3.4	About the Author.....	4
4	Questions on Database Usage.....	5
4.1	SQL (Structured Query Language).....	5
4.2	Other Database Access Languages.....	5
4.3	How I use (or plan to use) SQL.....	6
4.4	Databases.....	6
4.5	Programming Languages and APIs.....	6
5	Questions on NewSQL.....	7
5.1	Identifiers.....	7
5.2	Quoted Names.....	7
5.3	NULL handling.....	7
5.4	Different syntax for different databases.....	8
5.5	Autoincrement columns.....	8
5.6	Strings (text data, varchar).....	8
5.7	Style.....	9
5.8	Grammar Examples.....	9
5.9	Joins	10
5.10	Exception handling.....	10
6	Miscellaneous Questions.....	10
7	Importance.....	11
8	Contact Information (optional).....	11

1 Revision History

Version	Date	Topic	Remarks	Who
0.1	2003-05-24	Initial Draft		tm
0.2	2003-05-27	Questions	Rewrite paragraph	tm
0.3	2003-06-04	About the project NewSQL	New paragraph	tm
0.4	2003-06-12	Grammar Examples	Added the new section	tm
0.4	2003-06-12	About the Author	Added the new section	tm
0.5	2003-06-13	All topics	Re-state questions based on user feedback	tm
1.0	2003-06-16	All topics	Grammar corrections	tm

Table 1 - Revision History

2 Summary

The survey will be also suggestive and show people the problems of SQL. That is ok, as most people don't know the problems of SQL because they didn't work enough with it and don't know it in detail. Maybe this survey can convince people that NewSQL is a good thing.

3 About the project NewSQL

3.1 The Language

NewSQL is a new database access language to be developed. Targets are:

- Easier to learn than SQL
- Elegant and consistent
- Well defined and standardized

NewSQL is a replacement for SQL. It is not an extension or subset of SQL, not an object database language, and not an Object-Relation mapping tool.

3.2 The Translator

A translator from NewSQL to the various dialects of SQL will be implemented. This is done in the form of a JDBC driver. Features of the translator are:

- All major databases are supported - no porting required
- Existing data can be accessed
- JDBC interface - no need to learn a new programming API
- High speed converter ensures highest possible performance

See also: <http://newsq.sourceforge.net>

3.3 Related Project LDBC

LDBC (Liberty DataBase Connectivity) is a JDBC driver that provides vendor-independent database access. LDBC is based on ANSI-SQL and JDBC.

See also: <http://ldbc.sourceforge.net>

3.4 About the Author

NewSQL is the diploma work of Thomas Müller, Class I-99-1, Hochschule für Technik und Architektur (<http://www.hta-be.bfh.ch>), Bern, Switzerland. Thomas Müller is the original author of Hypersonic SQL (an open source SQL database written in Java, see also <http://hsqldb.sourceforge.net>).

4 Questions on Database Usage

4.1 SQL (Structured Query Language)

don't know	heard about	know	know well	Topic
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	insert, update, delete, select
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	order by
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	subqueries (nested queries)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	exists, in(.,.,...)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	count(*), min(x), max(x) or sum(x)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	group by
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	having
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	distinct
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	create table
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	create index
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	referential integrity
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	autoincrement columns / identity columns
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	sequences (Oracle style)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	transaction support (commit, rollback)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2-phase-commit (distributed transactions)

4.2 Other Database Access Languages

Other Database Access Languages

don't know	heard about	know	know well	Topic
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	OQL (object query language)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	relational algebra
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	other:

4.3 How I use (or plan to use) SQL

don't know	heard about	I did some	I did a lot	Topic
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	I run ad-hoc statements
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	I write an application that uses SQL
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	I use tools like MS Access
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	I use object-relation mapping tools
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	other:

4.4 Databases

don't know	heard about	know	know well	Topic
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Oracle
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Microsoft SQL Server
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	MySQL
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	PostgreSQL
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Microsoft Access
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	IBM DB2
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	other:
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	other:

4.5 Programming Languages and APIs

don't know	heard about	know	know well	Topic
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Java
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	JDBC API
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	C, C++
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Visual Basic
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	C#
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	LDBC (cross database SQL translator)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	object-relation mapping tools:
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	JDO
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	other:

5 Questions on NewSQL

5.1 Identifiers

Identifiers are table names, column names and so on. In SQL, they are case insensitive. In NewSQL, identifiers should be:

- Case sensitive like Java, C, C++, C#,...
- Case insensitive like in SQL, COBOL
- No opinion or I can't answer this question

5.2 Quoted Names

SQL supports a construct called 'quoted identifiers' where identifiers can include spaces and special characters (and in this case, the identifiers are case sensitive). Quoted identifiers are not supported in any other programming language.

- Quoted names should be supported
- There is no need to support quoted names
- No opinion or I can't answer this question

5.3 NULL handling

SQL supports NULL values, however the NULL behavior is different than one would normally expect. In SQL, NULL is not a value; and when comparing NULL with something else results in NULL (not FALSE and not TRUE: NULL). That means other than regular programming languages, SQL knows not only TRUE or FALSE as results for comparison, but also NULL. Some examples:

- The query: `SELECT * FROM ADDRESS WHERE NAME=NULL` does not return rows where the NAME is NULL (except for Microsoft SQL Server).
 - The query: `SELECT * FROM ADDRESS WHERE NAME=NAME` returns all rows of the address table except where the rows where NAME is NULL
 - If there is a unique index on NAME, there may still be multiple rows with the NAME NULL (as NULL is not equal to NULL) (even for Microsoft SQL Server).
 - If there is a unique index in multiple columns, there may be only one row where one column alone is NULL (for some reason); except, if all columns are NULL.
 - When ordering data (ORDER BY) then the position of the NULL rows is vendor dependent, some database sort NULL values at the beginning,...
 - The logic operators AND and OR behave differently in aspect to NULL values: AND propagates the NULL value. For OR, a null value is equal to FALSE. This is normally invisible, except if NOT is used.
- I like these NULL rules
 - I think these NULL rules are confusing and should be simplified
 - No opinion or I can't answer this question

5.4 Different syntax for different databases

Each database vendor implements his own flavor of SQL.

The following SQL statement needs to be translated to other databases:

- Oracle:

```
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255))
CREATE SEQUENCE TEST_ID
INSERT INTO TEST VALUES(TEST_ID.NEXTVAL, 'HELLO')
```
- Microsoft SQL Server:

```
CREATE TABLE TEST(ID INT IDENTITY PRIMARY KEY, NAME VARCHAR(255))
INSERT INTO TEST VALUES('HELLO')
```

- I didn't know that
- I know about incompatibilities, but never myself had problems
- I already ran into some problems / incompatibilities
- I did work on a project that involved support for multiple databases.
Please specify project and databases:

.....

5.5 Autoincrement columns

ANSI SQL does not support autoincrement columns (sometimes called identity column, sequences or automatic surrogate keys). However, most databases support autoincrement columns in some way: For example, in MS SQL Server they are called identity columns. Oracle does not support this kind of columns, but Oracle supports sequences - this is something like a user defined function that returns another number for each call.

- Autoincrement columns should be supported
- Oracle style sequences should be supported
- No opinion or I can't answer this question

5.6 Strings (text data, varchar)

In the SQL standard, the user must set the maximum size (for example, 60 characters) for string columns when the table is created. Often, it turns out later on that the size is too small, and must be changed.

Modern programming languages like Java or C++ don't have size restrictions for strings. The size restriction in SQL probably originates from the early days of computing: COBOL also knows size restrictions. However, in modern databases, variable size strings are not slower than fixed size strings.

- Each string should have a size limitation (like in COBOL)
- Strings should not have size restrictions like in Java, C++,...
- No opinion or I can't answer this question

5.7 Style

Certain features are supported by all databases, but in a slightly different way. An example is comparison syntax: in Java, the comparison operators are: ==, !=, <=, >=, <, >. In SQL, they are: =, <>, <=, >=. There are language details that define the language style, like the usage of brackets (), [], {} and uppercase / lowercase usage. I like the NewSQL to look like:

- Java, C#, Perl, Python
- SQL, COBOL
- Other:

5.8 Grammar Examples

In the following examples, please make a circle around things you like, and strike through things you don't like. You can also make remarks if you like.

SQL:

```
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255))
INSERT INTO TEST VALUES(1,'Hello')
SELECT * FROM TEST
SELECT T1.ID,T2.NAME FROM TEST T1, TEST T2 WHERE T.ID=T2.ID
UPDATE TEST SET NAME='Hi' WHERE ID=1
DELETE FROM TEST WHERE ID=1
DROP TABLE TEST
```

Java Style:

```
test=new table(int id, string name, key(id))
test.add(1,"Hello")
test.get()
t1=test; t2=test; t1.join(t2[t1.id==t2.id]).get(t1.id,t2.name)
test[id==1].set(name="Hi")
test[id==1].delete()
test.drop()
```

Simplified SQL:

```
create table test(id int, name string, primary key(id))
insert test (1,'Hello')
select test
select t1:test join t2:test on t1.id==t2.id get t1.id, t2.name
update test set id=1 where name=='Hi'
delete test where id==1
drop test
```

Which grammar do you like most?

- SQL
- Java Style
- Simplified SQL
- No opinion or I can't answer this question

5.9 Joins

In many cases, it is required to select data from multiple tables at the same time, because the data is distributed in multiple tables. In SQL, explicit joins must be used to do that.

Example:

```
SELECT Order.Price, Address.Name FROM Order, Address
WHERE Order.AddressId=Address.Id
```

An alternative syntax for this query is:

```
SELECT O.Price, A.Name FROM Order O
INNER JOIN Address A ON O.AddressId=A.Id
```

Object databases support automatic navigation that makes (most) joins unnecessary. In OQL (object query language), the statement could look like this:

```
SELECT O.Price, O.Address.Name FROM Order O
```

- NewSQL should support automatic navigation similar to OQL
- Explicit joins like in SQL are enough
- No opinion or I can't answer this question

5.10 Exception handling

Current databases throw different kinds of exceptions for the same error, for example for syntax error. Therefore cross platform applications can not rely on the exception codes.

- NewSQL should define exception codes for the most important errors
- Exception codes don't need to be the same across databases (like in SQL)
- No opinion or I can't answer this question

6 Miscellaneous Questions

- NewSQL will be good and useful (I hope)
- A new database programming language is not required; SQL is ok
- I don't need SQL, because I use object-relation mappings
- It is a problem that no big company supports NewSQL

Remarks:

.....

7 Importance

How important are this features (Very: very important; Not: not important):

Very	4	3	2	Not	Don't	
					know	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	update, insert, delete, select
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	referential integrity
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	stored procedures
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	build in functions
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	user defined functions
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	select with join
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	ordering
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	grouping
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	distinct
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	outer joins
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	having
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	subqueries
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	views
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	transaction support
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	savepoints
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2-phase-commit (distributed transactions)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	type casting
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	insert from a select
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	access rights: user names, password
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	access rights: roles
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	table level access restrictions
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	fine grained restrictions (column level)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	insert: default values

8 Contact Information (optional)

Name:

E-Mail: