

## Survey Results – NewSQL

---

Author: Thomas Müller

Version: 0.3

Date: 2003-08-28

## Table of Contents

1	Revision History.....	2
2	Summary.....	3
3	Survey Process.....	3
3.1	Paper Survey.....	3
3.2	Online Survey.....	3
3.3	Adressing the Public.....	3
3.4	Lessons learned.....	4
4	Questions on Database Usage.....	4
4.1	SQL (Structured Query Language).....	4
4.2	Other Database Access Languages.....	5
4.3	How I use (or plan to use) SQL.....	5
4.4	Databases.....	6
4.5	Programing Languages and APIs.....	7
5	Questions on NewSQL.....	7
5.1	Identifiers.....	7
5.2	Quoted Names.....	8
5.3	NULL handling.....	8
5.4	Different syntax for different databases.....	8
5.5	Autoincrement columns.....	9
5.6	Strings (text data, varchar).....	9
5.7	Style.....	10
5.8	Grammar Examples.....	10
5.9	Joins .....	15
5.10	Exception handling.....	16
6	Miscellaneous Questions.....	16
7	Importance.....	18
8	Other Interesting Results.....	19
8.1	Operating System.....	19
8.2	Browser used.....	19

## 1 Revision History

<b>Version</b>	<b>Date</b>	<b>Topic</b>	<b>Remarks</b>	<b>Who</b>
0.1	2003-08-18	Initial Draft		tm
0.2	2003-08-27	Added newest results		tm
0.3	2003-08-28	Formatting changes		tm

*Table 1 - Revision History*

## 2 Summary

The results of the survey are promising. Quite many people (64 so far) took part, and a lot of useful feedback has been sent. This document analyses the results of this survey.

## 3 Survey Process

### 3.1 Paper Survey

The survey was originally ment to be mainly a paper survey. The survey document was printed, then given to a few 'survey beta testers' at the workplace of the author (Swisscom IT Services AG, Switzerland). Some little problems with the survey where corrected (mainly spelling, some clairification).

After that, the survey was printed and given to the teachers of two classes: the computer science class of the school 'Höhere Fachschule für Technik und Architektur, Bern' and the 'Software Schule Bern'. In addition to that, the survey was handed out to some co-students of the author.

Unfortunately, only few students filled out the paper survey. The main reason is probably that the students did not have enough time. In the case of the 'Software Schule Bern', the survey papers were lost. Because the online survey was ready at that time and it was already almost vacation time, the papers were not printed again and instead, the online survey was used.

### 3.2 Online Survey

Originally, the idea was to a use static html form, where the results of the survey are sent as a regular e-mail. But that would mean people could not fill out the survey anonymously. It is not clear if this would be a problem, but it is likely that less people would have filled out the survey if it was not anonymous.

It was found that PHP script can be used in Sourceforge, and so it was possible to create a anonymous survey. By registering the IP-number of the sender, it was still possible to guarantee that each person could fill out the survey only once.

The survey was also published on the internet, on the project web site.

### 3.3 Adressing the Public

After the survey was online, this was announced at the website 'freshmeat.net'. Freshmeat is a news service for (mainly open source) software projects. If there is a new version of an open source software available, this is announced at Freshmeat. This site is quite popular among developers, and so many people (probably about half) filled out the survey because the heard about it in Freshmeat.

In addition to that, the autor send an email to his ex-coworkers in California (at PointBase) and Switzerland (Miracle Software AG). Some of them forwarded the mail to other developers, and like that quite a lot of people heard about it.

### 3.4 Lessons learned

It seems to be easier to only publish the survey online. By using a paper survey, it is hard to distribute the survey very far. It is a lot easier to use an online survey.

There are technical risks involved in using the online survey. For example, it is hard to authenticate the users. Somebody might fill out the survey multiple times, by using multiple IP-numbers. But given the nature of this survey, it is not likely that somebody did that. It just seems nobody would have an interest in manipulating the survey.

For more important surveys, some mechanism to authenticate the user should be built in.

## 4 Questions on Database Usage

### 4.1 SQL (Structured Query Language)

The first question was about what features of the SQL language are known and used. Here the list sorted by most-known feature to least-known feature:

<b>Question</b>	<b>know well</b>	<b>know</b>	<b>heard about</b>
insert, update, delete, select	54	9	0
order by	54	8	1
count(*), min(x), max(x) or sum(x)	47	12	3
create table	43	15	3
group by	40	17	2
distinct	38	14	8
subqueries (nested queries)	35	22	5
transaction support (commit, rollback)	34	20	5
create index	32	23	5
exists, in(.,.,...)	32	20	7
having	25	23	8
autoincrement columns / identity columns	24	19	11
referential integrity	23	19	13
sequences (Oracle style)	16	9	17
2-phase-commit (distributed transactions)	10	19	20

## 4.2 Other Database Access Languages

There are other database access languages besides SQL. This question was about what other languages are in use.

<b>Question</b>	<b>know well</b>	<b>know</b>	<b>heard about</b>	<b>don't know</b>	<b>no answer</b>
relational algebra	8	19	16	18	4
other	6	4	0	8	47
OQL (object query language)	2	10	21	27	5

This result is somewhat surprising. It seems a lot of people heard about OQL, but actually only very few people ever used it.

The result for relational algebra is maybe biased because this language is taught by the 'Höhere Fachschule für Technik und Architektur, Bern'. It would be interesting to know how many people know it beside the students of this school.

Here the list of 'Other' access languages (not ordered):

- ADABAS + NATURAL, DAO, EnterpriseObjects Framework (WebObjects)
- FileMaker, JDBC, PL-SQL, QBE, QBE, ISAM, RDBMS SQL
- logic programming, xpath, xquery, xml tree merges

## 4.3 How I use (or plan to use) SQL

This question is about in what context SQL is normally used.

<b>Question</b>	<b>I did a lot</b>	<b>I did some</b>	<b>heard about</b>	<b>don't know</b>	<b>no answer</b>
I write an application that uses SQL	44	12	2	1	5
I run adhoc statements	44	9	2	3	6
I use tools like MS Access	11	22	13	9	9
I use object-relation mapping tools	10	10	24	14	6
other	9	2	0	4	49

Where others are:

- FileMaker, Python, Zope, SQL Explorer, Toad, PowerCenter
- I use SQL to generate SQL statements
- I write SQL translators
- WebObjects: SQL statements encapsulated by eof
- batch processing with cmdline tools that send SQL queries and check on the result
- data warehousing, data marts, modeling
- ported a legacy filemaker (yuck!) database to sql
- writing for technical manuals

## 4.4 Databases

This question is about what database is used.

<b>Question</b>	<b>know well</b>	<b>know</b>	<b>heard about</b>
MySQL	18	28	13
Microsoft SQL Server	18	24	14
Oracle	18	23	16
Microsoft Access	13	26	16
PostgreSQL	7	16	26
IBM DB2	4	15	31
other	12	15	6

The audience of this survey seems to use and know quite many different databases, including open source software.

Other databases mentioned:

<b>Database</b>	<b>Count</b>
PointBase	8
Firebird/Interbase	4
Informix	3
Sybase	3
Cloudscape	1
DB-XML	1
Openbase	1
SQLBase	1
RDBMS	1
SQLite	1
Versant	1
Clipper	1
FileMaker	1
Hypersonic SQL	1
db4o	1
Ingres	1
Foxpro	1
Pervasive.SQL	1
Frontbase	1

The Java database PointBase was probably known by quite many people because the survey was also sent to some employees of PointBase; the average developer probably doesn't know PointBase so well.

## 4.5 Programming Languages and APIs

This question is about what the developer used so far.

<b>Question</b>	<b>know well</b>	<b>know</b>	<b>heard about</b>
C, C++	25	22	11
Java	23	29	8
JDBC API	23	15	16
other	15	3	0
Visual Basic	10	20	18
object-relation mapping tools	7	5	8
C#	4	9	38
JDO	1	3	22
LDBC (ldbc.sf.net)	0	1	11

The O/R Mapping tools used are:

- Castor, Hibernate, Rational, TopLink, Class::DBI, JDX, WebObjects
- WLS tools for CMP, CMR
- miracle xrp kernel
- Self-made stuff for adhoc display
- Wrote my own

Other languages / APIs used are:

- Delphi, Ada, ASP, FileMaker, HTML, IDAPI/BDE, Lisp, ODBC, Perl
- PHP, Python, Shell, The Third Manifesto, Zope

## 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:

<b>Answer</b>	<b>Count</b>
Case insensitive like in SQL, COBOL	32
Case sensitive like Java, C, C++, C#,...	24
No opinion or answer	8

## 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).

<b>Answer</b>	<b>Count</b>
There is no need to support quoted names	35
Quoted names should be supported	19
No opinion or answer	10

## 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.

<b>Answer</b>	<b>Count</b>
I think these NULL rules are confusing and should be simplified	38
I like these NULL rules	18
No opinion or answer	8

## 5.4 Different syntax for different databases

Each database vendor implements his own flavor of SQL.

<b>Answer</b>	<b>Count</b>
I already ran into some problems / incompatibilities	32
I know about incompatibilities, but never myself had problems	18
I did work on a project that involved support for multiple databases	11
I didn't know that	3

There was a logical mistake in the the online survey, because the answer 'I did work on a project that involved support for multiple databases' should be a question on it's own. Still, it is clear that most people know about compatibility problems. It also seems most developers have to deal with more than one database. Projects that involve multiple databases seem to be quite common.

## 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.

<b>Answer</b>	<b>Count</b>
Autoincrement columns should be supported	38
Oracle style sequences should be supported	14
No opinion or answer	12

This is a clear vote for autoincrement column types. Oracle style sequences are slightly more complicated to use, that is probably why most people would like autoincrement columns.

## 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.

<b>Answer</b>	<b>Count</b>
Strings should not have size restrictions like in Java, C++,...	44
Each string should have a size limitation (like in COBOL)	10
No opinion or answer	10

It seems the size limitation of fields is not popular.

## 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:

Answer	Count
Java, C#, Perl, Python	45
SQL, COBOL	14
Other	2
No Answer	3

Modern programming languages like Java are the clear winner. Others are:

- Ruby -- 'non-strict but non-ambiguous punctuation, keywords used to reduce symbol-gibberish'
- Both

## 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
```

- All-caps hard to read; re-use of operator (=) confusing; word order isn't natural language nor logical order; Abbreviated keywords (INT) are hard to remember.
- Ausser dem Create und Insert Statement sind mir alle Statements sympatisch. Diese braucht ich aber auch wenig, daher evt. diese Sympathie.
- Ba: You have to read very far into the code to see what tables are referenced. Goo: Mostly readable like text. But I \_dont\_ like SQL
- Doesn't gotta be upper-case, yanno :)
- I like that it's really simple and concise.
- Hey, for joins, use ANSI SQL92 grammar !
- I kind of like the verbose style of sql
- I like the syntax which is more like English

- I think that these examples show pretty good that SQL is not \*that\* hard to understand. On the other hand, I'm probably biased because I'm fairly good at reading SQL.
- It's okay
- Know it, which is not to say like it.
- Like All
- Like it since I am used to the syntax
- Like: simplicity
- Recht bekannt.
- Relational entities should not have accessible object handles like ID. The INSERT should demand the table attributes be listed as when as values. The WHERE clause on UPDATES should be a general SELECT-like WHERE clause. CREATE, DROP, TRANSACTION, and COMMIT should be more implicit.
- `SELECT T1.ID,T2.NAME FROM TEST T1, TEST T2 WHERE T.ID=T2.ID`
- SQL has some standards. I think this is the main advantage of this example.
- ```
Tables MyTable = new Table;  
    MyTable.AddNeww(1,"Hello");  
    Arrays MyArray = MyTable.Records('T1.ID','T2.NAME').Filter('T.ID=T2.ID');  
    MyTable.Name.Filter('ID=1') = 1;  
    MyTable.Delete.Filter('ID=1');  
    MyTable.Drop;
```
- The insert statement isn't guaranteed to work, since SQL doesn't guarantee the insert order of columns unless you declare it.
- Well, you made a typo on line 4: T.ID
- even non-programmers can grasp what's going on - flattens the learning curve. I think it's charming.
- familiar.
- i don't understand this
- literate, maintainable and mechanical.
- okay
- `select a as b, b / 10 from c;`
- this doesn't work on mysql but should.
- well, i hate sql syntax

**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()

```

- Don't like: treating table as object in the language, since that forces the same OO rules to be used in any abstraction. Perl has different SQL abstraction modules and people have a choice.
- Easy to understand.
- test[id==1].set() is resuing the syntax for accesing an array element. I am not sure that it is a good idea.
- Geil.
- I like this!
- I think this looks good. I'm a C/C++ user, so I'd like to have a ';' after a statement.
- What I don't like here (or just would need to get used to) is 'test[id==1]'. This implies to me that id==1 returns a boolean value which isn't true here..... Remains to be seen how things like sub-queries look in this syntax.
- Java style seems a bit funny at first, but it reads quite good. The join example seems somewhat obfuscated, but I guess one would be able to write it differently in the end product.
- Like:
 

```

test.drop()
test.add(1,"Hello")

```
- Many many time I had to write update/insert statment for data. becuae the staments differ it led to annoying bugs (when you have 20 or so fields it is really hard to maintain).
- Insert/Update staments should look and behave the same (except of course for the condition:
  1. test.add(id=1,name="Hello")
  2. test[id==1].set(name="Hi")
- Might be a little tougher for non-programmers, like web designers, to grok.
- Much like. Would like even more if syntax was identical to something, say, like Torque-generated beans support.
- Not bad, like the programmatic style
- Perhaps one of the major advantages to this approach is easier input validation. Less chance for SQL insertion.
- Statement #4 is not very natural. Only good programmers (non-SQL) would probably like this in my opinion.

- Tables should not be objects.
- This is a completely new language. You could also say it's C# like or C++ like. I'm not sure if it is a good idea to invent a new language which "looks like" Java but isn't java (in the case of procedural elements). At the end, if you would like to include something like PL/SQL, you'll reinvent Java too. A possible solution is to make a generator which generates the rest around similar to SQLJ.
- Also I'm not sure if subqueries or all aspects of them are possible that way (see simplified SQL).
- This is a new wave and i like it most. However i think test.drop() should be like "destroy test" - i.e. unlike Java's auto GC, tables cannot be garbage collected, if not used 8-), so semantically dropping a table is an explicit destruction construct, which should be reflected in syntax.
- ND GOD DAMN, WHERE IS SEMICOLONS at the end of lines.
- This style is absolutely readable
- Ugly, hybrid syntax: is neither java nor SQL, but takes poor features from both. Uses symbols with non-intuitive uses -- [] should be a matrix/subscript operator, not a filter operator. semicolons separating things that are not statements is poor. Empty parentheses are annoying. Syntax seems to draw from CSS, Xpath, java, javascript, c#, perl and python all at once, but for no reason.
- Very confusing syntax
- Wouldn't want Java-Style SQL-Statements
- finde ich unübersichtlich
- i like this a lot better. when you consider that lots and lots of db access is through software and no longer via mildly trained db operators, it makes a lot of sense to recast the query language into a truly structured and coherent modern language.
- i think to understand this. test.add is nice.
- FileMaker also works in this style. But for selects FileMaker (3.0) ist somehow interaktive. (to be checked)
- intuitive for me, I do a lot of Java. if I need to work on a database, I prefer statements in a simple language. the fourth line is overloaded with dots and () for what it does. overall: kinda miss the semicolons ...
- is not sql.
- looks good
- looks strange, the syntax is not db optimised and probably cumbersome in some cases. It does not feel stateful but it is (my guess). what is the scope of the variable? a bit confusing
- means I have to learn a new language, but to what overall benefit.
- not bad
- ok
- too techie..feels too much like programming

**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

```

- As long as the table names are short, this example is really easier. If the table names are longer, you'll write more code for that. Also I see some problems if you would like to use subqueries with columns from the "superquery" e.g.
 

```

select * from a where exists (
  select * from b where b.somecolumn=a.othercolumn)

```
- If it is possible to keep this possibilities, there is no reason not simplify SQL.
- Bäh, nix halbes und nix ganzes.
- Column selection is not clear
- Do not like the use of ':' in the column:table relation.
- Don't like: join syntax, it's very unclear
- Generally, I like the examples of simplified SQL the least. They may be simplified, i.e. use less keywords, but most of the time this removes clarity. Example 1 "drop test": now, what exactly am I dropping here? A table or a database or something else? Example 2 "select test": again, what is it exactly that I am selecting here?
- I don't like statement #4. The SQL version reads more naturally in my opinion. Also, isn't statement #5 different from #5 in SQL above? Here you are updating the ID value, but in SQL above you are updating NAME.
- I expect SQL to look like SQL
- I like the db optimised syntax. but to be honest i like the SQL join command the most
- I'd prefer SQL-Style
- Keywords like FROM aids in parsing. If you can manage without these key words, I do not have any problem.
- Like All except:
 

```

select t1:test join t2:test on t1.id==t2.id get t1.id, t2.name
update test set id=1 where name=='Hi'

```
- Relational entities should not have accessible object handles like id.
- This is much like the old SQL. and i think, that trying to support all the features of SQL this way you will end up with ANSI SQL syntax, which is hard to parse, requires a lot of lookahead to figure out what's the task.
- Though I voted mostly for Java7C++ style this doesn't look too bad..... I'll select Java in the querstion below, but with hesitation.

- Too similar to SQL to avoid confusion - I think it'll just fall between the stools
- Very readable and self-explanatory.  
Less typing
- also not bad.
- hmm, that isn't too bad, actually...
- i still not understand the part with update
- key statement should be outside the field list in "create"
- no major advantage
- not bad, enough like SQL that it's easy to see what it means (and to learn.) The join syntax looks more complex than SQL. Rest is pretty good.
- ok but how is the following query saving time?  
select t1:test join t2:test on t1.id==t2.id get t1.id, t2.name  
- seems bit more confusing to me
- ok but not literate
- sieht doch gut aus
- upper-case is better and adds to readability. apart from that, I don't prefer this notation over SQL.

### Which grammar do you like most?

| Answer               | Count |
|----------------------|-------|
| SQL                  | 24    |
| Java Style           | 19    |
| Simplified SQL       | 14    |
| No opinion or answer | 7     |

This is probably the most important question, but unfortunately the result is not clear. Most people seem to like SQL, but it may be only because it is familiar. Or – as a recent study in USA showed – because SQL is the first answer.

## 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.

| Answer                                                    | Count |
|-----------------------------------------------------------|-------|
| Explicit joins like in SQL are enough                     | 28    |
| NewSQL should support automatic navigation similar to OQL | 23    |
| No opinion or answer                                      | 13    |

## 5.10 Exception handling

Current databases throw different kinds of exceptions for the same error, for example for syntax error.

| Answer                                                                   | Count |
|--------------------------------------------------------------------------|-------|
| NewSQL should define exception codes for the most important errors       | 49    |
| No opinion or answer                                                     | 11    |
| Exception codes don't need to be the same across databases (like in SQL) | 4     |

## 6 Miscellaneous Questions

| Question                                                       | Yes | No |
|----------------------------------------------------------------|-----|----|
| NewSQL will be good and useful (I hope)                        | 39  | 25 |
| A new database programming language is not required; SQL is ok | 17  | 47 |
| I don't need SQL, because I use object                         | 5   | 59 |
| It is a problem that no big company supports NewSQL            | 17  | 47 |

Remarks:

- A cross-vendor SQL would be useful especially if you're developing for multiple platforms, as it would simplify your code. I think the more SQL-like it is, the better chance it has. Simplifications would be good, but should I think it should support standard SQL syntax also.
- But if it is going to be good and useful, NewSQL needs desperately to get back to more of the relational roots of relational databases rather than have compatibility with existing databases as one of its primary goals. I am pessimistic anything good will come of this effort because of the apparent misunderstanding regarding the role of NULL evidenced above.
- I appreciate your desire to make SQL more transparent for application developers - however, the areas where SQL databases excel is in the management of gigabytes/terabytes of data. The realm of large databases is what gives rise to the need for optimized and compiled SQL, much in the same way that driver and kernel development require C/C++ code. I can't count the number of times I've had to rewrite SQL statements generated by a mapping layer to correct performance issues. I do agree with your comment about lack of standardization - I've had to learn specific coding styles to be successful as a DBA.
- I think this is a great idea! if it will be done well we might be able to convince an open source data base to support the idea of NewSql.

- 
- I think your biggest challenge will be fighting the SQL mindset that has so many people are already comfortable with (regardless if it is the best way to do it or not). SQL might not be a perfect solution and in some ways it is dated, but trying to get people to move to another database access language might prove difficult.
  - I've been using Torque under Jython (java-based python) to do a lot of database work recently and have been pretty impressed--except perhaps for joins I think it ends up being a pretty nice syntax.
  - Ich habe Dein hypersql gerne eingesetzt, klasse.
  - Ich bin auch auf der nach einem guten Mittel zwischen SQL und OO. NEWSQL wird große Akzeptanz finden, da sehr viele Leute auf Python und Java gehen werden. . NET ist zuuuuu langsam und im Moment zu instabil, außerdem ein Technologiebruch im M\$ Konzern, der denen den Hals brechen wird, weil viele sich nach Linux, PostgreSQL und Migrationspfaden umschaauen.
  - If you choose to implement a completely new syntax/style/language you will lose a lot on people who already know SQL and have to learn something completely new. On the other hand, you will gain from SQL beginners who will (hopefully) benefit by all the improvements of NewSQL.
  - Make NewSQL very easy, so even I can use it! ha!
  - NewSQL could be useful but given that there are gazillion applications out there - it is better to have something that is like an extension rather than rewrite. Support for legacy apps is very important. Companies do not have time/resource to rewrite applications.
  - Nice idea at least, but object relational mapping is in a good usable state.
  - One trouble is that people know SQL. It's a "standard" (albeit, not a closely followed one).
  - SQL is ok as a general purpose DB-access language. To make life easier for application developers I'd propose using some object-relation mapping tools.
  - Sincerely... Good luck.
  - Some of these questions are a little hard to answer without more explanation of what the intent of NewSQL is.
  - Tables  
MyTable = new Table;  
MyTable.AddNeww(1,"Hello");  
Arrays MyArray = MyTable.Records('T1.ID','T2.NAME').Filter('T.ID=T2.ID');  
MyTable.Name.Filter('ID=1') = 1;  
MyTable.Delete.Filter('ID=1');  
MyTable.Drop;
  - Thomas could you check FileMaker? This is a tool where some not programmers created nice applications. Maybe there are some ideas what to avoid, if you wants something easy.
  - The most important for PostFinance would be that High Performance from one to the other Database Vendor is also supported. Application programmers should use a meta model.
  - Try to keep the newSQL grammar clear and simple, and base it only on a couple of rules. The Java language is a good example to start from.

- What about triggers? I would rate its importance = 3.
- You need to consider a sounder theoretical basis for newSQL. I would recommend a pure relational approach along the lines of Date and Darwin's Tutorial D language ([www.thethirdmanifesto.com](http://www.thethirdmanifesto.com)). Check out [www.dataphor.com](http://www.dataphor.com) for a commercial attempt.
- da hast du dir was vorgenommen!
- good luck on the whole thing. I'll try to follow it.

## 7 Importance

How important are this features. Sorted by most important to least important:

| <b>Question</b>                           | <b>very</b> | <b>less</b> | <b>less</b> | <b>least</b> | <b>don't know</b> |
|-------------------------------------------|-------------|-------------|-------------|--------------|-------------------|
| update, insert, delete, select            | 58          | 3           | 0           | 0            | 3                 |
| select with join                          | 45          | 6           | 4           | 3            | 6                 |
| ordering                                  | 44          | 12          | 4           | 0            | 4                 |
| transaction support                       | 43          | 5           | 6           | 3            | 7                 |
| access rights: user names, password       | 39          | 9           | 6           | 4            | 6                 |
| grouping                                  | 38          | 15          | 6           | 1            | 4                 |
| distinct                                  | 36          | 10          | 6           | 3            | 9                 |
| referential integrity                     | 32          | 11          | 5           | 2            | 14                |
| access rights: roles                      | 31          | 12          | 8           | 7            | 6                 |
| subqueries                                | 30          | 18          | 8           | 3            | 5                 |
| build in functions                        | 29          | 16          | 9           | 4            | 6                 |
| stored procedures                         | 23          | 18          | 7           | 9            | 7                 |
| insert from a select                      | 22          | 22          | 8           | 6            | 6                 |
| outer joins                               | 22          | 11          | 16          | 4            | 11                |
| user defined functions                    | 21          | 17          | 9           | 10           | 7                 |
| having                                    | 20          | 13          | 17          | 3            | 11                |
| insert: default values                    | 20          | 9           | 15          | 13           | 7                 |
| views                                     | 19          | 18          | 10          | 6            | 11                |
| table level access restrictions           | 19          | 13          | 14          | 8            | 10                |
| type casting                              | 11          | 15          | 16          | 11           | 11                |
| savepoints                                | 11          | 10          | 12          | 13           | 18                |
| 2-phase-commit (distributed transactions) | 11          | 7           | 14          | 11           | 21                |
| fine grained restrictions (column level)  | 9           | 10          | 16          | 17           | 12                |

## 8 Other Interesting Results

It was absolutely not one of the targets of this survey to create operating system or browser statistics. However, it is very simple to collect that data in a online survey, as no user input is required.

### 8.1 Operating System

| <b>Operating System</b> | <b>Count</b> |
|-------------------------|--------------|
| Windows                 | 52           |
| Linux                   | 8            |
| Mac OS                  | 3            |
| FreeBSD                 | 1            |

About 80% of the people used a Microsoft Windows operating system. Of the rest, Linux takes the big share (about 12%).

### 8.2 Browser used

| <b>Browser</b>    | <b>Count</b> |
|-------------------|--------------|
| Internet Explorer | 33           |
| Mozilla           | 25           |
| Konqueror         | 3            |
| Opera             | 2            |
| Safari            | 1            |

Most people use the Microsoft Internet Explorer, but about half use Mozilla and others. Other recent (July 2003) browser usage statistics show 90% Internet Explorer usage or more. The audience (mainly software engineers) of this survey are different from the common internet user. Maybe the reason is that software engineers are early adopters.