HEAD TO HEAD

Postgres and MySQL in direct comparison

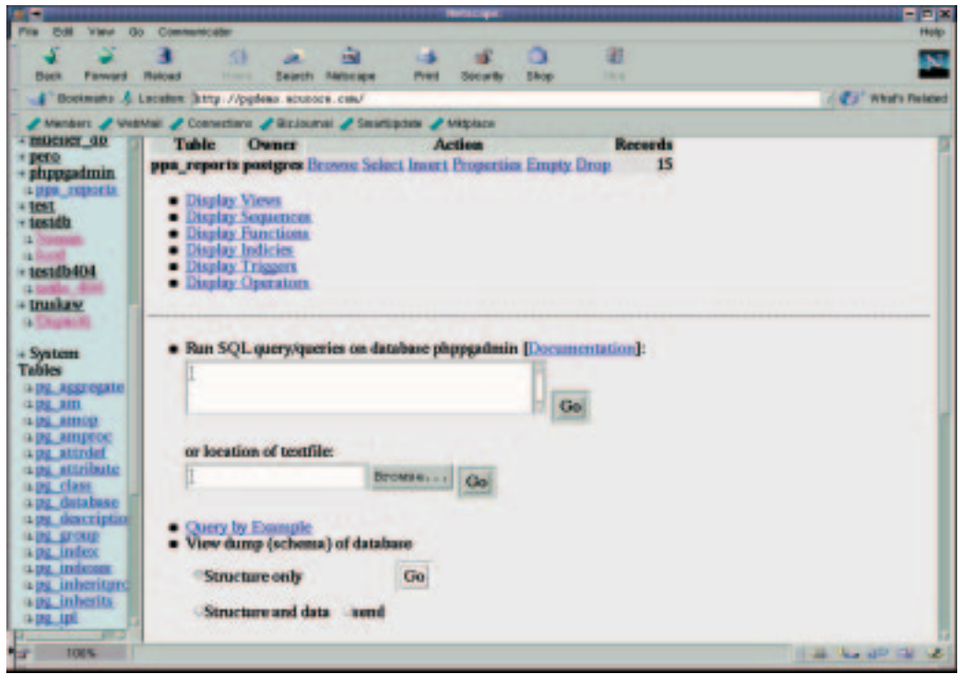
DIRK GOMEZ, HAGEN HÖPFNER

MySQL and Postgres are database systems that have long been available as open source and are especially popular as back ends for web applications. In this direct comparison, both of them have to show what they can do.

Postgres (official title: PostgreSQL) is an object-relational database Management System (DBMS), whose source code and documentation are freely available under the BSD licence and which is being worked on by numerous developers all over the world. It runs on almost all platforms such as Solaris, SunOS, HPUX, AIX, Linux, Irix, FreeBSD and most Unix variants. The current stable version is 7.0.3.

Development of Postgres began in 1977 at Berkeley University under the name of Ingres. From 1986 to 1995 Michael Stonebraker, a professor at this university, was working on the development of an object-relational DBMS named Postgres (after-Ingres). The firm Illustra marketed the code as a commercial product and was later bought up by Informix. In 1994 two Berkeley doctoral students, Jolly Chen and Andrew Yu, added the ODBMS SQL and renamed it Postgres95.

Relatively quickly, it became clear to them how much interest there is in an open source DBMS: The mailing list had by then already signed up over 1000 subscribers. With a total of 250,000 lines of C code, though, it takes some time to get used to, so it was necessary "to find a few people with lots of time, rather than finding a lot of people with a bit of time" (Jolly Chen). Today there are some 15 master developers, who supervise the code of their own



PHP PGAdmin, the PHP-supported web front end for Postgres in action.

respective modules, and numerous additional voluntary developers. New features can be added quickly due to the large number of developers.

Postgres Installation

Installation of Postgres is simple, most Linux distributions contain corresponding packages. In some distributions, for example Mandrake 7.x, the installation program installs Postgres in full. Otherwise the source can be downloaded from the Postgres website and you can compile it yourself. Constructing databases is also very simple and is well described in the excellent documentation.

Postgres Standard functions

Postgres largely adheres to the SQL92 standard. Transactions, foreign keys and user views are no problem; nor, since Version 7.0.2, are outer joins.

Glossary

A couple of important terms from the domain of databases:
A database (DB) is actually nothing more than a loose collection of data, regardless of their storage method. The term of database management system (DBMS) encompasses all software modules necessary to manage a database. A database application (DBA) is an application program that sets up a database management system. If DBA and DBMS are considered together, this is called a database system (DBS).

Nested queries, so-called Subselects using *SELECT*, are possible:

```
SELECT city FROM weather
   WHERE temp lo = (SELECT max(temp lo) FRO?
M weather);
```

Postgres Data integrity/Transactions

Foreign keys help the developer to determine the correctness of data using database resources: Foreign Key Constraints produce a dependency between one or more columns of a table with one or more columns. When data is changed, this dependency is checked, so inconsistent changes are no longer possible.

Postgres is an ACID database (see box "Transactions..."):

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

Transactions are supported at row level and secured by an internal version management system (Multi Version Concurrency Control), a process which is considerably harder and more time-consuming to implement than the usual locking of other database systems. With the MVCC process, readers do not wait for writers: Each transaction sees a corresponding database snapshot, data changes made by change transactions running concurrently are not followed up in this snapshot.

Postgres Expanded functionality

As in almost all professional database systems, triggers and stored procedures are familiar. On the

other hand, the option of replicating databases is not currently implemented.

Postgres is catalogue based. In the catalogue, not only the tables, but also data types, functions and access methods are stored. This makes this database system easy to expand. Even inheritance is possible:

```
CREATE TABLE cities (
 name      text,
 population float,
 altitude   int
);

CREATE TABLE capitals (
 state     char(2)
) INHERITS (cities);
```

The instances in the table *capitals* inherit all attributes of the parent node *cities* (*name*, *population*, *altitude*). In order to query all cities – including capital cities – that are located at a height of over 500 metres, the SQL statement specified on the next page can be used:

```
SELECT c.name, c.altitude
 FROM cities* c
 WHERE c.altitude >500;
```

The asterisk * tells the SQL parser that the query is interrogating the table *cities* and all tables descended from it in the class hierarchy. The relational model has only atomic attributes, but Postgres also has arrays:

```
CREATE TABLE SAL EMP (
 name          text,
 pay_by_quarter  int4[],
 schedule       text[][]
);

INSERT INTO SAL EMP
 VALUES (`Ulrich',
 `{10000,12312, 9423, 18400}',
 `{{"meeting", "beer"}, {}}');

INSERT INTO SAL EMP
 VALUES (`Tom',
 `{14000,11000, 9200, 12000}',
 `{{"meeting", "lunch"}, {}}');

SELECT name
 FROM SAL EMP
 WHERE SAL EMP.pay_by_quarter[1]
      < SAL_EMP.pay_by_quarter[2];
```

A table with a one-dimensional and a two-dimensional array is created, to which a line is added. After that all names of colleagues are queried, who earned more in the second quarter than in the first quarter.

Triggers and Stored Procedures

Triggers and stored procedures are used in networks to minimise server round trips. Both are executed within the database, there is no network traffic between application and database. In a normalised data model triggers are mainly needed for auditing and to improve performance, if they are to be transparent for the application:

- Auditing is the storage of change accesses to objects: A statement changes a dataset, a trigger saves these changes in an audit table.
- Data models are often made non-standard in order to improve performance. The information is made redundant, because then faster accesses are possible with corresponding indices.

Stored procedures are another means of moving application logic within the database. These are proper programs with control structures that are executed within the database and can deliver results. So for example an HTML page can be prepared within the database and sent complete back to the web server.

Triggers and stored procedures are created in a programming language called PG/SQL, which is based on Oracle's PL/SQL, or in PL/Tcl or PL/Perl. So it

Benchmarks

Great Bridge, a firm offering professional support for Postgres, issued a press release in August 2000 about a TPC-C-Benchmark, where Postgres performed better than MySQL and two commercial DBMSes. Postgres had no problem keeping up with its commercial competitors, while the other two open source products (MySQL and Interbase) fell behind sharply as their load increased.

But the benchmark software used ODBC – this test is therefore to be evaluated with caution, because many ODBC drivers are poorly programmed. Tim Perdue of Sourceforge and PHPBuilder reports on a somewhat more realistic test, where he uses Postgres to stand in for MySQL for part of a PHP-based web site. In this case, Postgres itself scaled about three times better, though page generation does take considerably longer.

In the insert test MySQL rapidly collapsed, as it is based on table locking, while Postgres has no problems even with long insert transactions, due to its smart locking algorithm.

MySQL also shows, at http://www.mysql.com/information/benchmarks.html numerous benchmark comparisons of database systems, all based on the test suite developed by MySQL itself, and which were also sometimes executed with ODBC.

is possible to develop within the database with these two script languages. Overall, the Postgres-SQL dialect and the database's own programming language are almost of equal value to those of commercial databases and applications can be ported at reasonable expense.

Postgres - interfaces and clients

Postgres offers numerous interfaces to other programming languages or systems, including:

- LIBPQ, LIBPQEASY
- ECPG and LIBPQ++ for C++
- ODBC
- TCL
- Python
- Ruby
- JDBC
- Perl
- PHP
- Delphi

The migration tools from Microsoft Access are of interest for those migrating. But PHP PGAdmin, the Postgres equivalent to PHP MyAdmin, the popular web front end for MySQL, is still at the nursery stage. It is certainly not that easy to emulate the greater complexity of Postgres on an ergonomic web interface. The acceptance of Postgres will, however, benefit enormously from such a tool.

Anyone who wants to can go to *http://pgdemo.acucore.com*, where you can make databases and tables to your heart's content and thereby become more familiar with the interface and the database. Most graphical front ends such as Glider and GtkSQL as well as KSQL also work with Postgres. Often, though, the developers have taken MySQL as their starting point, so that Postgres-specific features have only insufficient support.

Info on Postgres

The FAQ-Maintainer Bruce Momjian: PostgreSQL: Introduction and Concepts, Addison Wesley, ISBN:0201703319, online in HTML format: http://www.postgresql.org/docs/awbook.html
Postgres sites: http://www.postgresql.org/ and http://www.pgsql.com/
Bruce Momjian's website: http://candle.pha.pa.us/
Professional support: http://www.greatbridge.com/
TPC-C-Benchmark: http://www.tpc.org
Apache today on the benchmark: http://apachetoday.com/news_story.php3
Tim Perdue on MySQL and Postgres: http://www.phpbuilder.com/columns/tim20000705.php3

Transactions: The ACID model

Transactions are required to meet the ACID characteristic requirements. ACID stands for Atomicity, Consistency, Isolation and Durability. This means: All operations in a transaction must either be executed in full or not at all, must not create any inconsistent database condition and not be affected by other transactions. The changes resulting from the successful execution of a transaction must also be durably stored in the database.

Second Candidate: MySQL

MySQL is a relational database management system (DBMS) under the GPL. Its origin lies in the database tool UNIREG developed in 1979 by Michael Widenius for the Swedish firm TcX. In 1994 MySQL arose from this project, and this is still being developed today. MySQL now comes in versions for Linux, FreeBSD, Solaris and other commercial Unix variants, the 32bit versions of Windows and also OS/2. The official pronunciation is My-es-cue-ell. People who say "My Sequel" are not, however, liable to be prosecuted.

MySQL has been available for a long time for private users under Linux free of charge and has developed into a near standard for web databases. Since last year, MySQL has been under the GPL. This means ambitious users now have the option of adding the components that are still lacking. But please note that the older versions of MySQL are still subject to the old and highly complicated licence conditions.

There is still the option of acquiring current versions of MySQL under a commercial licence, when the components of the database software are intended to be integrated into commercial proprietary products. But there is a legal problem when it comes to what the firm MySQL AB as holder of the copyright understands by the term Linking: For this, it is enough, in fact, if a proprietary, commercially available product relies on MySQL in order to function.

MySQL - Installation

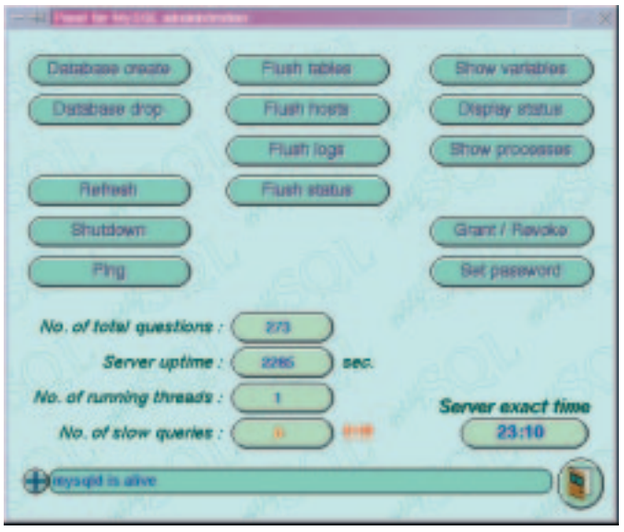
MySQL is part of all common Linux distributions. If you want, the installation programs will also set up the database system. There are binaries as tarballs for all common Unix systems and as RPMs for Linux on Intel architecture. The installation is described in great detail for all architectures in the documentation.

MySQL – Standard functions

MySQL offers conformity to Standard Entry Level SQL92. Newer MySQL versions support transactions, but user views and nested queries are not provided by MySQL, so these have to be emulated. Take the following example:

```
Person(Person ID, Name, Forename, Address)
Telephone call (Telephone call ID, Person I?
D, Telephone number)
```

There are two tables, *Person* and *Telephone call* with the corresponding attributes. With nested queries, it would be possible without further ado to



MySQLGUI is the official GUI front end of MySQL.

find out the *Name* and *Forename* as well as the *Address* of the person who dialled a certain *Telephone number*. A corresponding query looks like this:

```
SELECT Person.Name, Person.Forename,
         Person.Address FROM Person
            WHERE Person.Person_ID IN
            (
            SELECT Telephone call.Person I
D FROM Telephone call
                   WHERE Telephone call.Tel
ephone number="0207"
             )
```

This is not possible in MySQL and often has to be resolved on the application side. To do this interim results should be stored and evaluated. It becomes problematic when the database changes during this process. Fortunately nested queries can be emulated with MySQL. In the documentation it says in this respect that queries in the form

```
(1) SELECT * FROM table1
           WHERE id IN (SELECT id FROM table2);
(2) SELECT * FROM table1
           WHERE id NOT IN (SELECT id FROM ta
ble2);
```

are not supported. The same effect can however be achieved as follows:

```
(1) SELECT table1.* FROM table1,table2
           WHERE table1.id=table2.id;
(2) SELECT table1.* FROM table1
           LEFT JOIN table2 ON table1.id=ta
ble2.id
           WHERE table2.id IS NULL;
```

MySQL – Data integrity/Transactions

The guarantee of data integrity when using MySQL is incumbent on the application programmer. He must design his database application with sufficient knowledge of the database schema in such a way that the data's integrity is not jeopardised. In our Person-Telephone call database he can first ascertain with a query in the form

Info on MySQL

MySQL Online-Reference: http://www.mysql.com/documentation/mysql/bychapter/
MySQL-Homepage: http://www.mysql.com
MySQL PHP-API: http://www.php.net/manual/en/ref.mysql.php
mysql.pas: http://www.fichtner.net/delphi/mysql.delphi.phtml
Python Interface to MySQL: http://dustman.net/andy/python/MySQLdb/
MySQL++: http://www.mysql.com/downloads/api-mysql++.html
MyODBC: http://www.mysql.com/downloads/api-myodbc.html
Perl DBI: http://www.mysql.com/downloads/api-dbi.html
MM MySQL: http://mmmysql.sourceforge.net/
MyTLC: http://www.mytcl.cx/
Delphi-Interface: http://www.productivity.org/projects/mysql/
Optimization: http://www.mysql.com/news/article-26.html

```
SELECT * FROM Person WHERE Person_ID="100";
```

that a person with the *Person_ID 100* exists. In the next step a telephone call which was conducted by the person with the *Person_ID 100* can then be entered in the telephone call table. If no such person exists, a corresponding error message can be generated. The key word *FOREIGN KEY* is certainly accepted by MySQL - but no error message is given - it is completely ignored.

For transactions the current versions of MySQL use secure-transaction tables from the Berkeley Database (DBD). This type of table must be explicitly stated when it is created. Later conversions by means of *ALTER TABLE* are also possible; for syntax and details, please see the documentation.

MySQL - Expanded functionality

MySQL offers lots of syntactical goodies to make life easier for the application developer. For example REPLACE instead of DELETE and INSERT, the SHOW statement, to get information about tables. Even accesses to tables from other databases than the current one are possible. A complete list of all expansions is included in the documentation. Anyone who relies on portability should only use this function with the utmost caution, as it does not work in other databases, or works differently than it is meant to.

Referential Integrity

With so-called foreign keys it is possible to guarantee data integrity during the execution of database operations. Thus an address can only be added to a database if there is already additional information on the corresponding person in the database. Foreign key constraints create an error message, if they are violated.

Triggers and stored procedures are not currently implemented, but the latter should be possible in one of the future versions.

One clear plus compared with Postgres is the replicability of databases. A slave database can be created on the basis of a master database, and queries can be passed on to the former if necessary. The master creates a binary logfile for this, which the slave updates during a connect. This procedure increases both performance and failure security. Write-accesses to the database must, however, always be done to the master.

MySQL has no user views. But these can easily be emulated by the database application. After all, the application program has to read the data out of the database anyway, in order then to display it. So there may be another pre-processing step between reading and displaying, to filter out the relevant data for the user.

MySQL - interfaces and clients

As a rule, no halfway-normal user wants to type in SQL instructions purely by hand. (Exceptions prove the rule.) Also, for example, a biologist cannot be expected to learn SQL, in order to insert the findings from his experiments in a database. After all, a database application (see box "Glossary") has to be programmed. For the user to be able to manipulate the data in the database via this application program, appropriate interfaces (Application Programming Interfaces, APIs) have to be provided. In MySQL, these are the following:

C API
JDBC
Pascal API
PHP API
TCL API
C++ API
ODBC
Perl DBI API
Python API
Delphi API

For interactive web content, sites ranging from the private to the serious commercial have proven the value of so-called LAMP projects. LAMP stands for: Linux, Apache, MySQL, PHP.

The web front end PHP MyAdmin has been making life easier for MySQL administrators for some time now. Gnome-based and Gtk-based front ends called Glider and GtkSQL respectively can be obtained from Sourceforge. MySQLGUI can be downloaded direct from the MySQL website. This tool needs the FLTK toolkit and serves in the monitoring and administration of MySQL databases. The KSQL Project, which has set itself the goal of creating a database-independent front end for the KDE desktop, is already well advanced. MySQL functions are at present those best supported.

Conclusion

Both candidates are reliable and stable open source databases, which are especially but not exclusively suitable as back ends for dynamic websites. Postgres does, however, offer considerably more options than MySQL, which does not meet all the standard requirements for a relational data management system. Postgres on the other hand, with object-relational features such as inheritance and free datatype definitions, actually goes beyond them. So it is better suited than MySQL for the compilation of complex data models. Postgres development is now in full swing again since Version 7.0.x and many administrators are currently investigating the possibility of migrating to this system.

MySQL does have the plus points of a very large user community, which has developed an untold number of web applications, especially in connection with PHP. In many cases all one needs is a simple and relatively static data structure and redundancies can also be taken in stride. Then, in some circumstances, MySQL even offers performance advantages, which can even be expanded further to various hosts, if there is a great deal of traffic, with the new option of replicability.

Anyone who develops only free software is on the safe side in terms of licence law with both databases. Because of the generous BSD Licence, nor need one worry about commercial applications based on Postgres. But anyone wanting to sell proprietary products with MySQL should look very carefully at the licence conditions on the MySQL website.

Earlier versions of Postgres were often unstable and quite often there were even data losses. The Postgres developer team has invested a great deal of time in regression tests, which guarantee high stability and data security. Releases now only occur after longer Beta phases. The new versions of Postgres have become considerably more effective (see box "Benchmarks").

There are not known to be any serious stability problems with MySQL. With the new feature of replicability, the system has taken another major step in the direction of increased failure safety. Especially in conjunction with PHP, MySQL achieves good performance values in web applications (see box "Benchmarks"). A paper by the founder of MySQL, Michael Widenius on performance optimisation is recommended in this respect

Swings and roundabouts

Where can MySQL and Postgres be put to best use? Both are excellent choices for two large domains: These are firstly program development and secondly Internet databases. Commercial database management systems such as Oracle or DB2 may be more powerful, but are also more expensive. So as a rule it makes no sense to buy such a product before any applications have been developed for it.

This is a great opportunity for free databases. The complete development of database applications can be done on both MySQL and on Postgres. This is made possible for example in PHP applications by transferring all queries in SQL. With skilful programming, in the event of a later porting onto a large database management system, it is only the program-specific functions that are replaced by the corresponding functions of this DBMS.

The second domain of application is the already mentioned Internet databases. On the Internet, good performance is what matters most. Also, write operations on databases on the Internet are fairly rare. This is where MySQL's missing functionality is actually helpful: The less checking that needs to be done, the faster a DBMS is as a rule. Should the respective Internet application become fairly complex, though, and for example also realise insertions in several tables, then Postgres is definitely the better choice. At this point, the disadvantages of the missing functions tip the balance.

But at this point, again, (see box "Benchmarks") it has to be said that all these performance measurements have their own tricks. All too often, one finds that each system is the best according to one special benchmark. ■

The authors

Hagen Höpfner is an IT student at the Otto-von-Guericke University in Magdeburg. In his spare time he is an enthusiastic father and plays guitar in a rock band. (http://www.gutefrage.de) Dirk Gomez has been developing database applications for almost ten years and works at ArsDigita, a supplier of web-based community systems.

Requirements for a DBMS (Codd's Rules)

A certain basic functionality is demanded of databases, which goes beyond simply storing and retrieving data. Over time, a catalogue of functions has come to set the standard. These nine Codd's Rules are:

- *Integration: Storage of redundant data leads inevitably to conflicts. So one of the tasks of a DBMS is to avoid redundancies.*
- *Operations: A DBMS must enable the storage, searching and changing of data.*
- *Catalogue: The catalogue stores all the information on the structure of the data.*
- *User Views: These allow data to be filtered. The user of a view is shown only the data necessary to him. Another user can be shown the same data in another context in a different way.*
- *Consistency monitoring: This is intended to make it as hard as possible for the user to violate the integrity of the database.*
- *Access control: A DBMS must ensure that only authorised users gain access to the database.*
- *Transactions: A transaction is combination of various database operations, which must comply with the ACID characteristics (see box "ACID").*
- *Synchronisation: In multi-user operation the DBMS must synchronise concurrent database operations.*
- *Data protection: in this connection, data protection means that a DBMS must enable the regeneration of the database – for example, after a system failure.*