PASCAL

What a Database and a DBMS Are

A while ago I posted the following quote of the week at: http://www.dbdebunk.com.

Question: We are a not-for-profit organization that is looking for a suitable Linux-based database to suit our needs. MySQL is looking good but I haven't seen the term "relational" used with MySQL. Is it because it is really the foundation for any database that could in fact be relational or because it is not natively supported? Also would you recommend any existing open source MySQL databases that would suit the needs of CRM, etcetera?

Answer: Any database that allows you to establish a relation between different pieces of data is a relational database. MySQL is a relational database, in that it allows tables to be joined together and also supports the concept of foreign keys. As for CRM, it depends on whether you are planning to develop your own or acquire a CRM solution. MySQL is well suited to serve as a back end for a CRM solution, one MySQL partner providing CRM solutions based on MySQL is SugarCRM.

Mike Hillyer, who answered the question posted the following on his blog.

The only thing I hate more than being wrong is being called on it. So what happened that spawned this post? Well, as some of you may know, I am the MySQL 'Expert' at www.searchdatabase.com, you can see a list of the answers I have provided so far here. Well, on a fateful afternoon in June I gave a wrong answer. Not only that, but someone alerted www.dbdebunk.com and apparently I became the dumbass of the week for October 2004: Are you missing what the error was like I did when I posted this? Let's look at a quote from www.whatis.com:

"A relational database is a set of tables containing data fitted into predefined categories. Each table (which is sometimes called a relation) contains one or more data categories in columns. Each row contains a unique instance of data for the categories defined by the columns."

So yes, MySQL could be considered a relational database, but not for the reason I stated. MySQL is a relational database because it has data organized into tables, with the tables organized into rows and columns. In fact it has nothing to do with the relations you create using SQL. Well I feel a little dumb, and a bit less like an 'Expert', but I guess you live and learn. On the bright side the quote archives at dbdebunk have some pretty famous names quotes, so maybe I am in good company.

Here we have a purported expert who, when alerted that he gave a wrong answer about the core concept of his field, resorts to Internet sources such as www.whatis.com for guidance. That is like a heart specialist giving a wrong answer to the question "What is a heart?", and going to find the correct one on the Internet. Would you want him to treat you?

1. The question was really about a *DBMS*, not a database. I know many dismiss such corrections as "pedantic" and unnecessary, but (a) when it comes to newbies, it is important to make them aware of the distinction (b) given the current amount of ignorance in the industry, I am not willing to assume that there are no practitioners who confuse the two, or at least are unaware of the distinction.

2. For a product to be relational, it must be a *DBMS* first. There are postings at DATABASE DEBUNKINGS which make it quite clear, later improvements notwithstanding, that MySQL is a *file manager*; see, for example, *MySQL and Innobase: Are They DBMSs, Let Alone Relational?* (www.dbdebunk.com/page/page/622534.htm)

3. The answer given by Hillyer is so obviously wrong on its face, that there is no point to explain why here (although I am sure there are still many who, like Hillyer, still don't know). What is interesting to mention is the following set of two weekly quotes from the MySQL documentation, and by a MySQL proponent (author?):

"Reasons NOT to Use Foreign Keys constraints: There are so many problems with foreign key constraints that we don't know where to start:

- Foreign key constraints make life very complicated, because the foreign key definitions must be stored in a database and implementing them would destroy the whole "nice approach" of using files that can be moved, copied, and removed.
- The speed impact is terrible for INSERT and UPDATE statements, and in this case almost all FOREIGN KEY constraint checks are useless because you usually insert records in the right tables in the right order, anyway.
- There is also a need to hold locks on many more tables when updating one table, because the side effects can cascade through the entire database. It's MUCH faster to delete records from one table first and subsequently delete them from the other tables.

- You can no longer restore a table by doing a full delete from the table and then restoring all records (from a new source or from a backup).
- If you use foreign key constraints you can't dump and restore tables unless you do so in a very specific order.
- It's very easy to do "allowed" circular definitions that make the tables impossible to re-create each table with a single create statement, even if the definition works and is usable.
- It's very easy to overlook FOREIGN KEY ... ON DELETE rules when one codes an application. It's not unusual that one loses a lot of important information just because a wrong or misused ON DELETE rule.
  The only nice aspect of FOREIGN KEY is that it gives ODBC and some other client programs the ability to see how a table is connected and to use this to show connection diagrams and to help in building applications."

"The FOREIGN KEY syntax in MySQL exists only for compatibility with other SQL vendors' CREATE TABLE commands; it doesn't do anything. The FOREIGN KEY syntax without ON DELETE ... is mostly used for documentation purposes. Some ODBC applications may use this to produce automatic WHERE clauses, but this is usually easy to override. FOREIGN KEY is sometimes used as a constraint check, but this check is unnecessary in practice if rows are inserted into the tables in the right order. MySQL only supports these clauses because some applications require them to exist (regardless of whether or not they work). In MySQL, you can work around the problem of ON DELETE ... not being implemented by adding the appropriate DELETE statement to an application when you delete records from a table that has a foreign key. In practice this is as quick (in some cases quicker) and much more portable than using foreign keys."

The chance of anybody with this level of understanding coming up with a DBMS, let alone a relational one, is nil. The whatis answer is, of course, bunk. About the only sentence that makes any sense is the one about tables, and even it is backwards: it's not that the tables are "sometimes called relations". They must be tables which obey a special discipline, which makes them *faithful representations of mathematical relations*. Without that discipline – and a DBMS explicitly designed to exploit that discipline – that is, application of predicate logic and set mathematics to the integrity and manipulation of the data-there is no relational fidelity, and the practical benefits derived from it do not materialize.

But Hillyer accepts that "definition" as correct, because he has never bothered to educate himself on the fundamentals. And why should he bother? After all, he is regarded as an expert without it.

Based on experience, my guess is that the reaction to this article will be about my arrogance and mistreatment of Hillyer; nobody will really express concern about the misleading effect of his pronouncements on many of his readers. Indeed, instead of being concerned that so many are ignorant of the basics, Hillyer comforts himself for being "in good company", If you ever wondered how ignorance can be so profound and widespread, well, this is but one example of the responsible mechanism, and the culture in which it operates..

Here are, for the benefit of the reader, correct definitions of a database and DBMS:

- A database represents a set of axioms.
- Responses to queries represent theorems.
- The process of deriving theorems from axioms is a proof, which is made by manipulating symbols according to agreed mathematical rules.
- The theorems are true if and only if the axioms are true and the proofs are mathematically correct.

- A (properly designed) RDBMS is a deductive logic system: it derives theorems from axioms.
- Theorems are guaranteed to be true if and only if the DBMS ensures that (a) axioms are true and (b) the manipulation (proofs) are mathematically correct.
- It is the function of the integrity component of a DBMS to ensure that the axioms are true.
- It is the function of the manipulation component of the DBMS to ensure that the theorems are true.

By true we mean, of course, consistent with the integrity constraints in effect, which represent business rules in the database.

Fabian Pascal
Fabian Pascal is onafhankelijk IT-analist, consultant en auteur gespecialiseerd in data management. Zie ook zijn website www.dbdebunk.com