

Re: Updating the SQL key value

Source: <http://coding.derkeiler.com/Archive/PHP/comp.lang.php/2007-04/msg00024.html>

- *From:* Jerry Stuckle <jstucklex@xxxxxxxxxxxxxx>
 - *Date:* Sun, 01 Apr 2007 09:50:41 -0400
-

Toby A Inkster wrote:

Jerry Stuckle wrote:

Toby A Inkster wrote:

Firstly, I said char, not varchar. char columns are fixed width storage and consequently a lot faster to search than varchar columns. On a 32-bit processor, char(4) is particularly fast as it corresponds to the processor's word length.

True, but unless it's a binary column the search must be case-insensitive.

I normally use PostgreSQL, in which all strings are handles case-sensitively unless you specify otherwise.

Not all databases do this. And that still doesn't mean a char(4) will be compared with the same code as an int.

Finally, even if none of the above are true, an int column will almost always be word aligned in memory. A char field – even if it's char(4), is not always word aligned.

Perhaps not word-aligned when stored as row data, but certainly row-aligned in the index. And you'd nearly always want to index a primary key column.

Are you sure? I would highly doubt a char(4) would be word aligned in any index. What a waste of space, i.e.

Re: Updating the SQL key value

char(4) and char(1) as your index. You should have 60% more space taken up in alignment than you have in data.

Firstly, in this example, the licence code is an arbitrary string chosen by the creator of the data, so it needs to change no more often than an integer key would.

So why not just use an auto-number field. It seems you're unnecessarily creating an unnecessary bottleneck.

If you use a piece of useful data as a primary key, instead of a meaningless number, this in many cases reduces the need to perform joins between tables, resulting in simplifying queries and speeding things up. As an example, consider extracting a list of article titles plus the author's login name from our articles and users tables:

If the "useful data" means you are reducing the need to join tables, you have a vary poorly normalized database.

My schema:

```
CREATE TABLE articles (  
  article_id integer PRIMARY KEY,  
  title varchar NOT NULL,  
  body varchar NOT NULL,  
  author char(16) REFERENCES users  
);  
CREATE TABLE users (  
  login char(16) PRIMARY KEY,  
  email varchar NOT NULL  
);  
-- Insert lots of data here.  
SELECT title, author FROM articles;
```

Yours:

```
CREATE TABLE articles (  
  article_id integer PRIMARY KEY,  
  title varchar NOT NULL,  
  body varchar NOT NULL,  
  author integer REFERENCES users  
);  
CREATE TABLE users (  
  user_id integer PRIMARY KEY,
```

Re: Updating the SQL key value

```
login char(16) NOT NULL,  
email varchar NOT NULL,  
UNIQUE (login)  
);  
-- Insert lots of data here.  
SELECT a.title, u.login AS author  
FROM articles a  
INNER JOIN users u ON a.author=u.user_id;
```

The second schema has to query two tables and perform an inner join, which is slower than performing a single query to a single table.

Now – what happens if you have two authors named "John Smith"? Your design just fell apart.

Also, a persons login id may not be the same as their name. And if I'm authoring an article, I want my name on it, not my userid.

Not a design I would use.

Sure. But you don't have to use referential integrity. And no, databases do not just point to another table, as I indicated. Look at the data in your tables. You will find the char fields.

You will find the char fields if you use SQL. I'm talking about the binary data that the RDBMS actually keeps in memory while it's running. They use all kinds of tricks for speed boosts, including using pointers for primary keys instead of using the real data.

Sure, they use all kinds of speed boosts. But they still have to search the tables. And they still have to do comparisons on strings.

Sure. And in this case transactions cause their own problems. You need at least 4 calls to the database:

```
START TRANSACTION  
SELECT MAX(colId)+1 FROM mytable  
INSERT INTO mytable ...  
COMMIT
```

Significantly slower than a single call which returns an auto-numbered column. And worse yet, it will tie up the table for additional changes until the COMMIT. Very bad for concurrency.

Re: Updating the SQL key value

I'm sure autonumbering results in something fairly similar happening deep down in the RDBMS anyway.

I think you're making a bad assumption there, Toby. I suspect there is nothing like that going on "deep down in the database". At least not in any database I've worked with – which include DB2, Oracle, and SQL Server in addition to MySQL.

--

=====

Remove the "x" from my email address

Jerry Stuckle

JDS Computer Training Corp.

jstucklex@xxxxxxxxxxxxxx

=====

.