

Re: Data Normalization

Source:

<http://www.tech-archive.net/Archive/Microsoft.Public.Access.Queries/2008-10/msg01431.html>

- *From:* R Tanner <tanner.rob@xxxxxxxxxx>
 - *Date:* Fri, 17 Oct 2008 07:48:48 -0700 (PDT)
-

On Oct 17, 8:41 am, R Tanner <tanner.ro...@xxxxxxxxxx> wrote:

On Oct 17, 6:53 am, John Spencer <spen...@xxxxxxxxxx> wrote:

In Response to your question on Invoices and customers.
You came pretty close, except the INVOICES table would contain a CustomerID foreign key. The Customer table would not have an invoice id.

One Customer has (or could have) many Invoices. The many side gets the Foreign key.

In response to your question on Primary keys
The Primary key should meet the following constraints:
A) It must be UNIQUE. No duplications
B) It should be stable (that is it should not change over time). There are ways to handle changes in the value but if possible (in my opinion) they should be avoided.
C) It must be available for every record.
Social Security Numbers (SSN) for people in the US are problematic. New Borns don't have them immediately, visitors from overseas don't have them, sometimes people won't share them, etc. It depends on the purpose of the database whether an SSN would be a good candidate for a primary key.

Names of people are not unique and are not stable (marriage, divorce, etc). Names of the States in the United States are stable and probably will not change. Abbreviations for names of the States are stable and

Re: Data Normalization

do not change frequently (To Date: once in my lifetime).

So there is nothing wrong with using a Natural Key (some item or items of data in each record) or with using an artificial key (autonumber or other unique value that has no relation to the contents of the record).

This subject Natural vs Artificial can become a holy war with some individuals. I use both depending on the table and situation.

=====
John Spencer
Access MVP 2002–2005, 2007
Center for Health Program Development and Management
University of Maryland Baltimore County
=====

R Tanner wrote:

On Oct 16, 4:57 pm, John W. Vinson
<jvinson@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

On Thu, 16 Oct 2008 14:51:30 –0700
(PDT), R Tanner <tanner.ro...@xxxxxxxx>
wrote:

I understand what the purpose and value of data normalization is, but I just have one thing I am confused about. If you split up a table, then it will create two tables. The child table will have a lookup column to the parent table, which will have whatever value is applicable that relates that given record to it's parent. What I don't understand is how that is more efficient. In fact it seems to duplicate the data if

Re: Data Normalization

anything...

It duplicates *one field*. Your related table may have many fields!

Just for example, consider a People table with a PersonID, related one to many to an Address table. Each person may have several addresses; each record of the address table would have fields like AddressNo, Direction, Street, Suffix (i.e. St., Ave., Blvd.), Postcode, City, State and Country. These fields don't need to occur in the parent table and need not be duplicated.

--

John W. Vinson [MVP]

Also in reply to your post about the primary and foreign keys

--

When I use the table analyzer and I split a table, the lookup column to the parent table will have the actual values that are in the parent table. Should this be the case, or would it not take up less memory in the database to have the lookup column look up the primary key in the parent table, which would be an autonumber...-- Hide quoted text --

-- Show quoted text --

So let me pose this question to you John. I am trying to split one table into about 4 other tables. These are the steps I am taking:

1. Run a Total's Query on the data I want to have in the parent table.
2. Run a Make Table Query
3. Identify my primary key in the new Parent table. (the data will not change, but more may be added, so I have a boolean helper column

Re: Data Normalization

called current I use in my SQL statement)

4. Create the relationship between the child table and my new parent table. (I think maybe I am doing something wrong here. Two things I understand about creating these relationships:

1. The datatype must be the same
2. The primary table can only maintain a relationship with the child table through it's primary key (Because of this, I identified my actual data as the primary key and created the relationship between it and the actual data in the child table. What I have been trying to do is figure out how to build the SQL statement that will create the issue ID column I need in my child table that contains the foreign key)

Does anything sound amiss in this scenario to you? Thank you for emphasizing that the foreign key is only on the many side. I think I vaguely recognized that, but you helped to clear it up.– Hide quoted text –

– Show quoted text –

Also, in the following SQL statement, what is the purpose of the xyz?

```
SELECT [ID] AS xyz_ID_xyz, [ID] AS xyz_DispExpr_xyz, [Issue], [If Other] FROM Issues ORDER BY [Issue], [If Other];
```

.