Join Tables and Primary Keys

Posted At : May 24, 2007 1:44 PM | Posted By : Steve
Related Categories: SQL

I have found that many people set up the primary keys for join tables in a manner that I find undesirable. This leaves open the possibility of some sneaky bugs in their programs.

I'll start by covering what I mean by a join table (as opposed to what I will call a data table).

Most basic data is stored in data tables. For example a "users" table is a data table, as is a "groups" table. Each row in a "users" table represents one user, as each row in a "groups" table would represent a group.

It could be, however, that each user could be in more than one group and each group could have more than one user as a member. In order to keep the database normalized, another table should represent these relationships ("users2groups").

I wouldn't classify this as a data table because each row in this table represents a relationship between two things, rather than a thing itself. So, the "users2groups" table is a join table.

I have frequently seen a structure like this for join tables:

users2groups:
user2group_id (primary key)
user_id
group_id

The thinking here, I imagine, is that we know that each table needs a primary key so we add one to the table. Herein lies the problem. The value of "user2group_id" represents a relationship, not data.

You won't ever look up a row using the "user2group_id" because the row has no value in isolation. Instead, you will look up rows by the user_id or the group_id.

The real problem, however, is that this allows duplication. See this sample data:

user2group_id	user_id	group_id
1	2	3
2	2	3

Using a user2group_id primary key allows this duplication, which has no meaning.

At first this may seem like the same problem as two rows of matching data in a data table, but it isn't. First of all, a duplicate user row means something - that two users have identical data. More than that, it is relatively easy to detect.

A duplicate row in a join table has no meaning, but it can cause problems that are difficult to debug.

The solution to this problem is simple, drop the extraneous "user2group_id" and make your primary key a compound primary key consisting of both "user_id" and "group_id".

users2groups:
user_id (primary key)
group_id (primary key)

This will ensure that you cannot have one relationship represented by more than one row.

You should even be able to make this change without causing any problems for your existing application. Unless, of course, you already have a relationship duplicated in the table.

Next time you make a join table, I suggest using a compound primary key. The structure is simpler and it could help to eliminate some bugs.