Copy Legacy Data with SQL

Posted At : February 1, 2008 2:00 PM | Posted By : Steve
Related Categories: SQL

In talking to a friend of mine yesterday, I realized that a great many programmers are not aware of one of the nicest features of SQL, the ability to insert the results of a query into a table. This is really handy for copying data from an old format to a newer format or if you have a need to denormalize part of a database.

Personally, I find it especially handy when I am upgrading legacy code. I often upgrade code with an unmaintainable structure. In one recent case, the table structure had such gems as varchar primary keys (the values of which were typed in by the user), comma-delimited lists in the place of many-to-many joins, and data that should be in multiple columns instead stored in one column with textual delimiters.

So, I wrote some code to copy data from old table structure to a new one, using the handy INSERT INTO SELECT feature of SQL:

```
INSERT INTO newtable (
field1,
field2,
field3,
oldpkval
)
SELECT
fielda AS field1,
fieldb AS field2,
fieldc aS field3,
pkfield
)
FROM oldtable
```

I aliased my column names the sake for clarity, but that isn't required. It is required, however, that the SELECT statement return the same number of columns, and in the same order, as the INSERT statement.

Whenever I copy legacy data, I make sure to have a column in my new table to store the value of the primary key from the legacy data. This will allow me to refer to the old data if I missed something in the transfer, but it will also allow me to ensure that the code can be run multiple times without creating duplicate rows. In order to achieve that, I do the following:

```
INSERT INTO newtable (
field1,
field2,
field3,
oldpkval
)
SELECT
fielda AS field1,
fieldb AS field2,
fieldc aS field3,
pkfield
)
FROM oldtable
```

```
WHERE NOT EXISTS (
SELECT newpkfield
FROM newtable
WHERE oldpkval = newtable.pkfield
)
```

With this code in place, the SELECT statement won't return any rows that have already been copied to the new table.

I have found this helpful any time I need to copy data into a table from one or more other tables.

It is important to note, for some situations, that the entire SELECT statement runs before any rows are inserted. This may seem obvious and won't usually matter, but if you write any complicated SELECT statements it may become important to note.

In writing statements like this, I generally write and test the SELECT statement several times before I finally add the INSERT clause above.