# Prevent duplicate Inserts

Posted At : March 16, 2005 1:46 PM | Posted By : Steve
Related Categories: SQL

I frequently need to insert a record that relates to tables to each other (or relates one table to external data) and I cannot have a duplicate record. Here is my method for doing this in case it helps anyone and as a convenient reference for myself.

Let's suppose that you have three tables: Users, Classes and Students. Classes has a primary key of ClassID (int). Users has a primary key of UserID (int). Students has a two-column primary key of ClassID and UserID which relate to the corresponding columns in Classes and Users respectively.

In order to add a user as a student in a class, you could query for the student and then run an insert query or you could accomplish both in one query using SELECT INTO. The following are two equivalent SQL statements to add a student record only if it doesn't exist already:

```
INSERT INTO Students
SELECT  #Val(arguments.ClassID)# AS ClassID, UserID
FROM  Users
WHERE  UserID = #Val(arguments.UserID)#
AND  NOT EXISTS (
   SELECT UserID
   FROM Students
   WHERE ClassID = #Val(arguments.ClassID)#
    AND UserID = #Val(arguments.UserID)#
  )
```

OR

```
INSERT INTO Students
SELECT  ClassID,  #Val(arguments.UserID)# AS UserID
FROM  Classes
WHERE  ClassID = #Val(arguments.ClassID)#
AND  NOT EXISTS (
   SELECT UserID
   FROM Students
   WHERE ClassID = #Val(arguments.ClassID)#
    AND UserID = #Val(arguments.UserID)#
  )
```

Both of these statements use "INSERT INTO SELECT" in conjunction with a "NOT EXISTS" subquery.

If you use INSERT INTO [TableName] followed by a SELECT statement (instead of the usual field list and value list), then the result of the SELECT statement is inserted into the table (making it possible to insert multiple rows with one select statement). Keep in mind, however, that if you do not specify a column list then the SELECT statement must return the same number of columns as the table to which data is being inserted and in the same order.

The "NOT EXISTS" subquery ensures that the SELECT statement only returns a record if the subquery doesn't. The subquery matches the result of the SELECT statement (and therefore the values being inserted). Therefore, the SELECT statement will return no rows (and therefore no rows will be inserted) if the record already exists.

This technique doesn't throw an error if the record already exists, but nor does it require a separate query (and therefore a transaction block).

Good luck!