RECENT ENHANCEMENTS AND EXTENSIONS TO SQL

Other Enhancements

In addition to the enhancements to windowed tables described previously, the CREATE TABLE command was enhanced by the expansion of CREATE TABLE LIKE options. CREATE TABLE LIKE allows one to create a new table that is similar to an existing table, but in SQL:1999, information such as default values, expressions used to generate a calculated column, and so forth, could not be copied to the new table. In SQL:2008, a general syntax of CREATE TABLE LIKE . . . INCLUDING was approved. INCLUDING COLUMN DEFAULTS, for example, will pick up any default values defined in the original CREATE TABLE command and transfer it to the new table by using CREATE TABLE LIKE . . . INCLUDING. It should be noted that this command creates a table that seems similar to a materialized view. However, tables created using CREATE TABLE LIKE are independent of the table that was copied. Once the table is populated, it will not be automatically updated if the original table is updated.

An additional approach to updating a table was enabled in SQL:2008 with the new MERGE command. In a transactional database, it is an everyday need to be able to add new orders, new customers, new inventory, and so forth to existing order, customer, and inventory tables. If changes that require updating information about customers and adding new customers are stored in a transaction table, to be added to the base customer table at the end of the business day, adding a new customer used to require an INSERT command, and changing information about an existing customer used to require an UPDATE command. The MERGE command allows both actions to be accomplished using only one query. Consider the following example from Pine Valley Furniture Company:

```
MERGE INTO Customer_T as Cust
    USING (SELECT CustomerID, CustomerName, CustomerAddress,
    CustomerCity, CustomerState, CustomerPostalCode
      FROM CustTrans_T)
      AS CT
    ON (Cust.CustomerID = CT.CustomerID)
WHEN MATCHED THEN UPDATE
  SET Cust.CustomerName = CT.CustomerName,
    Cust.CustomerAddress = CT.CustomerAddress,
    Cust.CustomerCity = CT.CustomerCity,
    Cust.CustomerState = CT.CustomerState,
    Cust.CustomerPostalCode = CT.CustomerPostalCode
WHEN NOT MATCHED THEN INSERT
  (CustomerID, CustomerName, CustomerAddress, CustomerCity,
  CustomerState, CustomerPostalCode)
    VALUES (CT.CustomerID, CT.CustomerName, CT.CustomerAddress,
    CT.CustomerCity, CT.CustomerState, CT.CustomerPostalCode);
```