

New JET Engine Features Enhance Access 2.0

Access 2.0's new database engine streamlines the application development process while providing improved SQL connectivity.

BY STEVE ROTI

The enhanced JET engine in Access 2.0 is a major improvement over the previous version. New features like engine-level validation, cascading updates and deletes, and subqueries will make application development easier by removing some of the coding burden from Access Basic. Although the JET engine 2.0 still trails most database servers in terms of support for SQL, it is near the top of the heap among PC DBMSs and the enhancements will make Access more attractive as a front end in client/server systems.

The JET database engine is the under-the-hood, behind-the-scenes, out-of-plain-view portion of Microsoft Access. Of course, the JET database engine version 1.1 is also included in Visual Basic 3.0, so you can also think of Visual Basic applications that use Access databases as being systems built upon the JET engine.

If you have used Access in the past you have undoubtedly heard of the JET engine, but if you looked for more information about it in the manuals you probably didn't find much. So what exactly is the JET engine, and why should Visual Basic developers care about it?

While the role of the JET engine within

Steve Roti is the owner of Olympic Software, a database consulting firm in Portland, Oregon. He develops SQL database applications on Windows, Unix, NetWare, VMS, and Macintosh. Steve is also a contributing editor of DBMS magazine. He can be reached via CompuServe at 70323,3614.

Access 2.0 is clear, how does it compare to other SQL DBMS engines? In order to provide some perspective on this upgrade, it is useful to step back and see where Access fits in the market.

First, it's important to note that Access is a file-server-technology database management system (DBMS), not a database server. That means it has more in common with PC products such as Paradox and FoxPro than high-end server products such as Oracle, SQL Server, and SQLBase. File-server DBMSs are usually easy to install and administer, but they can't support as many users or manage as large a database as servers can. In addition to acting as a PC DBMS, Access also fits into the client/server equation by acting as a client to any SQL database server that has an ODBC driver available.

JET engine data can also be used by Visual Basic 3.0 through the use of a Compatibility Layer. The Compatibility Layer (VBCL) between the JET engine and Visual Basic 3.0 is discussed in detail elsewhere in this issue (see Andrew Brust's Database Design column, "Compatibility Layer Gives VB Access to Jet 2.0," p. 63), but it is also worth mentioning here that the VBCL is largely an artifact of the Access file-server-technology design. Database-server technology separates programs into frontends (client applications) and back ends (servers), which serves to isolate applications from changes in database file format. If a vendor decides to enhance the file format, all they have to do is ship a new version of the server and your applications continue to run without need for something like the VBCL. Not so with file server DBMSs, where the application and the engine are tied more closely together and changes to the

database file format may break the application. That's why the Compatibility Layer is needed. Then there is the question of how Access SQL stacks up against industry standards and other SQL products. Microsoft claims that Access 2.0 SQL is compliant with ANSI SQL-89 Level 1. This is the lowest level in the SQL-89 standard, and is off the bottom of the chart on the newer SQL-92 standard. Most other SQL DBMSs are at SQL-92 Entry SQL level, so it is safe to say that Access SQL supports fewer features than most.

For example, there are no data control language (DCL) commands in Access SQL for controlling security. The standard SQL DCL commands are grant and revoke. Instead, permissions must be granted to users in Access Basic code. Still, this is an improvement over Access 1.1 when security was strictly a user interface feature and couldn't be set through code at all.

Another example is validation of data in the engine. Although the new engine-level validation in the JET engine 2.0 is a valuable feature for database designers and developers, it doesn't begin to compare to the power of triggers in the high-end servers. Engine-level validation allows simple nonprocedural rules written as expressions, whereas triggers in SQL Server and Oracle allow complex procedural code for validation purposes.

Finally, the SQL documentation for the JET engine 2.0 is weak. Access SQL is documented primarily in the online Help system, which contains plenty of good examples but is hard to use as a reference. There is one relevant chapter in the *User's Guide on Advanced Queries* and one in the Advanced Topics manual on *Developing Client/Server Applications*. More printed

documentation on Access SQL would be useful, either from Microsoft or from a third party.

That introduces the usefulness of the JET engine, but it does not tell you much about what it looks like or how it does its job. In fact, the JET engine has no user interface, so it is best explained in terms of how it communicates with the other software components. The two programmatic ways of talking to the JET engine are through Access Basic and through Access SQL. Since there is a companion article that covers Access Basic in detail, this article will focus on SQL and refer to Basic only to point out features not accessible through SQL.

Like other SQL database engines, the JET engine accepts queries (usually select, insert, update, and delete), performs the requested action on the database, and returns the appropriate data or status information. VB programmers build queries with the Execute method or the RecordSource property of the data control. Access users build queries with the Query window or via Basic code.

The good news is that the Access 2.0 JET database engine understands more SQL commands than previous versions. The bad news is that the 2.0 JET engine (size 990K) is about 280K larger than its predecessor, but that's the price of new features. Let's look at what's new and improved in the JET engine.

ENGINE-LEVEL VALIDATION

Engine-level validation lets the table designer specify rules for the data that will reside in the table. These rules are always enforced, no matter how data gets into the

table. Data can be entered through a form or datasheet, inserted using a query or Access Basic code, or imported, but the new engine-level validation will ensure that it always meets the rules.

There are two types of validation rules available in the JET engine 2.0: field and record. Field validation rules are set in the Field Properties section of the Table Design window.

An example rule for a field named Order Date might be:

```
>=#1/1/94#
```

This would ensure that no orders could be entered prior to 1994. Record-level validation rules are set in the Table Properties window and are used for more complex rules involving more than one field. Here's an example rule to ensure that orders are scheduled to ship within 30 days:

```
[Required Date]<=[Order Date]+30
```

When you create a validation rule, you can also create the error message text that appears when the rule is violated. Note that validation rules and text can also be set through Access Basic code using the ValidationRule and ValidationText properties for Field objects, Recordset objects, and TableDef objects.

The new engine-level validation rules must be written as nonprocedural expressions, so their scope is somewhat limited. If you want to perform more complex data validations you can always choose to use a macro with a form, although you lose the universality of engine-level validation when you do that.

DATA DEFINITION QUERIES

To put data definition queries in context with other types of queries, it is useful to know how SQL commands are categorized. SQL commands are grouped into three categories: data manipulation language (DML), data definition language (DDL), and data control language (DCL). DML commands are the ones that change or retrieve data in the database (insert, update, delete, and select). DDL commands are used to build the structure of the database tables and indexes (create, alter, and drop). DCL commands deal with data access privileges (grant and revoke).

SQL data definition queries are new in the JET engine 2.0. Previously, tables and indexes had to be created through the Access user interface, and that option remains available in Access 2.0. But the use of DDL commands to create tables and indexes is standard in other SQL DBMSs, so database designers may now choose to use that method rather than the user interface method.

DDL queries are found on the SQL Specific submenu on the Access 2.0 Query menu along with Union and Pass-Through queries, which will be discussed later in this article. The Query menu is only available when designing a query. Creating a new table with a DDL query is easy:

```
CREATE TABLE orders (
order_number CHAR(10),
order_date DATE,
requested_date DATE);
```

In addition to simply naming the table and declaring the names, datatypes, and sizes of its columns, you can also define constraints such as primary and foreign keys. Constraints can also be added with the alter table command:

```
ALTER TABLE orders
ADD CONSTRAINT order_key PRIMARY KEY _
(order_number);
```

The alter table command can also be used to add and drop columns. To get rid of an entire table, there is a drop table command. The create and drop commands can also be used to build indexes that speed data retrieval. Note that these data definition queries are only for building tables in Access databases. In order to use SQL DDL commands to create tables in remote database servers accessed through ODBC, pass-through queries are used instead of data definition queries.

CASCADING UPDATES AND DELETES

Another new table-oriented feature is cascading update and delete operations. To understand the value of cascading updates and deletes, it is necessary to first under-

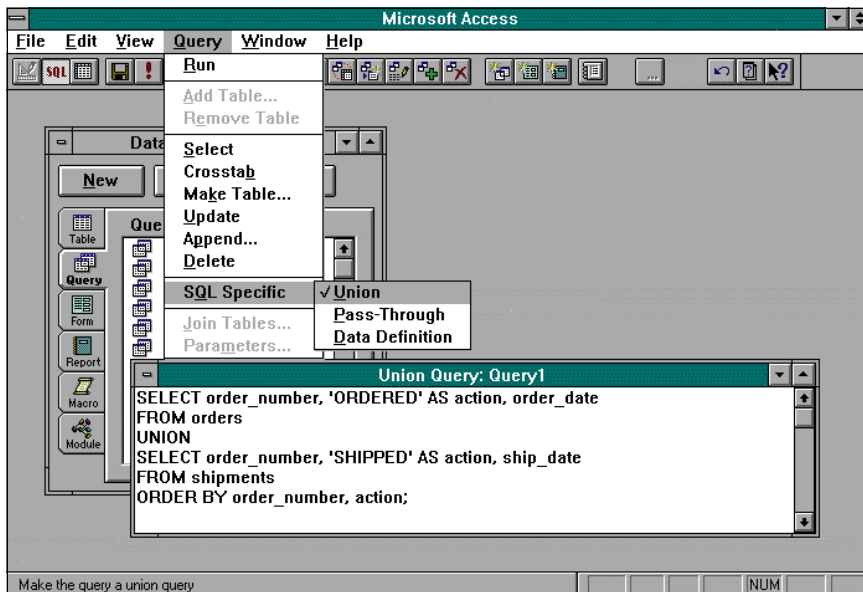


FIGURE 1 The SQL Specific Query Menu of Access 2.0 shows some of the new JET engine 2.0 features. Here a Union query is being written in the SQL View window. Other SQL Specific options are Pass-Through queries and Data Definition queries.

stand the concept of referential integrity. Access gives database designers the choice of whether to enforce referential integrity when a relationship is created between two tables. An example of the value of enforcing referential integrity would be to ensure that order numbers inserted into an order line table correspond to valid order numbers stored in an order table. The order number is called a primary key in the order table and a foreign key in the order line table. There is a one-to-many relationship between the order table and the order lines table.

If referential integrity is enforced for a relationship, then the designer can also choose whether to cascade updates and deletes. The default is to restrict (or prevent) update and deletes when there are matching foreign key values in a related table, but cascading overrides the default. In the example given above, cascade update would cause a change to an order number in the order table to be propagated to all lines for that order in the order line table. Cascade delete would cause a delete of a row in the order table to also delete all associated lines in the order line table. Using the Cascade options can result in less Access Basic application code to write. Without cascading, all related tables must

be updated or deleted separately. With cascading, the same thing can be accomplished with a single command. The possibility of multiple levels of cascading that can be declared across multiple relationships is a valuable addition to the JET engine referential integrity features.

Cascade update and cascade delete are set as two separate check boxes in the Relationships dialog box. Unlike most SQL DBMSs, cascade cannot be specified in the DDL command that creates the foreign key, meaning that the new data definition queries described above can only take you so far in building an Access database; beyond that you still have to rely on the Access user interface. Furthermore, the restrict and cascade options for referential integrity are usually accompanied by the set null option in other SQL DBMSs. Set null allows the delete to proceed, but causes the foreign key values in the related table to be set to null. So although cascade update and cascade delete are useful features, they fall short of providing full declarative referential integrity support.

UNION QUERIES

Like data definition queries, Union queries are another one of the special types of SQL queries that cannot be designed in the

Access 2.0 QBE grid. Instead, they are found in the SQL Specific submenu of the Query menu (see Figure 1). Union queries are used to combine rows from two tables by appending them together vertically. For example, a union query to combine orders from one table and shipments from another table might look like this:

```
SELECT order_number, _ORDERED_ AS _
      action, order_date
FROM orders
UNION
SELECT order_number, _SHIPPED_ AS _
      action, shi p_date
FROM shipments
ORDER BY order_number, action;
```

If there were one order row and two shipment rows, the results of this query might look like this:

order_number	action	order_date
1234567890	ORDERED	1/1/94
1234567890	SHIPPED	1/21/94
1234567890	SHIPPED	1/31/94

By default, union queries eliminate duplicate rows from the results. If you want to return duplicate rows, you can use *union all* in place of *union* in the query.

SUBQUERIES

Think of subqueries as SQL select statements that are nested inside other SQL DML commands. Traditionally, subqueries have been allowed in WHERE and HAVING clauses to restrict rows affected by the query based on values in another table. More recently, the ANSI SQL-92 standard and a few SQL DBMSs have allowed subqueries in the column list of select statements. Access offers support for all of these uses of subqueries.

An example of a select statement using a subquery to find orders that have not been shipped might look like this:

```
SELECT order_number, order_date,
requested_date
FROM orders
WHERE order_number NOT IN
    (SELECT order_number
    FROM shipments);
```

The subquery SELECT order number

FROM shipments returns a list of all order numbers that have been shipped and the WHERE clause uses that list to find orders that have not yet been shipped (i.e. orders that are *not in* that list of shipped orders).

The example above is a relatively simple use of a subquery. Note that the subquery can be evaluated once for the entire main query. A more advanced use of subqueries involves correlated subqueries that are evaluated once for each row in the main query. The reason for this is that correlated subqueries are dependent upon column values in the main query and the values change from row to row. Think about a query to return information about orders that were shipped late. One solution might look like this:

```
SELECT order_number, order_date, _
requested_date
FROM orders AS o
WHERE requested_date <
    (SELECT MAX(shi p_date)
    FROM shi pments
    WHERE order_number = _
o.order_number);
```

For each row in the outer SELECT statement, the subquery calculates the latest shipping date for shipments with the same order number and returns that date for comparison with the requested date of the order. If the requested date is earlier than the latest ship date for that order, the row from the orders table is retrieved.

REMOTE INDEX JOINS

Access defines a heterogeneous join as a join between a local Access table and remote SQL database server table accessed via ODBC. The trick to getting reasonable performance in heterogeneous joins is minimizing network traffic and local processing by performing as much work as possible on the server.

Earlier versions of the JET engine requested all records in the remote table and performed the join locally. The JET engine 2.0 performs a remote index join by requesting only those records in the remote table that match a local key value. This can improve server efficiency when using Access in a client/server architecture by reducing network traffic.

Remote index joins are used only if the remote field being joined is indexed and the local table is much smaller than the remote table. For example, if the local table contains 10 records and the remote table contains 50, the JET engine will request all 50 records. However, if the local table contains 10 records and the remote table contains 1000 records, the JET engine will send 10 separate queries to the server requesting records that match the 10 individual local key values.

CONFIGURABLE BACKGROUND POPULATION

To address the issue of resource control on SQL database servers, the JET engine 2.0 has added a feature called configurable background population that allows database administrators to optimize the rate at which query results are retrieved from the server. There are two parts to this new feature: a server-based table named MSysConf and a smarter JET engine that knows where to look for the table.

On the server, entries in the MSysConf table determine the rate at which the JET engine pulls the result set back from the server, specifically the number of records in each fetch and how often a fetch is done. If the entries in the MSysConf table are set to minimize server usage, Access applications may see a performance penalty.

One limitation of the configurable background population feature is that all users are controlled by the same MSysConf table entries, so you can't specify different rates for multiple user groups.

TOP N QUERIES

Top N queries are designed to answer questions such as "Who are my top 10 customers?" In standard SQL, this is not a trivial query to write. The typical solution involves a subquery. Access SQL adds a TOP N clause to limit the number of records returned by the JET engine in a SELECT query. The result set can be limited to the top N records in the set or the top N% of the records in the set.

For example, a TOP N query to return the names of students with the highest 25 grade point averages in their class might look like this:

```
SELECT TOP 25 first_name, last_name
FROM students
WHERE graduation_year = 1994
ORDER BY grade_point_average DESC;
```

The same query modified to return the names of the students with grade point averages in the top 10% of their class might look like this:

```
SELECT TOP 10 PERCENT first_name, _
last_name
FROM students
WHERE graduation_year = 1994
ORDER BY grade_point_average DESC;
```

MANY NEW FEATURES OFFERED

Earlier, I mentioned ODBC. Let's take a brief look at it. There is a new ODBC driver for JET engine 2.0 databases that shows up in the list of ODBC <SQL Databases>. This driver shipped with Access 2.0 and is intended for Microsoft Office users who want to connect to Access 2.0 databases. For example, people using MS Query from Ex-

cel or the data lookup capabilities in Word 6 would use this new ODBC driver. Also, any other applications that use ODBC can use this driver to get at databases managed by the JET engine 2.0.

It's not just new features that appear in Access 2.0. The JET engine 2.0 also includes certain aspects of the Rushmore query algorithm technology from FoxPro, which causes many common types of queries to run much faster. The Rushmore enhancements use multiple indexes to optimize query performance. Microsoft claims up to 100-150 times improvement for queries involving longitude-latitude data, such as finding everything in a radius around a city.

SQL pass-through queries have also been integrated into the JET engine. These queries, the third type of SQL Specific query available from the Access 2.0 Query menu, are used to pass SQL statements directly to an ODBC database server such as Oracle or SQL Server. The statements are written using the SQL dialect of the server and are not interpreted by the JET engine. Pass-through queries work directly with the tables on the server rather than indirectly through the Access attach table mechanism. Pass-through queries streamline connectivity in client/server applications and are useful for taking advantage of advanced server features such as stored procedures.

The JET engine 2.0 supports five new languages (Danish, Dutch, Finnish, Japanese, and Norwegian) and nine new sort orders (Arabic, Czech, Greek, Hebrew, Hungarian, Japanese, Polish, Russian, and Turkish). The sort order of existing databases can be changed to use one of the newly supported sort orders.

All SQL DBMSs have some limitations regarding updating data through views. Some disallow view updates entirely, and others limit updates to single table views or to columns from the *one* side of a one-to-many join. The JET engine 2.0 allows updates on multitable queries, including updates to fields from both sides of the join. For example, in a query that joins data from an orders table and a shipments table, you can update data from both tables in the same command.

SQL EDITOR

The SQL View window in Access 2.0 has been improved significantly. Although this is not really a JET engine feature, it affects the ease of sending SQL commands to the JET engine and is worth mentioning here. Any SQL statement can be entered in the SQL View window, including statements that cannot be created with the Query By Example grid. The most obvious change to the SQL View window is that Access 2.0 now lets you use the menu bar and other

windows while you are editing your SQL statement.

In earlier versions of the JET engine, an update to a single column of a table on a database server would cause triggers to execute on all of the columns because the SQL command sent to the server included all columns in the SET clause. In the JET engine 2.0, updates will now affect only columns that have been changed by the user, and will not cause triggers to execute on unchanged columns. Similarly, inserts will insert only values set by the user and will not overwrite server defaults.

Automatic Datatype Conversion of Parameters are also new to the JET engine. In earlier versions of the JET engine, you always declared the datatype of parameters in queries against tables stored on a SQL database server. Now you no longer have to declare the datatype of parameters in these types of queries. The JET engine 2.0 infers parameter datatypes from their context in the WHERE clause before sending them to the server.

Now that you have heard about the new and improved aspects of Access 2.0, you should also hear a little about the down side for Visual Basic programmers. Access 2.0 uses a file format for database files that is different from the format used by Access 1.1. Visual Basic 3.0, which includes the Access 1.1 JET engine, cannot use databases in Access 2.0 format directly. If you want to share databases between Access 2.0 and Visual Basic applications, you have two choices.

First, you can keep the database files in Access 1.1 format, which both Access 2.0 and Visual Basic 3.0 can use.

Second, you can install the JET Database Engine 2.0/Visual Basic 3.0 Compatibility Layer, which allows Visual Basic 3.0 applications to use database files in Access 2.0 format. (Note: The Compatibility Layer is included in both the Access Developer's Toolkit and the Office Developer's Kit, two products that are not part of Access 2.0 and must be purchased separately.)

The Compatibility Layer is described in more detail in the Database Design column on p. 63, but suffice it to say that it does not give your applications access to all the new JET engine 2.0 features.

Some new features like engine-level validation and cascading updates and deletes will work with the Compatibility Layer if they are first defined in Access 2.0, but applications won't be able to use any of the language enhancements. That will have to wait until later this year when Visual Basic 4.0 comes out. The Compatibility Layer is really just an interim solution to allow Visual Basic 3.0 programmers to use Access 2.0 databases until Visual Basic 4.0 comes out. ■