Interfaces for accessing databases are no different. Although SQL syntax is still used to communicate directly with databases, the query-by-example (QBE) approach to querying databases has proved to be a popular interface, too. The ease of use and inclusion of programming capabilities in desktop database products such as Microsoft Access have led many organizations to use such packages as a front-end attached to a more powerful back-end database such as Oracle or Informix. For example, by using pass-through queries, formulated in Access but run against the back-end Oracle database, organizations achieve faster development times but utilize the more robust and powerful RDBMSs for their mission-critical applications. The more powerful databases, such as Oracle and SQL Server, have also developed GUI and QBE interfaces for their products.

**USING QUERY-BY-EXAMPLE**

**Query-by-Example (QBE):** A direct manipulation database language that uses a graphical approach to query construction.

**Figure 9-8**

Common logic distributions
(a) Two-tier client server environments

(b) n-tier client/server environments
(There are many possibilities; these are just samples.)
been widely available, especially in the PC-RDBMS market, for many years, and is now widely available at all levels of database sophistication. As a direct-manipulation query language, it is easy to learn for a wide variety of people wanting to make inquiries against a database. Also, its simplicity makes it a popular language for developing prototypes. Because some database systems, such as Microsoft Access, translate QBE queries into SQL, QBE can be used as at least a first pass at creating SQL code. QBE-based systems that generate SQL code can then be used to build presentation or client layer modules that access server databases. In this section, you will learn about the history of QBE and many of the querying capabilities of this important database manipulation language.

The History and Importance of QBE

Although QBE (like SQL) was originally developed for mainframe database processing, it has become prevalent in client/server and personal computer database systems as well. QBE was originally developed by Zloof (1977) and was first available for use with SQL/DS and DB2 database systems on IBM mainframes. The success of the first personal computer relational DBMS (PC-RDBMS) that was completely based on QBE, Paradox, encouraged other products to adopt a QBE interface as an option. Most current systems include a variation on QBE.

Coverage of QBE is essential for the understanding of modern database systems. This is true because QBE is considered a highly productive language in which to program. Visual versions of programming languages, such as Visual Basic, Visual C, and Visual Java have transformed the way programs are written. Studies (e.g., Greenblatt and Waxman, 1978; Thomas and Gould, 1975) have shown that even with relatively little training, student subjects find QBE easier to use than SQL or a relational algebra language. Although these studies are more than two decades old, no new database query language has been developed that beats QBE’s usability.

QBE is especially useful for end-user database programming. The visual programming environment gives the nonprogramming user a single view of data no matter what database task is performed. As we will see (and as was shown in Chapter 2), queries are developed interactively on a CRT screen in a format that resembles the desired output. In most programs, both queries and results are shown in the same format, usually a spreadsheet-type format.

Complete database applications can be written in QBE, but it is more common to use QBE for interactive querying or updating of a database. That is, QBE is particularly useful for ad hoc database processing. Rather than developing a complete application in QBE, it is more common to use QBE to prototype an application, saving the queries that are developed. These saved queries can then be enhanced by using the application generator tools, such as screen, form, and report generators, and by adding modules of code using the database language associated with the PC-RDBMS to provide custom behaviors.

QBE: The Basics

No official QBE standard has been defined for SQL (see Chapter 8 for a discussion of the SQL standardization efforts). For this reason, there is no minimal set of capabilities that a query interface must meet to be considered a QBE implementation. However, because QBE has evolved from research on database query languages, because the Windows visual interface has been standardized, and because it is a visual language, all vendors have adopted similar approaches to QBE.

Both data retrieval and data modification can be done via entering keywords, constants, and example data into the cells of a table layout. Because data definitions are stored in internal tables, even data definition is done through a similar table layout interface. In Microsoft Access 2002 (used along with Oracle 9i in this chapter for
examples) clicking on a SQL button will reveal the Access SQL code that has been generated as the QBE-based query has been constructed.

Figure 9-9 depicts the original Microsoft Access usability hierarchy. The pyramid conveys that Access will be usable at all levels of programming ability and complexity. At the lowest level of the pyramid are the objects, which allow the creation of tables, queries, forms, and reports without any specific programming knowledge. The use of expressions, or functions, to perform simple processes such as multiplication of fields, validation of data, or enforcement of a specific business rule is possible at the next pyramid level. At the next level, macros, users can take advantage of stored modules of Visual Basic for Applications (VBA) code to automate their application, again without explicit knowledge of VBA. At the next level, users can program their own modules of VBA code to custom tailor their own applications. At the top level, Windows API calls to functions or DLLs written in languages such as C, Java, or Visual Basic that can be used to write interfaces to other programs and sources of data.

QBE provides a simple, visual method for specifying qualified queries. Data for display may be limited to certain columns and records with desired values, just as was shown in Chapter 8 when using SQL. In fact, the Access SQL that conveys a query constructed in the QBE pane can always be viewed by clicking on the SQL icon.

In Access 2002, the QBE pane (see Figure 9-10) has an upper workspace where data model representations of the tables or queries involved in a query are placed. Previously established relationships will also show automatically. Occasionally, another relationship is needed for a single query, and that relationship can also be established in the QBE pane. The lower part of the QBE pane, called the query design pane or QBE grid, displays a spreadsheet-type form in which the fields needed for the query are placed, along with any sorting and limiting criteria needed for the query. Each column in the QBE grid contains information about a single field from a table or query in the upper pane. Figure 9-10 shows a QBE pane for a select query involving the product and order line tables that asks for information about orders on which each product has been ordered. Note that the result displayed in Figure 9-11 is called a dynaset. Although the result looks just like a table, it is not a base table like the raw materials table on which it was based. Rather, it is a dynamic or virtual set of records and is not stored in the database. Figure 9-12 shows the Access SQL view of this query.

The benefits of a dynaset are that the storage device being used requires less storage space, and that whenever the query is run, it will use the current copy of the database, which will include any records that have been added or updated since the query was last run. Thus, queries behave just like relational views, discussed in Chapter 8. If you need to save the results of the query into a new table, use a make-table query.
A quick review of multiple-table queries in SQL (in Chapter 8) will remind you of the need to specify which tables are needed in the query (in the FROM clause) and to set the equalities between each table that is needed in the query (in the WHERE clause). These links can be time-consuming to code in SQL if one is not an excellent keyboarder. In Microsoft Access 2002 QBE, these links are established when the relationships are made explicit in the Relationships screen, and putting the tables needed in the QBE pane brings those links in automatically, thus eliminating the need to type in each link. This can be a great saving in time and accuracy as a...
query is developed. Look at Figure 9-10, in which the two tables to be used in the query have been added to the QBE pane and the fields required for the query result have been selected and dragged to the QBE grid. Figure 9-12 shows you how much of the SQL has already been established just by the simple clicking and dragging that has occurred. Notice that even though the QBE grid does not contain the primary key column from the Product_t table (Product_ID) that has been used to establish the links among the two tables, the correct linkages have been made explicit in the SQL. Notice also how Access SQL varies from the Oracle SQL*Plus used in Chapter 8 (you may want to write the Oracle SQL equivalent to the code in Figure 9-12). The differences make it quite clear that each vendor has its own flavor of SQL.

**Selecting Qualified Records**

What if you are only interested in certain records? Figure 9-13 shows the basic approach to entering qualifications for which records to display from a table. Placing conditions under the associated column does this. The dynaset returned is shown in Figure 9-14. The condition can be an inequality, as in Figure 9-13, or it could involve a range or equality. For example, you could enter “Between 350 and 700” under the Standard_Price column in Figure 9-13 to learn which products are mid-range priced.

Operators, functions, and expressions can also be used when constructing criteria in queries. Mathematical operators, such as multiplication and division, operate with numeric fields. The relational operators, such as equal, not equal, and less than,
can be used with numeric fields, date fields, and text fields. Boolean or logical operators are used for setting conditions in expressions.

Functions are modules of code that always return a value based on a calculation or comparison performed by the function. String, logic, or number values can be returned, depending on the nature of the function. Microsoft Access has many common functions already stored for use, such as sum or average, or specialized functions may be written in VBA. Expressions are a term or series of terms controlled by operators. Access will try to help with expression syntax by inserting needed characters, such as quotes, if they are left off.

**Self-Join**

Some queries are much easier to construct using a QBE interface because its visual representation is clearer than a complex syntax for most people. An example is the situation where a query requires a table to be joined with itself. This type of query is called a **self-join** or a **recursive join**. For example, suppose you want to know what orders have been placed that include any of the same products that were ordered on order number 1004. Such a query might arise because of, say, a packaging problem with this order, and you need it to determine if any other orders have had the same problem. An Access 2002 QBE query to answer this question appears in Figure 9-15a.
A self-join is done by putting two copies of the Order_Line_t table in the QBE pane, setting a relationship between product IDs (see the relationship line connecting the Product_ID fields in the two tables), and setting a criterion of 1004 in one table and a criterion of NOT 1004 in the other table that will return product ID and quantity values. It can be seen in Figure 9-15b that only product number 8 has been ordered on other orders. Three were ordered on order number 1008 and ten were ordered on order number 1010. The SQL (which could be run against a local or server database) is shown in Figure 9-15c.

**Basing a Query on Another Query**

It can be difficult even in QBE to answer a question in a single query. One way to deal with such difficult queries is to break a query into multiple steps, save the query generated to answer a particular step, and then base the next query on the saved query rather than the base tables. Such an approach is sometimes similar to using subqueries in SQL. So, basing a query on another query will allow you to compute values for SUM, COUNT, and so on using the dynaset generated by the saved query, thus solving some of the difficulties inherent in using functions with groups of records discussed in Chapter 8.

Suppose you want to find those customers who did not buy anything from Pine Valley Furniture during October 2004. That is, you want to know what customer numbers from records in the Customer_t table are not listed in the records of the Order_t table for October 2004. Figures 9-16a through 9-16d show you how you would answer this question by building one query and then building a second query that uses the first query. The first query (QBE in Figure 9-16a and dynaset result in Figure 9-16b) returns the customers who placed orders in October. The QBE for this query was saved in a query named First Query. The second query (QBE in Figure 9-16c and dynaset result in Figure 9-16d) then uses an outer join to compare the list of all customers (table Customer_t) with those who placed orders (the virtual table defined by First Query) and returns a dynaset with only those customers who are not found in the dynaset from the first query (see the Is Null qualification in Figure 9-16c).

**Using SQL Pass-Through Queries**

One reason that Microsoft Access has been popular as a client interface for client/server applications is the ease with which one can send commands to any ODBC (open database connectivity) database server directly. Using a pass-through query, which is written in the SQL dialect of the ODBC database server rather than
Figure 9-16
Query based on another query (Pine Valley Furniture Company)

designation of
(a) MS Access 2002 base query to be used for next query in Figure 9-16c

(b) MS Access 2002 dynaset returned by Figure 9-16a query

(c) MS Access 2002 query based on Figure 9-16a
Microsoft Access SQL, you work directly with the tables on the server instead of linking to them. This bypasses the Microsoft Access Jet database engine and gives faster performance. All syntax checking, interpretation, and translation of the SQL queries will take place on the server database. Network traffic will be reduced because only the initial SQL query and the records that are returned have to be passed between the server and the client.

Microsoft Access pass-through queries can be used to retrieve records, change data, or execute stored procedures or triggers located on the database server. You can even create new tables in the server database. Be careful, however, not to perform an operation that affects the state of the connection, as unexpected results may occur.

This method does not allow you to link the tables of the ODBC database to Microsoft Access. (Linking tables is covered later in this chapter.) This means that you cannot create an updateable record set based on these tables. The user must be familiar with the SQL dialect used by the ODBC database. So, although performance gains can be considerable and it is possible to take advantage of the power of the large database server, the user must be knowledgeable about the SQL dialect to be used and must be prepared to undertake more manual operations to achieve complete functionality.

Creation of an Microsoft Access pass-through query requires that a connection string be specified, either as a property of the pass-through query or at the time that the query runs. Figure 9-17 shows the Oracle SQL syntax and the properties window of a Microsoft Access 2002 pass-through query containing an Oracle SQL statement.

Figure 9-16 (Continued)
(d) MS Access 2002 dynaset returned by Figure 9-16c query

Figure 9-17
MS Access 2002 SQL pass-through query with query properties window
It is important to note that each RDBMS, such as Oracle, Informix, or SQL Server, will have its own syntax for the ODBC connect string that must be inserted into the properties window or entered at the time that the query is run. For example, the syntax of the Oracle connect string used here is:

```
ODBC;DSN = ODBC Connection Name;UID = User;PWD = Password;SERVER = Connection Alias from TnsNames.ora;
```

### USING ODBC TO LINK EXTERNAL TABLES STORED ON A DATABASE SERVER

The open database connectivity (ODBC) standard was developed in the early 1990s by the X/Open and SQL Access Group committees. It proposed several levels of standards that RDBMSs could attain, thus enabling any application program to access them using a common API for accessing and processing. Such RDBMSs are said to be ODBC-compliant. The standard has gained wide acceptance, originally propelled by Microsoft’s implementation of ODBC for their products. ODBC is also important for Internet applications because it allows for the development of applications that access different database products. In order to achieve this capability, ODBC uses the ANSI standard generic SQL statements presented in Chapter 8, but it is unable to take advantage of the extensions and special features that each vendor has given its engine.

The ODBC specification allows drivers to conform to various levels of the specification, and that affects the level of functionality of the drivers. Differences in how the drivers themselves are written may affect the performance achieved. Each vendor desiring to have an ODBC-compliant database provides an ODBC driver that can be installed on Windows machines. Thus, each Windows application can communicate, through the appropriate driver, with the desired version of the database server. For example, a Microsoft Access application can be connected to operate with an Oracle database server. The database tables are linked to the Microsoft Access application through the ODBC link and remain in the Oracle database. They are not brought into the Microsoft Access database.

You may hear the Oracle database server referred to as the database server, but it may also be called the remote server, the back-end server, or the SQL server. Because Microsoft’s back-end server is called SQL Server, a reference to SQL server may be a reference to a type of server or to a particular vendor database server. This can be confusing.

Five parameters must be defined in order to establish an ODBC connection:

1. Specific ODBC driver needed
2. Back-end server name to connect to
3. Database name to connect to
4. User id to be granted access to database
5. Password of user id

Additional information may be provided, if desired:

- Data source name (DSN)
- Windows client computer name
- Client application program’s executable name

These parameters may be defined from different locations. They may be included in the program, or through the DSN, or by the user when prompted. Including all of the parameters in the program will make it possible for the program to connect.