CHAPTER 1

Structured Query Language

From Inception to Now

ONCE UPON A TIME, 64KB came in a piano crate, not on a small chip, and maintaining and manipulating data was expensive. Memory wasn't the only problem: software was in its infancy, and simple tasks such as storing and indexing files required lots of support staff. As data grew, so did the need for better ways to maintain and manipulate that data. This need gave birth to the inexpensive hardware and efficient software we take for granted today.

Part of this evolution involved the methods for data storage and retrieval. Early databases relied on the flat-file format, in which all data was stored for each record in one large table. Of course, this arrangement led to redundancy and required lots of memory. A major goal during this period was to solve the problem of redundant data.

In 1970, Dr. E. F. Codd, an IBM researcher, published an article on relational database theory, and this article set the stage for the relational database model we use today. The article outlined a method for using relational calculus and algebra to store and retrieve data—large amounts of data—easily and efficiently. In a nutshell, this theory stored information in "tables" and allowed users to interact with the data using English commands. As a result, IBM introduced a new research group known as System/R.

The goal of this new group was to create a relational database system based on Codd's theory. A prototype was put into use in several organizations, and System/R eventually evolved into SQL/Data System (SQL/DS), which later found its real potential as DB2. The System/R project ended in 1979, but the viability of the relational data model was irrefutable.

Before ending the System/R project, IBM implemented a support language for System/R's multitable and multiuser access called Structured English Query Language, or SEQUEL. This early product was pronounced *sequel*. Eventually, the product became known and pronounced as *SQL* (*S-Q-L*). You may find people that still refer to it as *sequel*, but they have usually been working with the language for a long time.

The end result of ten year's worth of research and development work is now known as Structured Query Language (SQL), which has become the industry standard for relational databases. Today, dozens of vendors provide SQL for every environment, from the desktop PC to large mainframes.

Standards

During the late 1970s and throughout the 1980s, competitors came and went. Some, like Oracle, are still present, but most either went the way of the dinosaur or they realized that SQL was the future and adapted and adopted SQL as their own query language. Eventually, SQL became the industry standard, and today many versions of SQL are available.

The American National Standards Institute (ANSI) took notice and began work to create a standard definition for SQL. ANSI is a privately run, not-for-profit organization that coordinates voluntary standardization of the U.S. computer industry. You can learn more about this organization at `http://www.ansi.org`. The current standard is based mostly on the original IBM work with a lot of additions. ANSI eventually adopted SQL as the standard language for relational database management systems. This standard has also been adopted by the International Standards Organization (ISO) and the United States government.

NOTE *A European standard for SQL does exist. Known as X/OPEN, these standards support a UNIX-based environment. These standards play a vital role in the European market, but they are very different from the ANSI standards used throughout the U.S. industry.*

ANSI published its first SQL standard—SQL-86—in 1986. Subsequent updates (SQL-92 and SQL-99) were published in 1992 and 1999, respectively. Incremental parts are published as new technologies and functionality emerge.

You can download electronic copies of the standards from ANSI's Electronic Standards Store at `http://webstore.ansi.org`. Or, you can purchase hard copies (although very expensive) from American National Standards Institute, 1819 L Street, NW Suite 600, Washington, DC 20036.

The three standards are listed in Table 1-1 with their common names.

Table 1-1. ANSI Standards for SQL

VERSION	COMMON NAME	AVAILABLE
ANSI X3.135-1986	SQL-86	Hardcopy only
ANXI X3.135-1992	SQL-92	Online
ANSI X3.135-1999	SQL-99	Online

The Many Dialects

Although SQL is the accepted standard, SQL is available in many dialects, all of which are unique to their vendors. In fact, it's safe to say that SQL is available, in some form, for every important computer platform. Most of these dialects agree on simple data retrieval commands, such as SELECT. However, differences develop as the products add customized functionality. These differences are known as *vendor-specific extensions*. Because hundreds of database products support SQL, many different dialects are in use, and there's little hope that SQL will ever evolve into just one dialect.

Even products made by the same vendor (Microsoft in this case) use different dialects of SQL. Access doesn't really have an SQL dialect of its own. Access relies on Jet SQL, and SQL Server Desktop Engine and SQL Server 2000 both use Transact-SQL (T-SQL). This book focuses on Jet SQL and T-SQL.

Furthermore, this book limits its focus to Jet SQL and T-SQL in the Microsoft products Access, SQL Server Desktop Engine (MSDE), and SQL Server 2000, and using SQL with these products. Jet SQL extensions aren't supported via the query design graphic interface (Query Design window). You can access extensions only by using ADO code and OLEDB providers. Table 1-2 compares SQL tasks in both MS dialects.

Table 1-2. Differences Between Jet SQL and Transact-SQL (T-SQL)

FEATURE/KEYWORD	JET SQL	JET SQL EXTENSIONS	T-SQL
JOIN in UPDATE	○	○	∅*
JOIN in DELETE	○	○	∅*
DELETE with multiple tables	○	○	∅
Subquery in UPDATE/SET	∅	∅	○
DISTINCTROW	○	○	∅
TOP	○	○	○
TRANSFORM	○	○	∅
SELECT INTO	○	○	○
CREATE DOMAIN	∅	∅	○
ALTER DOMAIN	∅	∅	○
DECLARE CURSOR	∅	∅	○
FETCH	∅	∅	○
CREATE ASSERTION	∅	∅	○
DROP ASSERTION	∅	∅	○
UNION	○	○	○
UNION JOIN	∅	∅	○
FULL OUTER JOIN	∅	∅	○
PIVOT	○	○	∅
COMPUTE	∅	∅	○
FOR BROWSE	∅	∅	○
OPTION	∅	∅	○
ORDER BY	○	○	∅ (in views)
WITH OWNER ACCESS	∅	○	∅
GRANT	∅	○	○
LOCK	∅	○	○
FIRST	○	○	∅
LAST	○	○	∅
PIVOT	○	○	∅

Table 1-2. Differences Between Jet SQL and Transact-SQL (T-SQL) (continued)

FEATURE/KEYWORD	JET SQL	JET SQL EXTENSIONS	T-SQL
CREATE VIEW	∅	○	○
CHECK	∅	○	○

○ Supported

∅ Not supported

* Can be done in T-SQL if you use a second FROM clause in the form

```
UPDATE tblX Set X=Y
FROM tbl1, tbl2, etc.
```

This book deals with using Jet SQL or T-SQL in Access, SQL Server Desktop Edition, and SQL Server. Knowing how each product responds to SQL will help you create efficient solutions using the best product for each project.