"Basic SQL"

The SQL language considered one of the major reasons for the commercial success of relational databases. ***Because*** it became a standard for relational databases, users were less concerned about migrating their database applications from other types of database systems—for example,network or hierarchical systems—to relational systems.

This is ***because*** even if the users became dissatisfied with the particular relational DBMS product they were using, converting to another relational DBMS product was not expected to be too expensive and time-consuming because both systems followed the same language standards.

In practice, there are many differences between various commercial relational DBMS packages. However, if the user is diligent in using only those features that are part of the standard, and if both relational systems faithfully support the standard, then conversion between the two systems should be much simplified.

Another advantage of having such a standard is that users may write statements in a database application program that can access data stored in two or more relational DBMSs without having to change the database sublanguage (SQL) if both relational DBMSs support standard SQL.

The name SQL is Structured Query Language. ***Originally, SQL*** called SEQUEL (Structured English QUEry Language) and was designed and implemented at IBM Research for an experimental relational database system called SYSTEM R.

SQL is now the standard language for commercial relational DBMSs. ***A joint effort by*** the American National Standards Institute (ANSI) and the International Standards Organization (ISO) has led to a standard version of SQL (ANSI 1986), called SQL-86 or SQL1. Much expanded standard called SQL-92 (SQL2) was developed. The next standard that is well-recognized is SQL: 1999 (SQL3). Two later updates to the standard are SQL: 2003 and SQL: 2006, which added XML features among other updates to the language. Another update in 2008 incorporated more object database features in SQL.

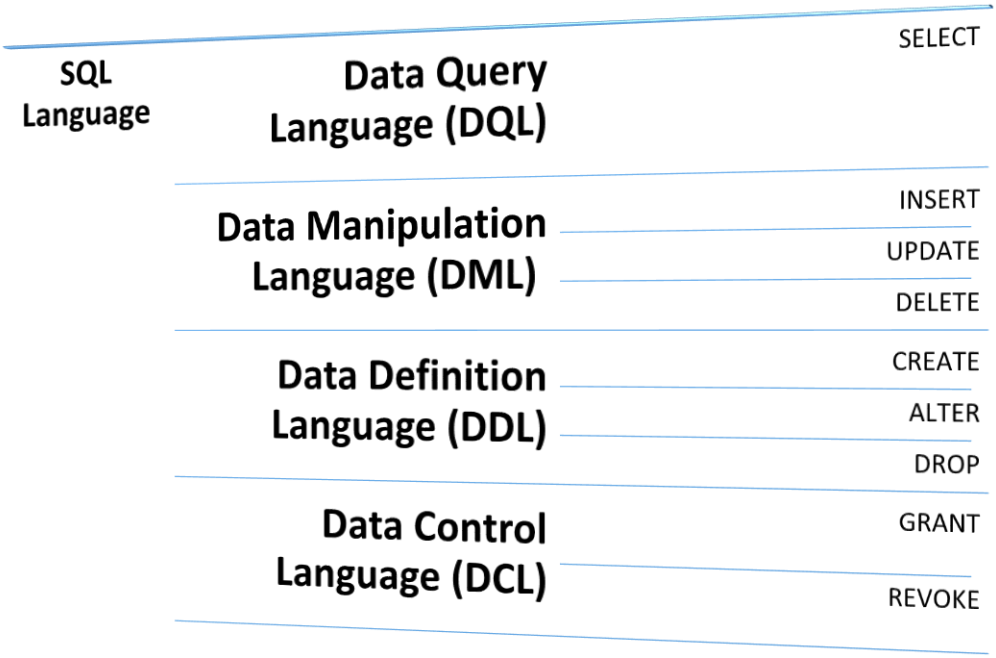
SQL is a comprehensive database language: It has statements for data definitions, queries, and updates. Hence, it is both a DDL and a DML. In addition, it has facilities for defining views on the database, for specifying security and authorization, for defining integrity constraints, and for specifying transaction controls. It also has rules for embedding SQL statements into a general-purpose programming language such as Java, COBOL, or C/C++.

The later SQL standards (starting with SQL: 1999) are divided into a ***core*** specification plus specialized ***extensions***.

The core is supposed to be implemented by all RDBMS vendors that are SQL compliant. *The extensions* can be implemented as optional modules to be purchased independently for specific database applications such as data mining, spatial data, temporal data, data warehousing, online analytical processing (OLAP),multimedia data, and so on.

SQL language can be divided into the following categories, as shown in figure 1:

- **Data Query Language (DQL)** Statements that query the database but do not alter any data or database objects. This category contains the SELECT statement. Not all vendors make a distinction here; many lump DQL into DML, as defined next.
- **Data Manipulation Language (DML)** Statements that modify data stored in database objects (that is, tables). This category contains the INSERT, UPDATE, and DELETES statements.
- **Data Definition Language (DDL)** Statements that create and modify database objects. Whereas DML and DQL work with the data in the database objects, DDL works with the database objects themselves. In other words, DDL manages the data containers whereas DML manages the data inside the containers. This category includes the CREATE, ALTER, and DROP statements.
- **Data Control Language (DCL)** Statements that manage privileges that database users have regarding the database and objects stored in it. This category includes the GRANT and REVOKES statements.



(Figure 1) Categories of SQL Languages

SQL Data Definition and Data Types

SQL uses the terms table, row, and column for the formal relational model terms relation, tuple, and attribute, respectively.

The main SQL command for data definition is the **CREATE statement**, which can be used to create schemas, tables (relations), and domains (as well as other constructs such as views, users, index, and triggers).

- The CREATE TABLE Command in SQL

The CREATE TABLE command used to specify a new relation by giving it a name and specifying its attributes and initial constraints.

```
CREATE TABLE table_name
(attribute1   data type,
 attribute2   data type,
 attribute3   data type);
```

Ex:

DEPT

DeptNo	DName	Loc

```
CREATE TABLE DEPT
        (DeptNo INT(2),
         DName VARCHAR(15),
         Loc VARCHAR(15));
```

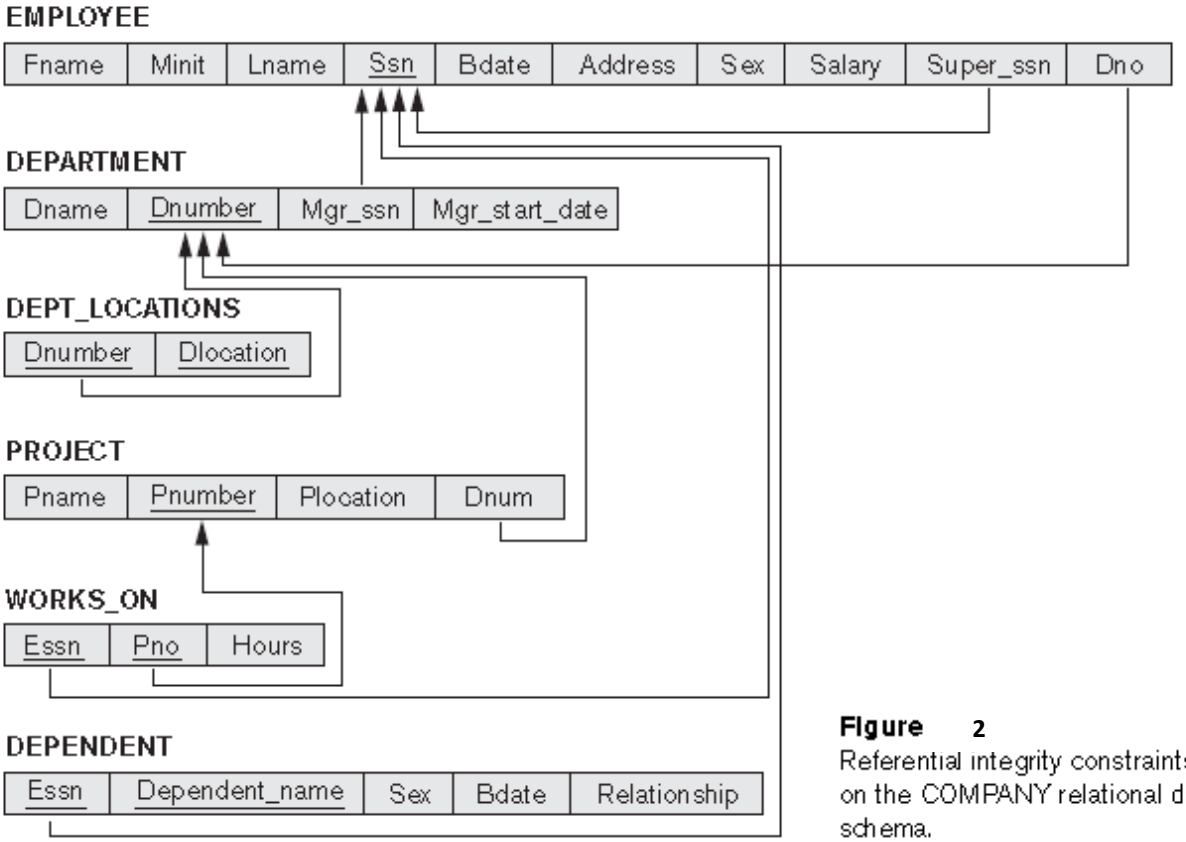


Figure 2
Referential integrity constraints displayed on the COMPANY relational database schema.

```
CREATE TABLE EMPLOYEE
        ( Fname             VARCHAR(15)         NOT NULL,
          Minit             CHAR,
          Lname             VARCHAR(15)         NOT NULL,
          Ssn               CHAR(9)             NOT NULL,
          Bdate             DATE,
          Address           VARCHAR(30),
          Sex               CHAR,
          Salary            DECIMAL(10,2),
          Super_ssn         CHAR(9),
          Dno               INT                 NOT NULL,
        PRIMARY KEY (Ssn),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE DEPARTMENT
        ( Dname             VARCHAR(15)         NOT NULL,
          Dnumber           INT                 NOT NULL,
          Mgr_ssn           CHAR(9)             NOT NULL,
          Mgr_start_date    DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber           INT                 NOT NULL,
          Dlocation         VARCHAR(15)         NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname             VARCHAR(15)         NOT NULL,
          Pnumber           INT                 NOT NULL,
          Plocation         VARCHAR(15),
          Dnum              INT                 NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn              CHAR(9)             NOT NULL,
          Pno               INT                 NOT NULL,
          Hours             DECIMAL(3,1)        NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn              CHAR(9)             NOT NULL,
          Dependent_name    VARCHAR(15)         NOT NULL,
          Sex               CHAR,
          Bdate             DATE,
          Relationship      VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Figure 4.1
SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 3.7.

Attribute Data Types in SQL

The basic data types available for attributes include numeric, character string, bit string, Boolean, date, and time.

■ **Numeric data types** include ***integer numbers*** of various sizes (INTEGER or INT, and SMALLINT) and ***floating-point (real) numbers*** of various precision (FLOAT or REAL, and DOUBLE PRECISION). *Formatted numbers* can be declared by using DECIMAL(i,j)—or DEC(i,j) or NUMERIC(i,j)—where i, the precision, is the total number of decimal digits and j, the scale, is the number of digits after the decimal point.

■ **Character-string data types** are either ***fixed length***—CHAR(n) or CHARACTER(n), where n is the number of characters—or ***varying length***—VARCHAR(n) or CHAR VARYING(n) or CHARACTER VARYING(n),where n is the maximum number of characters.

■ **Bit-string data types** are either of ***fixed length*** n—BIT(n)—or ***varying length***—BIT VARYING(n), where n is the maximum number of bits. Another ***variable-length bitstring data type*** called BINARY LARGE OBJECT or BLOB is also available to specify columns that have large binary values, such as images.

■ **A Boolean data type** has the traditional values of ***TRUE or FALSE***.

In SQL, because of the presence of NULL values, a **three-valued logic used**, so a third possible value for a Boolean data type is UNKNOWN.

■ **The DATE data type** its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD. **The TIME data type** with the components HOUR,MINUTE,and SECOND in the form HH:MM:SS.

Additional data types

■ **A timestamp data type** (TIMESTAMP) includes the DATE and TIME fields, plus a minimum of six positions for decimal fractions of seconds and an optional WITH TIME ZONE qualifier; for example, TIMESTAMP '2008-09-27 09:12:47.648302'.

■ **INTERVAL data type.** This specifies an interval—a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp. Intervals are qualified to be either YEAR/MONTH intervals or DAY/TIME intervals.