**Basic SQL**

The SQL language may be considered one of the major reasons for the commercial success of relational databases.

**The CREATE TABLE Command in SQL**

The CREATE TABLE command is used to specify a new relation by giving it a name and specifying its attributes and initial constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain of values, and any attribute constraints, such as NOT NULL. The key, entity integrity, and referential integrity constraints can be specified within the CREATE TABLE statement after the attributes are declared, or they can be added later using the ALTER TABLE command.

Syntax: **CREATE TABLE EMPLOYEE...**

The relations declared through CREATE TABLE statements are called base tables (or base relations); this means that the relation and its tuple are actually created and stored as a file by the DBMS. Base relations are distinguished from virtual relations, created through the CREATE VIEW statement, which may or may not correspond to an actual physical file. In SQL, the attributes in a base table are considered to be ordered in the sequence in which they are specified in the CREATE TABLE statement. However, rows (tuple) are not considered to be ordered within a relation.

**Attribute Data Types and Domains in SQL**

**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. The default for scale is zero, and the default for precision is implementation-defined.

**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.

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

The **DATE** data type has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD. The **TIME** data type has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form HH:MM:SS. Only valid dates and times should be allowed by the SQL implementation.

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. Literal values are represented by single quoted strings preceded by the keyword TIMESTAMP, with a blank space between data and time; for example, TIMESTAMP ‘2008-09-27 09:12:47.648302’.

**Specifying Constraints in SQL**

SQL allows NULLs as attribute values, a constraint **NOT NULL** may be specified if NULL is not permitted for a particular attribute. This is always implicitly specified for the attributes that are part of the primary key of each relation, but it can be specified for any other attributes whose values are required not to be NULL.

It is also possible to define a default value for an attribute by appending the clause **DEFAULT** to an attribute definition. The default value is included in any new tuple if an explicit value is not provided for that attribute.

**CHECK** clause following an attribute or domain definition.6 For example, suppose that department numbers are restricted to integer numbers between 1 and 20; then, we can change the attribute declaration of Dnumber in the DEPARTMENT table.

Example: Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);

**Specifying Key and Referential Integrity Constraints**

The PRIMARY KEY clause specifies one or more attributes that make up the primary key of a relation. If a primary key has a single attribute, the clause can follow the attribute directly. For example, the primary key of DEPARTMENT can be specified as follows:

Syntax: Dnumber INT PRIMARY KEY;

The **UNIQUE** clause can also be specified directly for a secondary key if the secondary key is a single attribute, as in the following:

example: Dname VARCHAR(15) UNIQUE;

**The SELECT-FROM-WHERE Structure of Basic SQL Queries**

SELECT FROM

|  |  |
| --- | --- |
| WHERE ;  Where is a list of attribute names whose values are to be retrieved by the query.   |  | | --- | | is a list of the relation names required to process the query. is a conditional (Boolean) expression that identifies the tuple to be retrieved by the query. In SQL, the basic logical comparison operators for comparing attribute values with one another and with literal constants are =, <=, >, >=, and <>. These correspond to the relational algebra operators =, , ≥, and ≠, respectively, and to the C/C++ programming language operators =, <=, >, >=, and !=. The main syntactic difference is the not equal operator. | |



