Basic SQL

Part 1

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.

Schema and Catalog Concepts in SQL

- An **SQL schema** is identified by a **schema name**, and includes an **authorization identifier** to indicate the user or account who owns the schema, as well as **descriptors** for *each element* in the schema.
- Schema **elements** include tables, constraints, views, domains, and other constructs (such as authorization grants) that describe the schema.

- A schema is created via the `CREATE SCHEMA` statement, which can include all the schema elements’ definitions.
- Alternatively, the schema can be assigned a name and authorization identifier, and the elements can be defined later.
- For example, the following statement creates a schema called `COMPANY`, owned by the user with authorization identifier `'Jsmith'`.
  - Note that each statement in SQL ends with a semicolon.
  - **`CREATE SCHEMA`** `COMPANY` **`AUTHORIZATION`** `'Jsmith';`

- In general, not all users are authorized to create schemas and schema elements.
- The privilege to create schemas, tables, and other constructs must be explicitly granted to the relevant user accounts by the system administrator or DBA.

- In addition to the concept of a schema, SQL uses the concept of a **catalog**—a named collection of schemas in an SQL environment.
- An SQL **environment** is basically an installation of an SQL-compliant RDBMS on a computer system.
- A catalog always contains a special schema called `INFORMATION_SCHEMA`, which provides information on all the schemas in the catalog and all the element descriptors in these schemas.
- Integrity constraints such as referential integrity can be defined between relations only if they exist in schemas within the same catalog.
- Schemas within the same catalog can also share certain elements, such as domain definitions.

The CREATE TABLE Command in SQL

```
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) );
```

- Typically, the SQL schema in which the relations are declared is implicitly specified in the environment in which the CREATE TABLE statements are executed.
- Alternatively, we can explicitly attach the schema name to the relation name, separated by a period.
    - `CREATE TABLE COMPANY.EMPLOYEE ...`

- The relations declared through `CREATE TABLE` statements are called **base tables** (or base relations); this means that the relation and its tuples 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 (tuples) are not considered to be ordered within a relation.

Attribute Data Types and Domains in SQL

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

- It is possible to specify the data type of each attribute directly; alternatively, a domain can be declared, and the domain name used with the attribute specification.
- This makes it easier to change the data type for a domain that is used by numerous attributes in a schema, and improves schema readability.
- For example, we can create a domain SSN_TYPE by the following statement:

```
CREATE DOMAIN SSN_TYPE AS CHAR(9);
```