Unit- 5: Creating and Altering Database and Tables

Query and Query Languages

- Query → is statement requesting for retrieval of information.
- Query Language→ a language through which user request information from database.
- Two types of query languages - **procedural** and **non-procedural**
- In a procedural language the user instructs the system to perform a sequence of operations on the database to compute the desired result. Example: Relational algebra.
- In a non-procedural language, the user describes the desired information without giving a specific procedure for obtaining that information. Example: tuple relational calculus, domain relational calculus, SQL etc.

Introduction to SQL

IBM developed the original version of SQL, originally called Sequel, as part of the System R project in the early 1970s. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query Language). Many products now support the SQL language. SQL has clearly established itself as the standard relational database language.

In 1986, the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) published an SQL standard, called SQL-86. ANSI published an extended standard for SQL, SQL-89, in 1989. The next version of the standard was SQL-92 standard, followed by SQL:1999, SQL:2003, SQL:2006, SQL:2008 and SQL:2011. SQL:2016 .is current version (as for 2019).

- SQL works with relational database.
- SQL is used to control all of the functions that a DBMS provides for its users, including:
  1. **Data definition**: SQL lets a user define the structure and organization of the stored data and relationships among the stored data items.
  2. **Data retrieval**: SQL allows a user or an application program to retrieve stored data from the database and use it.
  3. **Data manipulation**: SQL allows a user or an application program to update the database by adding new data, removing old data, and modifying previously stored data.
  4. **Access control**: SQL can be used to restrict a user's ability to retrieve, add, and modify data, protecting stored data against unauthorized access.

5. **Data sharing**: SQL is used to coordinate data sharing by concurrent users, ensuring that they do not interfere with one another.
6. **Data integrity:** SQL defines integrity constraints in the database, protecting it from corruption due to inconsistent updates or system failures.

The SQL language has several parts:

- **Data-definition language (DDL) -** The SQL DDL provides commands for defining relation schemas, deleting relations, and modifying relation schemas.
- **Data-manipulation language (DML) -** The SQL DML provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- **Integrity** - The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed.
- **View definition-** The SQL DDL includes commands for defining views.
- **Transaction control -** SQL includes commands for specifying the beginning and ending of transactions.
- **Embedded SQL and dynamic SQL** - Embedded and dynamic SQL define how SQL statements can be embedded within general-purpose programming languages, such as C, C++, and Java.
- **Authorization-** The SQL DDL includes commands for specifying access rights to relations and views.

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database

Data Definition Language

- DDL specifies the database schema
- The set of relations in a database must be specified to the system by means of a data-definition language (DDL). The SQL DDL allows specification of not only a set of relations, but also information about each relation, including:
    - The schema for each relation.
    - The types of values associated with each attribute.
    - The integrity constraints.
    - The set of indices to be maintained for each relation.
    - The security and authorization information for each relation.
    - The physical storage structure of each relation on disk

The important DDL statements in SQL are:

- CREATE DATABASE- creates a new database
- ALTER DATABASE- modifies a database
- CREATE TABLE- creates a new table
- ALTER TABLE- modifies a table
- DROP TABLE- deletes a table
- CREATE INDEX- creates an index (search key)
- DROP INDEX- deletes an index

Domain Types in SQL (Basic Data types)

The SQL standard supports a variety of built-in types, including:

• **char(n):** A fixed-length character string with user-specified length n. The full form, character, can be used instead.

• **varchar(n):** A variable-length character string with user-specified maximum length n. The full form, character varying, is equivalent.

• **int:** An integer (a finite subset of the integers that is machine dependent). The full form, integer, is equivalent.

• **smallint:** A small integer (a machine-dependent subset of the integer type).

• **numeric(p, d):**Afixed-point number with user-specified precision. The number consists of p digits (plus a sign), and d of the p digits are to the right of the decimal point. Thus, numeric(3,1) allows 44.5 to be stored exactly, but neither 444.5 or 0.32 can be stored exactly in a field of this type.

• **real, double precision**: Floating-point and double-precision floating-point numbers with machine-dependent precision.

• **float(n):** A floating-point number, with precision of at least n digits.

Creating and altering databases

- The CREATE DATABASE statement is used to create a new SQL database.
  Syntax:

  ```
  CREATE DATABASE databasename;
  ```

- In SQL, we can alter or change the name of the SQL database either by using ALTER DATABASE statement
  Syntax:

  ```
  ALTER DATABASE olddbname MODIFY NAME= newdbname
  ```

- By using “DROP Database” statement we can drop or delete the SQL database. Schema Definition in SQL.
  Syntax:

  ```
  DROP DATABASE databasename
  ```

Schema Definition in SQL.

- An SQL relation is defined using the **create table** command:

create table *r*
(A_1 D_1,
A_2 D_2,
...,
A_n D_n,
⟨integrity-constraint$_1$⟩,
...,
⟨integrity-constraint$_k$⟩);

where,

- r is the name of the relation
- each A_i is an attribute name in the schema of relation r
- D_i is the data type of values in the domain of attribute A_i

Example:

```
create table instructor(
ID char(5),
name varchar(20),
dept_name varchar(20),
salary numeric(8,2) );
```

Integrity Constraints in Create Table

SQL supports a number of different integrity constraints. In this section, we discuss only a few of them:

- **primary key (A_{j1} , A_{j2}, . . . , A_{jm}):** The primary-key specification says that attributes **A_{j1} , A_{j2}, . . . , A_{jm}** form the primary key for the relation. The primary key attributes are required to be **nonnull** and **unique;** that is, no tuple can have a null value for a primary-key attribute, and no two tuples in the relation can be equal on all the primary-key attributes. Although

the primary-key specification is optional, it is generally a good idea to specify a primary key for each relation.

- **foreign key (A_{k1} , A_{k2}, . . . , A_{kn}) references s:** The foreign key specification says that the values of attributes (A_{k1} , A_{k2}, . . . , A_{kn}) for any tuple in the relation must correspond to values of the primary key attributes of some tuple in relation **s**.
- **not null:** The not null constraint on an attribute specifies that the null value is not allowed for that attribute; in other words, the constraint excludes the null value from the domain of that attribute.

Examples

```
create table instructor (
        ID          char(5) ,
        name        varchar(20) not null,
        dept_name   varchar(20),
        salary      numeric(8,2),
        primary key (ID),
        foreign key (dept_name) references
department(dept_name));
```

And a Few More Relation Definitions

- **create table** *student* (
  ID **varchar**(5),
  name **varchar**(20) not null,
  dept_name **varchar**(20),
  tot_cred **numeric**(3,0),
  primary key *(ID),*
  foreign key *(dept_name*) **references** *department*);
- **create table** *takes* (
  ID **varchar**(5),
  course_id **varchar**(8),
  sec_id **varchar**(8),
  semester **varchar**(6),
  year **numeric**(4,0),
  grade **varchar**(2),
  primary key *(ID, course_id, sec_id, semester, year)* ,
  foreign key (*ID*) **references** *student,*
  foreign key (*course_id, sec_id, semester, year*) **references** *section*);

- **create table** *course* (
  course_id **varchar**(8),
  title **varchar(**50),
  dept_name **varchar**(20),
  credits **numeric**(2,0),
  primary key *(course_id),*

foreign key *(dept_name*) **references** *department*);

Altering tables

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- **To add Column**

  To add a column in a table, use the following syntax is used

  Syntax:

```
ALTER TABLE table_name
ADD column_name datatype;
```

  Example:

```
ALTER TABLE Customers
ADD Email varchar(255);
```

- **To delete a column:**

  To delete a column in a table, use the following syntax
  Syntax:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

  Example

```
ALTER TABLE Customers
DROP COLUMN Email;
```

- **To modify a column**

  To modify a column in an existing table, the SQL ALTER TABLE syntax is:

```
ALTER TABLE table_name
MODIFY column_name column_type;
```

Example

```
ALTER TABLE Customers
MODIFY Email varchar(50) NOT NULL;
```

- **To rename a column**

To rename a column in an existing table, the SQL ALTER TABLE syntax is:

```
ALTER TABLE table_name
RENAME COLUMN old_name TO new_name;
```

Example:

```
ALTER TABLE supplier
RENAME COLUMN supplier_name TO sname;
```

Adding Constraints to a table

- The ADD CONSTRAINT command is used to create a constraint after a table is already created.

  Example:

```
ALTER TABLE Users
ADD CONSTRAINT id_name_unique UNIQUE (id, name);
```

  This command adds the unique constraint to the columns id and name of the table users. Here name of the constraint is id_name_unique.

Removing Constraints from a table

- The DROP CONSTRAINT command is used to delete constraints like UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, etc.

  Example

```
ALTER TABLE Users
DROP CONSTRAINT id_name_unique;
```

Drop Statements: Table, Database

- DROP DATABASE command is used to delete the entire database and DROP TABLE command is used to delete entire table in a database.
  DROP DATABASE databasename; => to drop database
  DROP TABLE tablename; => to drop table

Lab format

1. Create database query
   Syntax: create database <databasename>;
   Query
   mysql> create database testdb;
   output:
2. Drop database
3. Create table
4. Alter table ->column add, delete column, modify column, rename column
5. Drop table
6. Add and removing constraints