

Chapter 5

Structured Query Language (SQL)

Health Informatics Track

Nov 2022

Outline

- Data Types
- SQL Data Definition Language (DDL)
- SQL Data Manipulation Language (DML)
- Specifying Constraints in SQL
- Basic Retrieval Queries in SQL
- Additional Features of SQL

Introduction

- 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,
 - specifying security and authorization,
 - defining integrity constraints, and
 - specifying transaction controls.

Data Types in SQL

- SQL uses the terms **table**, **row**, and **column** for the 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), types, and domains, as well as other constructs such as views, assertions, and triggers.
- Before we describe the **CREATE** let us see data types in SQL

Attribute Data Types and Domains in SQL

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

1. Numeric data types include:

- o **INT**, and **SMALLINT**: integer numbers of various sizes
- o **FLOAT**: floating-point (real) numbers of various precision
- Formatted numbers can be declared by using **DECIMAL(i, j)** or **DEC(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.

Attribute Data Types...

2. Character (string) data types are:

- **CHAR** (n) or **CHARACTER** (n): fixed length characters of (n= number of characters)
- **VARCHAR**(n) or **CHAR VARYING**(n) or **CHARACTER VARYING**(n): varying characters, (n= the maximum number of characters).
- When specifying a literal string value, it is placed between **single quotation** marks
- For fixed-length strings, a **shorter string is padded with blank characters to the right**. For example, if the value 'Smith' is for an attribute of type CHAR(10), it is padded with five blank characters.

Attribute Data Types...

- Another variable-length string data type called **CHARACTER LARGE OBJECT** or **CLOB** is also available to specify columns that have large text values, such as documents.
- The CLOB maximum length can be specified in kilobytes (K), megabytes (M), or gigabytes (G).
- For example, **CLOB(20M)** specifies a maximum length of 20 megabytes.

Attribute Data Types...

3. A Boolean data type is three-valued logic and has the values of **TRUE**, **FALSE** or **UNKNOWN**
4. The DATE data type has ten positions, and its components are **YEAR**, **MONTH**, and **DAY** in the form **YYYY-MM-DD**.
5. The TIME data type has at least eight positions, with the components **HOUR**, **MINUTE**, and **SECOND** in the form **HH:MM:SS**.
- Literal values are represented by single-quoted strings preceded by the keyword **DATE** or **TIME**; for example, **DATE '2014-09-27'** or **TIME '09:12:47'**.

SQL Data Definition Language (DDL)

- In SQL, the DDL is used create the framework of the database by specifying the database schema, which is the structure that represents the organization of data.
- Here's a list of DDL statements in MSSQL:
 - CREATE
 - ALTER
 - DROP
 - EXEC sp_rename

DDL (1): **CREATE**

- The **CREATE** command in SQL is used to create:
 - Schemas
 - Databases
 - Tables
 - Index
 - Unique index
 - View
 - CREATE OR ALTER VIEW
 - Procedures, etc.
 - Triggers

DDL (1): **CREATE . . .**

CREATE SCHEMA

- A **schema** is a collection of database objects like tables, triggers, stored procedures, etc.
- Database may have one or more schema.
- SQL Server have some built-in schema, for example: dbo, guest, sys, and INFORMATION_SCHEMA.
- dbo is default schema for a new database, owned by dbo user.
- While creating a new user with CREATE USER command, user will take dbo as its default schema.
- CREATE SCHEMA statement used to create a new schema in current database.
- **CREATE SCHEMA schemaname;**

DDL (1): **CREATE . . .**

- Example: create a schema for COMPANY DATABASE as comp
- **CREATE SCHEMA COMPANY;**
- To list all schema in the current database, use query as shown below :
- **SELECT *FROM sys.schemas**
- To delete user-defined schema,
- **DROP SCHEMA SCHEMANAME**

Example replace the **COMPANY** Schema with **comp**
DROP SCHEMA COMPANY then,
CREATE SCHEMA comp

DDL (1): **CREATE...**

CREATE DATABASE

- The **CREATE DATABASE** command is used to create databases new SQL database.
- The following SQL creates a database called "COMPANY":

CREATE DATABASE COMPANY;

- --To activate the COMPANY database and create its table:

USE COMPANY;

DDL (1): **CREATE...**

CREATE TABLE

- The **CREATE TABLE** command is used to specify a new relation (table)
 - by giving it a **name** and
 - specifying its **attributes (columns)** 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 possibly **attribute constraints**, such as **NOT NULL**.
- The **PK, 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

DDL (1): **CREATE . . .**

- 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.
- The following SQL creates a a table called “COMPANY” in the COMPANY DATABASE:
- USE COMPANY

CREATE TABLE COMPANY.comp;

- The .comp is schema.

DDL (1): **CREATE . . .**

Exercise

- In groups of three, map the logical schema of the COMPANY database into SQL RDB
- Create a schema called comp
- Create a database called COMPANY
- In the COMPANY database, create the 6 tables
- In multiple line comments , write your name and id
- Finish in ten minutes and I will collect with USB pen

DDL (1): CREATE . . .

CREATE VIEW

- In SQL, a view is a **virtual table** based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table.
- The fields in a view are fields from one or more real tables in the database.
- You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.
- A view is created with the CREATE VIEW statement.

DDL (1): **CREATE...**

- **CREATE VIEW Syntax**

```
CREATE VIEW view_name AS SELECT column1,  
column2, ... FROM table_name WHERE condition;
```

- Example, from sales.customers table of Bikestore database, if you want to view only from those state=NY*/

```
USE Bikestores;
```

```
CREATE VIEW NYcustomers AS SELECT customer_id,  
first_name, last_name, email FROM  
sales.customers WHERE state= 'NY';
```

DDL (1): **CREATE...**

```
SELECT COUNT(*) FROM sales.customers; ---1445
SELECT COUNT (*) FROM sales.customers WHERE
state='NY'; --1019
SELECT COUNT (*) FROM NYcustomers; --1019
SELECT * FROM NYcustomers;
```

DDL (1): **CREATE...**

CREATE OR ALTER VIEW

- The CREATE OR REPLACE VIEW command updates a view.
- The following SQL adds the “phone” column to the “NYcustomers” view:

```
CREATE OR ALTER VIEW NYcustomers AS SELECT  
customer_id, first_name, last_name, email, phone  
FROM sales.customers WHERE state= 'NY';
```

DDL (1): **CREATE...**

CREATE STORED PROCEDURES

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

- Stored Procedure Syntax:

```
CREATE PROCEDURE procedure_name  
AS sql_statement;
```

- **Execute a Stored Procedure**

```
EXEC procedure_name;
```

DDL (1): CREATE...

CREATE STORED PROCEDURES...

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- The following SQL statement creates a stored procedure named "SelectAllCustomers" that selects all records from the "sales.customers" table in Bikestore database:

• Example

```
CREATE PROCEDURE SelectAllCustomers AS  
SELECT * FROM sales.Customers;
```

DDL (1): **CREATE . . .**

CREATE STORED PROCEDURES...

- --Execute the stored procedure above as follows:

```
EXEC SelectAllCustomers; PROCEDURE  
SelectAllCustomers  
AS SELECT * FROM Customers;
```

- Execute the stored procedure above as follows:
- **Example**
- **EXEC** SelectAllCustomers;

DDL (2): **ALTER**

- SQL ALTER TABLE Statement is used to
 - add,
 - delete, or
 - modify columns in an existing table.
- **ALTER**, and **DROP** commands are also called schema change commands
- **INSERT INTO** is schema evolution command

DDL (2): **ALTER**

ALTER TABLE - ADD Column

- To add a column in a table, use the following syntax:
 - **ALTER TABLE** `table_name` **ADD** `column_name` `datatype`;
- **ALTER TABLE - DELETE a column**
- To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):
 - **ALTER TABLE** `table_name` **DROP COLUMN** `column_name`;

DDL (2): **ALTER**

- **ALTER TABLE - ALTER/MODIFY COLUMN**

- To change the data type of a column in a table, use the following syntax:

```
ALTER TABLE table_name ALTER COLUMN column_name  
datatype;
```

DDL (3): **EXEC SP_RENAME**

- To rename table names in SQL Server,

```
EXEC sp_rename 'old_table_name',  
'new_table_name'
```

SQL Data Manipulation Language (DML)

- The DML Provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database.
- Here's a list of DML statements:
 - INSERT: Adds new data to the existing database table
 - UPDATE: Changes or updates values in the table
 - DELETE: Removes records or rows from the table
 - SELECT: Retrieves data from the table or multiple tables

DML (1): **INSERT** Command

- The **INSERT INTO** statement is used to insert new records in a table.
- INSERT INTO Syntax:
- 1. Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2,  
column3, ...)
```

```
VALUES (value1, value2, value3, ...);
```

DML (1): **INSERT** Command

- 2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query.
- However, make sure the order of the values is in the same order as the columns in the table.
- Here, the INSERT INTO syntax would be as follows:

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

DML (2): **UPDATE** Command

- The **UPDATE** statement is used to modify the existing records in a table.
- **UPDATE** Syntax:
- **UPDATE table_name SET column1=value1, column2 = value2, ... WHERE condition;**

DML (3): **DELETE** Command

- The **DELETE** statement is used to delete existing records in a table.
- **DELETE** Syntax:

```
DELETE FROM table_name WHERE  
condition;
```

Specifying Constraints in SQL

- SQL constraints are used to specify rules for data in a table.
- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.
- Syntax

```
CREATE TABLE table_name (column1 datatype  
constraint,      column2 datatype  
constraint,...);
```

Specifying Constraints in SQL

- The following constraints are commonly used in SQL:
- **NOT NULL**: Ensures that a column cannot have a NULL value
- **UNIQUE**: Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and **UNIQUE**. Uniquely identifies each row in a table
- **FOREIGN KEY** - Prevents actions that would destroy links between tables
- **CHECK** - Ensures that the values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column if no value is specified

Basic Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database: the **SELECT** statement.
- **SELECT Syntax**
- **SELECT** *column1, column2, ...* **FROM** *table_name*;
- Here, *column1, column2, ...* are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:
- **SELECT** * **FROM** *table_name*;

Basic Retrieval Queries in SQL

- The SQL **SELECT DISTINCT** Statement
- The **SELECT DISTINCT** statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
- **SELECT DISTINCT Syntax**
- **SELECT DISTINCT column1, column2, ...FROM table_name;**

Basic Retrieval Queries in SQL

- The **WHERE** clause is used to filter records.
- It is used to extract only those records that fulfill a specified condition.
- **WHERE Syntax**
- **SELECT column1, column2, ...FROM table_name
WHERE condition;**

Basic Retrieval Queries in SQL

- **AND, OR and NOT Operators in SQL**

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition:
- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.

Basic Retrieval Queries in SQL

- The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...FROM table_name  
WHERE condition1 AND condition2 AND  
condition3 ...;
```

• OR Syntax

```
• SELECT column1, column2, ...FROM table_name  
WHERE condition1 OR condition2 OR  
condition3 ...;
```

Basic Retrieval Queries in SQL

- NOT Syntax
- `SELECT column1, column2, ...FROM table_name
WHERE NOT condition;`

Basic Retrieval Queries in SQL

- **The SQL ORDER BY Keyword**
- The **ORDER BY** keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default.
- To sort the records in descending order, use the **DESC** keyword.
- **ORDER BY Syntax**

```
SELECT column1, column2, ...FROM table_name  
ORDER BY column1, column2, ... ASC|DESC;
```

Basic Retrieval Queries in SQL

- **The SQL GROUP BY Statement**
- The **GROUP BY** statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
- The **GROUP BY** statement is often used with aggregate functions (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**) to group the result-set by one or more columns.
- **GROUP BY Syntax**
- **SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) ORDER BY column_name(s);**

Basic Retrieval Queries in SQL

- **Aggregates (count, Min, max, avg, sdev)**

Basic Retrieval Queries in SQL

- **The SQL CASE WHEN....THEN Statement**
- ```
UPDATE EMPLOYEE SET Salary = CASE WHEN Dno
= 5 THEN Salary + 2000 WHEN Dno = 4 THEN
Salary + 1500 WHEN Dno = 1 THEN Salary + 3000
ELSE Salary + 0 ;
```

# EXERCISE

---

- Describe the syntaxes and uses of **HAVING**, **UNION**, **JOIN** in SQL