SI Course 7 - SQL Part 1

1 - Introduction

SQL (Structured Query Language) is a database language designed to manage data stored in a relational database management system.

Many current relational database management systems, such as Oracle Database, Microsoft SQL Server, MySQL, IBM DB2, IBM Informix, and Microsoft Access, use SQL.

In a database management system (DBMS), SQL is used to:

- Create the database and table structure.
- Perform basic database management tasks (adding, deleting, and modifying).
- Execute complex queries to transform raw data into useful information.

SQL can be used to:

- Create the database and table structure => use it as a Data Definition Language (DDL).
- Delete, select, and update data in database tables => use it as a Data Manipulation Language (DML).
- The main SQL DDL (Data Definition Language) statements are:
  - CREATE DATABASE
  - CREATE TABLE
  - DROP TABLE
  - ALTER TABLE
- Most SQL queries will be compatible with the Microsoft SQL Server DBMS.

2 - Create a database

Example:
`CREATE DATABASE SW;`

A new database named SW is created using the SQL statement `CREATE DATABASE SW;`.
Once the database is created, the next step is to create the database tables.

3 - Create a Table

The general format of the CREATE TABLE command is as follows:

```
CREATE TABLE <table_name> (
ColumnName1 DataType1 OptionalConstraint1,
ColumnName2 DataType2 OptionalConstraint2,
...
OptionalTableConstraints
);
```

- `<table_name>` is the name of the database table, for example, `Employee`.
- Each field in CREATE TABLE has three parts: ColumnName, DataType, and OptionalConstraint.

3-1 Data Types

The data type, as described below, must be either a system data type or a user-defined data type. Many data types have a specified size, such as CHAR(35) or Numeric(8,2).

- **bit:** Integer data with a value of 1 or 0.
- **int:** Integer data (whole number) from -2^31 (-2,147,483,648) to 2^31 – 1 (2,147,483,647).
- **smallint:** Integer data from 2^15 (-32,768) to 2^15 – 1 (32,767).
- **tinyint:** Integer data from 0 to 255.
- **decimal:** Fixed-precision numeric data and scale, from -10^38 -1 to 10^38.
- **numeric:** Synonym for decimal.
- **timestamp:** Unique number within the entire database.
- **uniqueidentifier:** Globally unique identifier (GUID).
- **money:** Monetary values from -2^63 (-922,337,203,685,477,5808) to 2^63 - 1 (+922,337,203,685,477,5807), with precision to the ten-thousandth of a monetary unit.
- **smallmoney:** Monetary values from -214,748,3648 to +214,748,3647, with an accuracy of one ten-thousandth of a monetary unit.
- **float:** Floating-point numeric data from -1.79E + 308 to 1.79E + 308.
- **real:** Floating-point numeric data from -3.40E + 38 to 3.40E + 38.
- **datetime:** Date and time data from January 1, 1753, to December 31, 9999, with an accuracy of one hundred seconds or 3.33 milliseconds.
- **date:** Date, from 0001-01-01 to 9999-12-31
- **time:** Time, from 00:00:00.0000000 through 23:59:59.9999999
- **smalldatetime:** Date and time data from January 1, 1900 to June 6, 2079, with one-minute precision.
- **char:** Fixed-length non-Unicode character data with a maximum length of 8,000 characters.
- **varchar:** Non-Unicode variable character data with a maximum length of 8,000 characters.
- **text:** Non-Unicode variable character data with a maximum length of 2^31 – 1 (2,147,483,647) characters.
- **binary:** Fixed-length binary data with a maximum length of 8,000 bytes.
- **varbinary:** Variable-length binary data with a maximum length of 8,000 bytes.

- **image:** Variable-length binary data with a maximum length of 2^31 – 1 (2,147,483,647) bytes.

3-2 Column Constraints

The column constraints are: NULL, NOT NULL, UNIQUE, PRIMARY KEY, IDENTITY, and DEFAULT.
The NULL constraint indicates that NULL values are allowed.

The NOT NULL constraint indicates that a value must be provided when a new row is created.

Example:

```
USE SW
CREATE TABLE EMPLOYEES
(
EmployeeNo CHAR(10) NOT NULL PRIMARY KEY,
DepartmentName CHAR(30) NOT NULL DEFAULT 'Human Resources',
FirstName CHAR(25) NOT NULL,
LastName CHAR(25) NOT NULL UNIQUE,
BirthDate DATE NOT NULL,
);
```

3-2-1 IDENTITY Constraint

- We can use the optional IDENTITY column constraint to provide a unique, incremental value for this column.
- Identity columns are often used with PRIMARY KEY constraints to serve as a unique row identifier for the table.
- The IDENTITY property can be assigned to a column of tinyint, smallint, int, decimal, or numeric data type.
- This constraint:
  - Generates sequential numbers
  - Does not enforce entity integrity
  - Only one column can have the IDENTITY property
  - Must be defined as an integer, numeric, or decimal data type
  - Cannot update a column with the IDENTITY property
  - Cannot contain NULL values
  - Cannot bind default values and default constraints to the

For IDENTITY[(seed, increment)]:

- Seed - the initial value of the identity column

- Increment - the value to add to the last incremented column

We will use another example database to further illustrate SQL DDL statements by creating the tblHotel table in this HOTEL database.

```
USE HOTEL;
CREATE TABLE tblHotel (
HotelNo Int not NULL IDENTITY (1,1) PRIMARY KEY,
Name Char(50) NOT NULL,
Address Char(50) NULL,
City Char(25) NULL,
);
```

3-3 Table Constraints

Table constraints are identified by the keyword CONSTRAINT and can be used to implement various constraints such as:

- **UNIQUE constraint:** Prevents duplicate values from being entered in a column.
- **PRIMARY KEY constraint:** Defines a column, or a combination of columns, as the primary key.
- **FOREIGN KEY constraint:** Defines a column, or a combination of columns, whose values correspond to the primary key of another table.
- **CHECK constraint:** Restricts the values that can be entered into a table.

3-3-2 UNIQUE Constraint

- The UNIQUE constraint prevents duplicate values from being entered in a column.
- The PK and UNIQUE constraints are used to enforce entity integrity.
- Multiple UNIQUE constraints can be defined for a table.
- When a UNIQUE constraint is added to an existing table, the existing data is always validated.
- A UNIQUE constraint can be placed on columns that accept null values.
- Only one row can be NULL.
- A UNIQUE constraint automatically creates a unique index on the selected column.

General syntax for the UNIQUE constraint:

```
[CONSTRAINT constraint_name] UNIQUE [CLUSTERED | NONCLUSTERED] (col_name [,
col_name2 […, col_name16]]) [ON segment_name]
```

Example of using the UNIQUE constraint:

```
CREATE TABLE EMPLOYEES (
EmployeeNo CHAR(10) NOT NULL,
firstname char(50),
lastname char(50),
CONSTRAINT emp_unique UNIQUE(EmployeeNo,lastname)
);
```

3-3-3 Constraint PRIMARY KEY

```
USE SW
CREATE EMPLOYEES TABLE
(
EmployeeNo CHAR(10) NOT NULL UNIQUE,
DepartmentName CHAR(30) NOT NULL DEFAULT 'Human Resources',
FirstName CHAR(25) NOT NULL,
LastName CHAR(25) NOT NULL,
BirthDate DATE NOT NULL,
CONSTRAINT Employee_PK PRIMARY KEY(EmployeeNo,LastName)
);
```

```
USE SW
CREATE TABLE PROJECT
(
ProjectID Int NOT NULL IDENTITY (1000,100),
ProjectName Char(50) NOT NULL,
Department Char(35) NOT NULL,
MaxHours Numeric(8,2) NOT NULL DEFAULT 100,
StartDate DateTime NULL,
EndDate DateTime NULL,
CONSTRAINT ASSIGNMENT_PK PRIMARY KEY(ProjectI) ) );
```

3-3-4 FOREIGN KEY Constraint

- The FOREIGN KEY (FK) constraint defines a column or combination of columns whose values correspond to the PRIMARY KEY (PK) of another table.
- Values in a foreign key (FK) are automatically updated when the primary key (PK) values in the associated table are updated or changed.
- FK constraints must reference the PK or UNIQUE constraint of another table.
- The number of columns for the FK must be the same as for the PK or UNIQUE constraint.
- If the WITH NOCHECK option is used, the FK constraint will not validate existing data in a table.
- No indexes are created on the columns that participate in an FK constraint.

General syntax for the FOREIGN KEY constraint:

```
[CONSTRAINT constraint_name] [FOREIGN KEY (col_name [, col_name2 […,
col_name16]])] REFERENCES [owner.]ref_table [(ref_col [, ref_col2 […,
ref_col16]])]
```

- In this example, the HotelNo field in the tblRoom table is a foreign key to the HotelNo field in the tblHotel table shown previously.

```
USE HOTEL
CREATE TABLE tblRoom (
HotelNo Int NOT NULL IDENTITY(1,1),
RoomNo Int NOT NULL,
Type Char(50) NULL,
Price Float NULL,
CONSTRAINT pk PRIMARY KEY (HotelNo, RoomNo),
CONSTRAINT fk FOREIGN KEY (HotelNo) REFERENCES tblHotel(HotelNo)
);
```

3-3-5 CHECK Constraint

- The CHECK constraint restricts the values that can be entered into a table.
- It can contain search conditions similar to a WHERE clause.
- It can reference columns within the same table.
- The data validation rule for a CHECK constraint must evaluate to a Boolean expression.
- It can be defined for a column that has a related rule.

General syntax for the CHECK constraint:

```
[CONSTRAINT constraint_name] CHECK [NOT FOR REPLICATION] (expression)
```

In this example, the Type field is restricted to only the types 'Single', 'Double',

```
USE HOTEL;

CREATE TABLE tblRoom2 (
HotelNo Int NOT NULL,
RoomNo Int NOT NULL,
Type Char(50) NULL,
Price Float NULL,
CONSTRAINT prim_key PRIMARY KEY (HotelNo, RoomNo),
CONSTRAINT f_key FOREIGN KEY (HotelNo) REFERENCES tblHotel(HotelNo),
CONSTRAINT Valid_Type CHECK (Type IN ('Single', 'Double'))
);
```

In this second example, the employee's hire date must be before January 1, 2004, or have a quota limit of 300,000.

```
CREATE TABLE SALESREPS (
Empl_num Int Not Null
CHECK (Empl_num BETWEEN 101 AND 199),
Name Char (15),
Age Int CHECK (Age >= 21),
Quota Float CHECK (Quota >= 0.0),
HireDate DateTime,
CONSTRAINT QuotaCap CHECK ((HireDate < '2004-01-01') OR (Quota <=300000))
);
```

4- ALTER TABLE

- You can use ALTER TABLE statements to add and remove constraints.
- ALTER TABLE also allows you to add, remove, and modify columns.
- When a constraint is added, all existing data is checked for violations.

4-1- Adding a Constraint

In this example, we use the ALTER TABLE statement to add the IDENTITY property to a column called Name.

- We assume that the constraint "unqName" does not yet exist in the hotel database:

```
USE HOTEL;
ALTER TABLE tblHotel ADD CONSTRAINT unqName UNIQUE (Name);
```

- We assume that the constraint "df_city" does not yet exist in the hotel database. And that there is no **default** constraint defined on the "City" field of the "tblHotel" table.

```
USE HOTEL ALTER TABLE tblHotel
ADD CONSTRAINT df_city DEFAULT 'Vancouver' FOR City;
```

- We assume that the "pk_key" constraint does not yet exist in the hotel database. And that there is no **primary** constraint defined on the table.

```
USE HOTEL ALTER TABLE tblHotel
ADD CONSTRAINT pk_key PRIMARY KEY(HotelNo);
```

- We assume that the "fk_key" constraint does not yet exist in the hotel database.

```
USE HOTEL ALTER TABLE tblHotel
ADD CONSTRAINT fk_key foreing KEY(HotelNo) references category(id_category)
```

- We assume that the "ck_key" constraint does not yet exist in the hotel database.

```
USE HOTEL ALTER TABLE category
ADD CONSTRAINT ck_key check (id_category < 100 and id_category > 0)
```

- Add the NULL and NOT NULL column constraints:
- We assume that the id_category column already exists in the category table:

```
alter table category
alter column id_category int not null;
```

- We assume that the description column already exists in the category table, and that there is no primary key constraint on this column:

```
alter table category
alter column description char(25) null;
```

4-2- Removing a Constraint

- Assume the constraint "unqName" exists (in the tblHotel table of the Hotel database):

```
USE HOTEL;
ALTER TABLE tblHotel DROP CONSTRAINT unqName;
```

4-3- Adding a Column

- In the following example, we will use the ALTER TABLE statement to add the NumBeds column with the NOT NULL constraint to the tblRoom table.
- Assume the column "tblRoom" does not yet exist in the tblRoom table of the Hotel database:

```
USE hotel;
ALTER TABLE tblRoom
ADD NumBeds int NOT NULL;
```

4-4- Deleting a Column

- Assume that the "NumBeds" column exists (in the tblRoom table of the hotel database):

- Also assume that the NumBeds column is not involved in any constraints.

```
use hotel;
ALTER TABLE tblRoom
DROP Column NumBeds;
```

4-5- Changing the Type of a Column

- Assume the "BirthDate" column exists (in the Employees table of the sw database):

```
USE sw;
ALTER TABLE Employees
ALTER COLUMN BirthDate date;
```

5- Notes on Certain Column Constraints

- A column cannot have both a UNIQUE column constraint and a PRIMARY KEY column constraint at the same time (in the same row).
- The IDENTITY constraint can only be used with columns of type: int, bigint, smallint, tinyint, decimal, and numeric with s=0.
- The PRIMARY KEY column constraint can only be applied to a single column. To select multiple columns, you must use the table constraint (PRIMARY KEY).
- You cannot use both the PRIMARY KEY column and table constraints simultaneously.

6 - DROP TABLE

- The DROP TABLE command allows you to delete a table from the database.
- Make sure you have selected the correct database.

```
DROP TABLE employees;
```

Executing the SQL DROP TABLE statement above will delete the employees table from the database.

References

- Watt, Adrienne, and Nelson Eng. *Database design*. 2nd Edition. BCcampus, 2014
- Microsoft 2024, Data types (Transact-SQL), accessed November 4, 2024, https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16.