SQL Structured Query Language(SQL)

23 November 2024 22:05

Structured Query Language (SQL)
SQL is a standard language used to interact with relational databases. It is designed to perform tasks such as retrieving, updating, inserting and deleting data stored in a database. SQL is essential for managing data in relational database management system (RDBMS) such as MySQL PostGreSQL, SQL server, Oracle and SQLite.

Features of SQL
1. **Declarative Language:** Specify what data you need rather than how to retrieve it.
2. **Database Interaction:** Allows interaction with databases using CRUD operations:
   - i. **Create:** Insert data into tables.
   - ii. **Read:** Query data from tables.
   - iii. **Update:** Modify existing data.
   - iv. **Delete:** Remove data from tables.

3. **Standardized Language:** SQL follows ANSI (American national standard institute) standards, through specific implementation might have proprietary extensions.

4. **Data Definition and Manipulation:**
   - i. **Data Definition Language(DDL):** Define database structures(table, indexes, and views etc.)
   - ii. **Data Manipulation Language(DML):** Retrieve and manipulate data.
   - iii. **Data Control Language(DCL):** Managing access permissions.
   - iv. **Transaction Control Language:** whether the transaction is completed or not

SQL Data Definition Commands:

TABLE 7.1

SQL DATA DEFINITION COMMANDS

COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's database schema
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Validates data in an attribute
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows and columns from one or more tables (see Chapter 8, Advanced SQL)
ALTER TABLE	Modifies a table's definition (adds, modifies, or deletes attributes or constraints)
CREATE TABLE AS	Creates a new table based on a query in the user's database schema
DROP TABLE	Permanently deletes a table (and its data)
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

Data Manipulation Command:

TABLE 7.2

SQL DATA MANIPULATION COMMANDS

COMMAND OR OPTION	DESCRIPTION
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
UPDATE	Modifies an attribute's values in one or more table's rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to its original values
Comparison operators	
=, <, >, <=, >=, <>, !=	Used in conditional expressions
Logical operators	
AND/OR/NOT	Used in conditional expressions
Special operators	Used in conditional expressions
BETWEEN	Checks whether an attribute value is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a value list
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
Aggregate functions	Used with SELECT to return mathematical summaries on columns
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

Database and Table

- **A database is a collection of tables or schemas and other objects.**
- **A table or relation consist of rows and columns. Rows represent records or tuples and columns represents attributes(Fields).**

For example: A table of Employee

ID	Name	Department	Salary
101	Alice	HR	50000
102	Bob	IT	70000
103	Charlie	Finance	60000

Data types

SQL supports various data types to define the kind of data stored in a column.

Category	Data Types
Numeric	INT, FLOAT,DECIMAL,BIGINT
String	VARCHAR, CHAR, TEXT
Date and time	DATE, DATETIME,TIMESTAMP,TIME
Binary	BLOB, BYTEA
Boolean	BOOLEAN

Data dictionary means how we store data in the database.

TABLE 7.3

DATA DICTIONARY FOR THE CH07_SALECO DATABASE

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
PRODUCT	P_CODE	Product code	VARCHAR(10)	XXXXXXXXXX	NA	Y	PK	
	P_DESCRIPT	Product description	VARCHAR(35)	Xxxxxxxxxxxx	NA	Y		
	P_INDATE	Stocking date	DATE	DD-MON-YYYY	NA	Y		
	P_QOH	Units available	SMALLINT	####	0–9999	Y		
	P_MIN	Minimum units	SMALLINT	####	0–9999	Y		
	P_PRICE	Product price	NUMBER(8,2)	####.##	0.00–9999.00	Y		
	P_DISCOUNT	Discount rate	NUMBER(5,2)	0.##	0.00–0.20	Y		
	V_CODE	Vendor code	INTEGER	###	100–999		FK	VENDOR
VENDOR	V_CODE	Vendor code	INTEGER	#####	1000–9999	Y	PK	
	V_NAME	Vendor name	VARCHAR(35)	Xxxxxxxxxxxxxx	NA	Y		
	V_CONTACT	Contact person	VARCHAR(25)	Xxxxxxxxxxxxxx	NA	Y		
	V_AREACODE	Area code	CHAR(3)	999	NA	Y		
	V_PHONE	Phone number	CHAR(8)	999–9999	NA	Y		
	V_STATE	State	CHAR(2)	XX	NA	Y		
	V_ORDER	Previous order	CHAR(1)	X	Y or N	Y		

FK = Foreign key
PK = Primary key
CHAR = Fixed-length character data, 1 to 255 characters
VARCHAR = Variable-length character data, 1 to 2,000 characters. VARCHAR is automatically converted to VARCHAR2 in Oracle.
NUMBER = Numeric data. NUMBER(9,2) is used to specify numbers that have two decimal places and are up to nine digits long, including the decimal places. Some RDBMSs permit the use of a MONEY or a CURRENCY data type.
NUMERIC = Numeric data. DBMSs that do not support the NUMBER data type typically use NUMERIC instead.
INT = Integer values only. INT is automatically converted to NUMBER in Oracle.
SMALLINT = Small integer values only. SMALLINT is automatically converted to NUMBER in Oracle.
DATE formats vary. Commonly accepted formats are DD-MON-YYYY, DD-MON-YY, MM/DD/YYYY, and MM/DD/YY.

*Not all the ranges shown here will be illustrated in this chapter. However, you can use these constraints to practice writing your own.

Create a database:

CREATE DATABASE companyDb;

```
mysql> create database companydb;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+------------------------+
| Database               |
+------------------------+
| carshowroom            |
| companydb              |
| information_schema     |
| menagerie              |
| mydb                   |
| mysql                  |
| performance_schema     |
| sakila                 |
| student_attendance2024 |
| student_attendence     |
| studentattendancesbgs  |
| sys                    |
| world                  |
+------------------------+
13 rows in set (0.00 sec)
```

To start using database created by you, give the following command:

```
mysql> use companydb;
Database changed
mysql>
```

Creating table in the database:

```
mysql> CREATE TABLE employee(
    -> ID INT PRIMARY KEY,
    -> NAME VARCHAR(50),
    -> DEPARTMENT VARCHAR(10),
    -> SALARY DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> SHOW TABLES;
+---------------------+
| Tables_in_companydb |
+---------------------+
| employee            |
+---------------------+
1 row in set (0.00 sec)

mysql>
```

Inserting rows in a table:

```
mysql> INSERT INTO EMPLOYEE(ID, NAME, DEPARTMENT, SALARY)
    -> VALUES(101, 'Srija', 'HR', 50000);
Query OK, 1 row affected (0.01 sec)
```

Retrieve data:

```
mysql> select * from employee;
+-----+-------+------------+----------+
| ID  | NAME  | DEPARTMENT | SALARY   |
+-----+-------+------------+----------+
| 101 | Srija | HR         | 50000.00 |
+-----+-------+------------+----------+
1 row in set (0.00 sec)
```

Syntax of create table command:

```
CREATE TABLE tablename (
    column1        data type      [constraint] [,
    column2        data type      [constraint] ] [,
    PRIMARY KEY    (column1       [, column2]) ] [,
    FOREIGN KEY    (column1       [, column2]) REFERENCES tablename] [,
    CONSTRAINT     constraint ] );
```

For example:

```
CREATE TABLE VENDOR (
```

For example:

```
CREATE TABLE VENDOR (
V_CODE          INTEGER         NOT NULL     UNIQUE,
V_NAME          VARCHAR(35)     NOT NULL,
V_CONTACT       VARCHAR(25)     NOT NULL,
V_AREACODE      CHAR(3)         NOT NULL,
V_PHONE         CHAR(8)         NOT NULL,
V_STATE         CHAR(2)         NOT NULL,
V_ORDER         CHAR(1)         NOT NULL,
PRIMARY KEY (V_CODE));
```

```
CREATE TABLE PRODUCT (
P_CODE          VARCHAR(10)     NOT NULL        UNIQUE,
P_DESCRIPT      VARCHAR(35)     NOT NULL,
P_INDATE        DATE            NOT NULL,
P_QOH           SMALLINT        NOT NULL,
P_MIN           SMALLINT        NOT NULL,
P_PRICE         NUMBER(8,2)     NOT NULL,
P_DISCOUNT      NUMBER(5,2)     NOT NULL,
V_CODE          INTEGER,
PRIMARY KEY (P_CODE),
FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE);
```

```
CREATE TABLE CUSTOMER (
CUS_CODE        NUMBER          PRIMARY KEY,
CUS_LNAME       VARCHAR(15)     NOT NULL,
CUS_FNAME       VARCHAR(15)     NOT NULL,
CUS_INITIAL     CHAR(1),
CUS_AREACODE    CHAR(3)         DEFAULT '615'     NOT NULL
                                CHECK(CUS_AREACODE IN
                                ('615','713','931')),
CUS_PHONE       CHAR(8)         NOT NULL,
CUS_BALANCE     NUMBER(9,2)     DEFAULT 0.00,
CONSTRAINT CUS_UI1 UNIQUE (CUS_LNAME, CUS_FNAME));
```

```
CREATE TABLE LINE (
INV_NUMBER      NUMBER          NOT NULL,
LINE_NUMBER     NUMBER(2,0)     NOT NULL,
P_CODE          VARCHAR(10)     NOT NULL,
LINE_UNITS      NUMBER(9,2)     DEFAULT 0.00      NOT NULL,
LINE_PRICE      NUMBER(9,2)     DEFAULT 0.00      NOT NULL,
PRIMARY KEY (INV_NUMBER, LINE_NUMBER),
FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE ON DELETE CASCADE,
FOREIGN KEY (P_CODE) REFERENCES PRODUCT(P_CODE),
CONSTRAINT LINE_UI1 UNIQUE(INV_NUMBER, P_CODE));
```

SQL Indexes:

Indexes can be used to improve the efficiency of search and to avoid duplicate column values. In fact, when you declare a primary key, the dbms automatically creates a unique index. Even with this feature, you often need additional indexes. the ability to create indexes quickly and efficiently is important. You. using the create index command, SQL indexes can be created on the basis of any selected attribute. The syntax

is:

```
CREATE [UNIQUE] INDEX indexname ON tablename(column1[,column2])
```

For example:

```
CREATE INDEX P_INDATEX ON PRODUCT(P_INDATE);

CREATE UNIQUE INDEX P_INDATEX ON PRODUCT(P_INDATE);
```

← Use keyword UNIQUE, when you have selected an attribute that may contain duplicate data.

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)
```

Command used to save all the transaction or query executed by SQL engine.