CM1603 - Database Systems

Lecture 8| Introduction to SQL

Dileeka Alwis – Senior Lecturer Grade II / Level Coordinator,

Department of Computing, IIT

ROBERT GORDON UNIVERSITY ABERDEEN
TEF Gold
INFORMATICS INSTITUTE OF TECHNOLOGY

Learning Outcomes

- Covers LO3 for Module - Use SQL as a data definition and data manipulation language, and to query a relational database.

- Partially covers LO4 for Module – Implement and test a relational database using a query language with a suitable interface.

- On completion of this lecture, students are expected to be able to:
  - Create DDl and DML statements
  - Use mySql to create databases and manipulate data

Lesson Outline

- Introduction to SQL
  - SQL Statements
  - SQL Commands
- Data Definition Language
  - Create Databases
  - Create Tables
- SQL Data Types
- SQL Constraints
- Data Manipulation Language
  - CRUD Operations (Insert, Retrieve, Update, Delete)

Structured Query Language (SQL)

- The most widely used and the standard database query language for storing and managing data in Relational DBMS.
  - Not a programming language.
  - A language used to communicate with a database.
- The ANSI standard language for the definition and manipulation of relational database.
- The first commercial language introduced for E.F Codd's Relational model of database.
- Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language.

Structured Query Language (SQL)

- SQL is a **set-oriented language**
  - Can query *multiple rows* from one or more tables using just one statement (in contrast to record-oriented language, that processes one record at a time)

- SQL is **non-procedural**
  - Statements/Commands describe *what a user wants*, rather than how a task is accomplished step by step
  - Opposite of imperative programming languages, where statements are used to accomplish a task step by step

What can SQL do?

- Create new databases
- Create new database tables
- Insert records into tables
- Retrieve data from tables
- Update records in tables
- Delete records from tables
- Set permissions on database objects
- Create and execute other database objects views, functions, stored procedures etc.

SQL Statements

- The actions need to be performed on a database are done with SQL statements.
- It contain instructions to be executed in order to create and manipulate databases.
- SQL Statements are not case sensitive, but the SQL data is case-sensitive.
- Instructions are given in the form of statements, consisting of a specific SQL **command** and additional parameters which apply to that statement.
- Some database systems require a **semicolon** (;) at the end of each SQL statement.
  - Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

SQL Commands

- Commonly used SQL commands are grouped into the following categories:
  - Data Definition Language (**DDL**) : Used to define or change the database structure.
  - Data Manipulation Language (**DML**) : Used to manipulate the data stored in the database.
  - Data Control Language (**DCL**): Used to specify access controls to database elements and data.
  - Transaction Control Language (**TCL**): Used to manage transactions in the database.

Common SQL Commands

Data Retrieval	SELECT
Data Manipulation Language (DML)	INSERT UPDATE DELETE
Data Definition Language (DDL)	CREATE DATABASE/TABLE ALTER DATABASE/TABLE DROP DATABASE/TABLE RENAME DATABASE/TABLE TRUNCATE TABLE
Transaction Control	BEGIN/SAVE TRANSACTION COMMIT ROLLBACK
Data Control Language (DCL)	CREATE/ALTER/DROP USER GRANT REVOKE

Create & Remove Databases

- **Create a new DB**

  CREATE DATABASE <Database_Name> ;

Eg: CREATE DATABASE UniversityDB; (Create a database to save details about the university activities)

- **Remove a new DB**

  DROP DATABASE <Database_Name> ;

Eg: DROP DATABASE UniversityDB; (Delete the UniversityDB that you have created earlier)

Note: *Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!*

Create a new Table in a DB

```
CREATE TABLE  Table_Name (
    field_name1   data_type(field_size),
    field_name2   data_type(field_size),
          ....
   );
```

- Datatype specifies the type of data the field can hold (e.g. varchar, int, date, etc.).
- Field Size specifies the maximum number of characters need to be saved in the field.
- No need to specify the field sizes for INT, MONEY and DATE data types since it get the default.

SQL Data Types

Numeric types	integer	`integer, int, smallint, long`
	floating point	`float, real, double precision`
	formatted	`decimal(i,j), dec(i,j)`
Character-string types	fixed length	`char(n), character(n)`
	varying length	`varchar(n), char varying(n), character varying(n)`
Bit-string types	fixed length	`bit(n)`
	varying length	`bit varying(n)`
Date and time types		`date, time, datetime, timestamp, time with time zone, interval`
Large types	character	`long varchar(n), clob, text`
	binary	`blob`

Exercise

- Create the following **Student** table in the UniversityDB to save details about students. Use suitable data types and field sizes.

StudentID	FirstName	Surname	DOB	Gender
1001	Kate	West	12/10/1994	F
1002	Julie	McLain	3/7/1995	F
1003	Tom	Smith	24/12/1994	M
1004	Mark	Foster	5/11/1996	M
1005	Jane	Knight	17/6/1995	F
1006	Matt	Smith	24/12/1995	M
1007	Karen	Edwards	3/7/1995	M
1008	John	Williams	15/11/1996	M
1009	Allison	Cambell	10/10/1994	M
1010	Shirley	Thomas	15/4/1995	F

Answer

```
CREATE TABLE Student (
        StudentID INT,
        FirstName VARCHAR(30),
        Surname VARCHAR(50),
        DOB DATE,
        Gender CHAR(1)
) ;
```

SQL Constraints

- Used to specify validation rules for the data in a table.
- Used to limit or validate the type of data that is entered to a table.
- Ensures the accuracy and reliability of the data in the table.
- Constraints can be specified when:
  - creating a table (inside the CREATE TABLE statement)
    or
  - after creating a table (inside the ALTER TABLE statement)

SQL Constraints

- **NOT NULL**
  - Indicate that a field cannot store NULL values.
  - A column defined as NOT NULL is a mandatory column.
- **UNIQUE**
  - Ensure that each record for a field must have a unique value.
- **DEFAULT**
  - Add a default value to all new records when no specific value is entered to that field.

SQL Constraints

- **CHECK**
  - Ensure that the value in a field meets a specific condition.
  - Used to limit the value range that can be placed in a column.

- **PRIMARY KEY**
  - A combination of a NOT NULL and UNIQUE.
  - Ensures that a field (or combination fields) has an unique identity which helps to find a particular record in a table more easily and quickly.

- **FOREIGN KEY**
  - Ensure the referential integrity of data in one table to match values in another table.

Exercise

- Create the **Student** table with the following specifications.
  - StudentID: Primary Key
  - Firstname: Cannot be NULL
  - Surname: Cannot be NULL
  - Gender: Default value ‘M’
  - NIC: Values must be Unique
  - Age: Values must be higher than or equal to 18
  - Program: Values must be equal to the values in the ProgramCode filed of the Program table or can be Null.

SQL Constraints Example 1

```
CREATE TABLE Student (
    StudentID INT,
    FirstName VARCHAR(30) NOT NULL,
    Surname VARCHAR(50) NOT NULL,
    NIC varchar(12),
    Age int,
    Gender CHAR(1) DEFAULT 'M',
    Program CHAR(3)
    CHECK (Age>=18),
    UNIQUE (NIC),
    PRIMARY KEY (StudentID),
    FOREIGN KEY (Program) REFERENCES Program(ProgramCode)
) ;
```

Exercise

- Create the **Marks** table with the following specifications.
  - StudentID: + ModuleCode: Primary Key
  - StudentID : Values must be equal to the values in the StudentID filed of the Student table or can be Null.
  - ModuleCode : Values must be equal to the values in the ModuleCode filed of the Module table or can be Null.

SQL Constraints Example 2

```
CREATE TABLE Marks
(
   StudentID int,
   ModuleCode varchar(3),
   Marks int,
   PRIMARY KEY (StudentID, ModuleID),
   FOREIGN KEY (StudentID) REFERENCES Student (StudentID),
   FOREIGN KEY (ModuleCode) REFERENCES Module (ModuleCode)
);
```

CRUD Operations

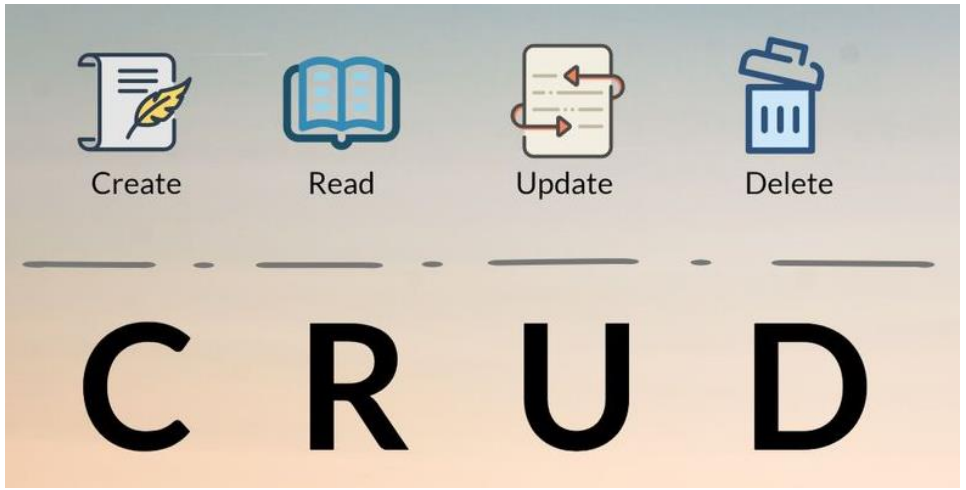
- The four basic operations that a database can perform

  - Create data: **INSERT**

  - Retrieve data: **SELECT**

  - Update data: **UPDATE**

  - Delete data: **DELETE**



Retrieving Records

- Retrieve all the records in a table with all the fields:

  SELECT *

  FROM <Table_Name**>;**

- Select specific fields in a table with all the records:

  SELECT <Field_Names>

  FROM <Table_Name>;

- Eg: Display all the details of students:

  SELECT *

  FROM Student;

- Eg: Display names of the students:

  SELECT FirstName, Surname

  FROM Student;

Insert a Record to a Table

INSERT INTO <Table_Name> **(**<Field_Names>**)**

VALUES (<Values to be inserted>**)**;

- When adding values to all the fields of the relation in the given order:

Eg: INSERT INTO Student
VALUES (1001, ‘Kate’, ‘West’, ‘1994-10-12’, ‘F’);

INSERT INTO Student (StudentID, FirstName, Surname DOB, Gender)
VALUES (1001, ‘Kate’, ‘West’, ‘1994-10-12’, ‘F’);

INSERT INTO Student
VALUES (1001, ‘Kate’, ‘West’, **NULL**, **NULL**);

Insert a Record to a Table

- When adding values ONLY to selected fields or when changing the order of the fields:

Eg: INSERT INTO Student (StudentID, FirstName, Surname)
VALUES (1001, 'Kate', 'West');

INSERT INTO Student (FirstName, StudentID, DOB, Surname)
VALUES ('Kate', 1001, '1994-10-12', 'West');

Update a Record

UPDATE <Table_Name >
SET <Field_Name> = <New_Attribute_Value>
WHERE <Field_Name> = <Searching_Atribute_Value>

Eg: Change the Kate's surname to Robbinson.

```
UPDATE Student
SET Surname = 'Robbinson'
WHERE StudentID = 1001;
```

- Note: Be careful when updating records in a table!
  Notice the **WHERE** clause in the UPDATE statement.
  The **WHERE** clause specifies which record(s) to be updated.
  If you omit the WHERE clause, all records in the table will be updated!

Delete a Record

DELETE FROM <Table_Name >

WHERE <Field_Name> = <Searching_Attribute_Value>

- Eg: Delete Kate's record

```
DELETE FROM Student
WHERE StudentID = 1001
```

- **Note:** Be careful when deleting records in a table!
  Notice the **WHERE** clause in the DELETE statement.
  The WHERE clause specifies which record(s) to be deleted.
  If you omit the WHERE clause, all records in the table will be deleted!

Thank you

Contact: dileeka.a@iit.ac.lk