Query Language

DDL (Data Definition Language)

is a part of SQL, we use and modify the structure of databases. The basic DDL directives include several functions, here is an explanation of some of them:

CREATE

This command is used to create a new database or create tables, or other objects (objects) such as view and index in the database.

ALTER

This command is used to modify the structure of an existing object in the database, such as adding a new column, or modifying the data type of an existing column.

DROP

This command is used to delete a database, table, or specific object from the database. It should be used with caution because it deletes the entire data.

TRUNCATE

Used to delete all data from a table but without deleting the table itself or its structure. It is faster than DELETE statement.

RENAME

This command is used to change the name of a table or object in the database.

COMMENT

This command is used to add comments or explanations to database objects, such as tables or columns, so that they are more understandable to other developers or users.

GRANT

This command is used to grant users specific permissions on specific tables or objects in the database, such as granting SELECT or INSERT permissions.

REVOKE

This command is used to revoke permissions from users that were granted using the GRANT command.

ANALYZE

This command collects statistics about the distribution of data within a specific table, and is often used to improve query performance by optimizing query plans.

```sql
-- إنشاء جدول Faculty
CREATE TABLE Faculty (
    FacultyID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Department VARCHAR(100),
    Email VARCHAR(100)
);

-- إنشاء جدول Students
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    Email VARCHAR(100),
    Major VARCHAR(100)
);

-- إنشاء جدول Courses
CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100),
    Credits INT,
    FacultyID INT,
    FOREIGN KEY (FacultyID) REFERENCES Faculty(FacultyID)
);

-- إنشاء جدول Enrollments
CREATE TABLE Enrollments (
    EnrollmentID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    EnrollmentDate DATE,
    Grade CHAR(2),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

```sql
--إضافة عمود Phone إلى جدول Faculty
ALTER TABLE Faculty
ADD Phone VARCHAR(15);

-- تغيير نوع البيانات لعمود Grade في جدول Enrollments
ALTER TABLE Enrollments
MODIFY Grade VARCHAR(5);


-- حذف جدول Enrollments
DROP TABLE Enrollments;

-- حذف قاعدة البيانات بأكملها
DROP DATABASE University;


-- بدون حذف الجدول نفسه Students حذف جميع البيانات من جدول
TRUNCATE TABLE Students;

-- تغيير اسم جدول Faculty إلى Professors
RENAME TABLE Faculty TO Professors;

-- إضافة تعليق لجدول Courses
COMMENT ON TABLE Courses IS 'Table that stores all course details';

-- إضافة تعليق على عمود CourseName في جدول Courses
COMMENT ON COLUMN Courses.CourseName IS 'The name of the course';

-- منح المستخدم user_name صلاحيات SELECT و INSERT على جدول Students
GRANT SELECT, INSERT ON Students TO user_name;


-- سحب صلاحية INSERT من المستخدم user_name على جدول Students
REVOKE INSERT ON Students FROM user_name;

-- تحليل جدول Courses لتحسين الأداء
ANALYZE TABLE Courses;
```

DML (Data Manipulation Language)

commands in English, along with their purposes in the database. **DML** is used to interact with the actual data in the tables within a database, unlike **DDL**, which focuses on defining the structure. Here are the main **DML** commands:

INSERT
Used to add new records to a table.

UPDATE
Used to modify existing data in a table. You can update all records or specify conditions to update specific records.

DELETE
Used to delete records from a table. You can delete all records or specific ones by specifying a condition.

SELECT
Used to retrieve data from one or more tables. You can select all columns or specific ones, add conditions with **WHERE**, sort results with **ORDER BY**, or group them with **GROUP BY**.

MERGE
Used to perform an "upsert" operation—insert new records and update existing ones in a single statement. This is useful for synchronizing data.

CALL
Used to execute a stored procedure within the database, which is often used for complex operations that involve multiple DML steps.

LOCK TABLE
Used to lock a table to prevent other users from modifying it while the lock is active, which is important for maintaining data consistency.

```sql
-- إضافة عضو هيئة تدريس جديد إلى جدول Faculty
INSERT INTO Faculty (FacultyID, FirstName, LastName, Department, Email)
VALUES (4, 'Noor', 'Ahmed', 'Biology', 'noor.ahmed@university.edu');

-- إضافة طالب جديد إلى جدول Students
INSERT INTO Students (StudentID, FirstName, LastName, DateOfBirth, Email, Major)
VALUES (104, 'Khaled', 'Mansour', '2001-08-10', 'khaled.mansour@student.edu', 'Biology');

-- تعديل قسم عضو هيئة التدريس Noor ليصبح 'Chemistry'
UPDATE Faculty
SET Department = 'Chemistry'
WHERE FacultyID = 4;

-- تعديل التخصص للطالب خالد
UPDATE Students
SET Major = 'Environmental Science'
WHERE StudentID = 104;

-- حذف سجل الطالب الذي رقمه 104 من جدول Students
DELETE FROM Students
WHERE StudentID = 104;

-- حذف جميع البيانات من جدول Enrollments
DELETE FROM Enrollments;

-- استرجاع جميع البيانات من جدول Faculty
SELECT * FROM Faculty;

-- استرجاع الأسماء فقط للطلاب الذين يدرسون في تخصص Computer Science
SELECT FirstName, LastName
FROM Students
WHERE Major = 'Computer Science';

-- استرجاع البيانات لجميع الدورات مع الترتيب حسب عدد الساعات
SELECT CourseName, Credits
FROM Courses
ORDER BY Credits DESC;
```

```sql
-- استرجاع عدد الطلاب في كل تخصص
SELECT Major, COUNT(StudentID) AS TotalStudents
FROM Students
GROUP BY Major;


MERGE INTO Courses AS Target
USING (SELECT 301 AS CourseID, 'Data Structures' AS CourseName, 3 AS Credits, 1 AS FacultyID) AS Source
ON (Target.CourseID = Source.CourseID)
WHEN MATCHED THEN
    UPDATE SET Target.CourseName = Source.CourseName, Target.Credits = Source.Credits
WHEN NOT MATCHED THEN
    INSERT (CourseID, CourseName, Credits, FacultyID)
    VALUES (Source.CourseID, Source.CourseName, Source.Credits, Source.FacultyID);


-- نفترض أن هناك إجراءً مخزنًا يسمى `EnrollStudent`
CALL EnrollStudent(101, 301, '2024-02-15'); -- هذا يستدعي الإجراء مع المعلمات المطلوبة

-- قفل جدول Students للتأكد من عدم تعديله بواسطة مستخدمين آخرين
LOCK TABLE Students IN EXCLUSIVE MODE;

-- قم بإجراء العمليات المطلوبة هنا بينما الجدول مغلق
-- لا تنسَ فك القفل لاحقًا حسب النظام المستخدم
```

DCL (Data Control Language)

commands in English, along with their roles in database security and access management. **DCL** is crucial for controlling permissions and defining who can access or manipulate data within the database. The main **DCL** commands are:

GRANT

Used to give permissions or privileges to users or roles to perform specific operations on the database or specific objects, such as **SELECT**, **INSERT**, **UPDATE**, and **DELETE**.

REVOKE

Used to remove permissions granted to users or roles via the **GRANT** command. This is essential for managing access control to data and ensuring only authorized actions are allowed.

DENY

(Not available in all database management systems) This command explicitly restricts a user or role from accessing specific objects, even if they have permissions from other sources.

```
-- منح المستخدم "user_name" صلاحية SELECT و INSERT على جدول Students
GRANT SELECT, INSERT ON Students TO user_name;

-- منح المستخدم "admin_user" جميع الصلاحيات (SELECT, INSERT, UPDATE, DELETE) على جدول Courses
GRANT ALL PRIVILEGES ON Courses TO admin_user;

-- منح مجموعة من الصلاحيات لمجموعة مستخدمين
GRANT SELECT, UPDATE ON Faculty TO role_name;

-- سحب صلاحية INSERT من المستخدم "user_name" على جدول Students
REVOKE INSERT ON Students FROM user_name;

-- سحب جميع الصلاحيات من المستخدم "admin_user" على جدول Courses
REVOKE ALL PRIVILEGES ON Courses FROM admin_user;

-- تقييد المستخدم "restricted_user" من الوصول إلى جدول Faculty، حتى إذا كان لديه صلاحيات أخرى
DENY SELECT ON Faculty TO restricted_user;

-- تقييد مجموعة من المستخدمين من تنفيذ أمر DELETE على جدول Enrollments
DENY DELETE ON Enrollments TO restricted_role;
```

DQL (Data Query Language)

is a subset of SQL commands primarily used to retrieve data from a database. The core function of **DQL** is to allow users to query and retrieve information stored in database tables without modifying the data itself. The primary **DQL** command is **SELECT**, but it includes several clauses and keywords that enable complex data querying and filtering.

SELECT

The core command in DQL, used to retrieve data from one or more tables. You can select all columns or specific ones, apply filters, sorting, grouping, and more.

FROM

Specifies the table(s) from which to retrieve data. It works in conjunction with **SELECT**.

- **WHERE**
  Used to filter records based on specified conditions, allowing more precise data retrieval.
- **ORDER BY**
  Sorts the result set by one or more columns, either in ascending (**ASC**) or descending (**DESC**) order.

GROUP BY

Groups rows that have the same values in specified columns into summary rows, often used with aggregate functions like **SUM**, **COUNT**, **AVG**, etc.

- **HAVING**
  Used with **GROUP BY** to filter groups based on conditions, as **WHERE** cannot be used with aggregated data.

JOIN

Used to combine rows from two or more tables based on a related column. Common types include **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, and **FULL JOIN**.

DISTINCT

Removes duplicate records from the result set, ensuring each row is unique.

- **LIMIT / TOP**
  Restricts the number of rows returned by the query. **LIMIT** is often used in MySQL, while **TOP** is used in SQL Server.
- **UNION / UNION ALL**
  Combines the results of two or more **SELECT** statements. **UNION** removes duplicates, while **UNION ALL** includes them.

SUBQUERIES

A query nested inside another query, often used to filter data based on results from another **SELECT** statement.

```sql
-- استرجاع جميع الأعمدة من جدول Students
SELECT * FROM Students;

-- استرجاع عمود FirstName و LastName فقط من جدول Faculty
SELECT FirstName, LastName FROM Faculty;

-- استرجاع الطلاب الذين يدرسون تخصص Computer Science
SELECT * FROM Students
WHERE Major = 'Computer Science';

-- استرجاع الدورات التي عدد ساعاتها أكثر من 3
SELECT * FROM Courses
WHERE Credits > 3;

-- استرجاع جميع الطلاب مع الترتيب حسب تاريخ الميلاد من الأحدث إلى الأقدم
SELECT * FROM Students
ORDER BY DateOfBirth DESC;

-- استرجاع الدورات مع الترتيب حسب الاسم تصاعديًا
SELECT * FROM Courses
ORDER BY CourseName ASC;

-- استرجاع عدد الطلاب في كل تخصص
SELECT Major, COUNT(StudentID) AS TotalStudents
FROM Students
GROUP BY Major;

-- استرجاع التخصصات التي يوجد بها أكثر من 2 طلاب فقط
SELECT Major, COUNT(StudentID) AS TotalStudents
FROM Students
GROUP BY Major
HAVING COUNT(StudentID) > 2;
```

```sql
-- استرجاع اسم الطالب واسم الدورة التي سجل بها باستخدام INNER JOIN
SELECT Students.FirstName, Students.LastName, Courses.CourseName
FROM Enrollments
INNER JOIN Students ON Enrollments.StudentID = Students.StudentID
INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID;

-- استرجاع جميع أعضاء هيئة التدريس وأي دورة يقومون بتدريسها حتى لو لم يكن هناك دورة (LEFT JOIN)
SELECT Faculty.FirstName, Faculty.LastName, Courses.CourseName
FROM Faculty
LEFT JOIN Courses ON Faculty.FacultyID = Courses.FacultyID;

-- استرجاع جميع التخصصات المختلفة للطلاب بدون تكرار
SELECT DISTINCT Major FROM Students;

-- استرجاع أول 5 طلاب فقط (في MySQL)
SELECT * FROM Students
LIMIT 5;

-- استرجاع أول 3 دورات فقط (في SQL Server)
SELECT TOP 3 * FROM Courses;

-- دمج نتائج تخصصات الطلاب والدورات باستخدام UNION لإزالة التكرارات
SELECT Major AS Field FROM Students
UNION
SELECT CourseName AS Field FROM Courses;

-- دمج نتائج تخصصات الطلاب والدورات مع الاحتفاظ بالتكرارات
SELECT Major AS Field FROM Students
UNION ALL
SELECT CourseName AS Field FROM Courses;

-- استرجاع الطلاب الذين مسجلين في دورة Data Structures
SELECT FirstName, LastName FROM Students
WHERE StudentID IN (
    SELECT StudentID FROM Enrollments
    WHERE CourseID = (SELECT CourseID FROM Courses WHERE CourseName = 'Data Structures')
);
```

TCL (Transaction Control Language)

commands in SQL are used to manage transactions in a database. These commands help control the changes made by **DML** statements and ensure data integrity within transactions. **TCL** commands are typically used to commit, rollback, or save transactions at specific points. Here are the main **TCL** commands:

COMMIT

Used to save all the changes made in the current transaction to the database permanently. Once committed, the changes cannot be rolled back.

ROLLBACK

Used to undo changes made in the current transaction. This command is helpful if an error occurs or if you need to cancel the transaction and return the data to its previous state.

- ## SAVEPOINT
  Creates a temporary savepoint within a transaction. You can roll back to this savepoint if needed, without affecting the entire transaction.

RELEASE SAVEPOINT

Deletes a previously defined savepoint. Once released, you cannot roll back to that savepoint.

- ## SET TRANSACTION
  Sets the properties for the current transaction, such as specifying whether it is read-only or setting the isolation level.

```
-- بدء عملية وإجراء تغييرات على البيانات
BEGIN;

-- تحديث بيانات طالب
UPDATE Students
SET Major = 'Physics'
WHERE StudentID = 101;

-- إدخال سجل جديد في جدول Enrollments
INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate, Grade)
VALUES (201, 101, 301, '2024-01-15', 'A');

-- حفظ التغييرات بشكل نهائي في قاعدة البيانات
COMMIT;

-- بدء عملية وإجراء تغييرات على البيانات
BEGIN;

-- إدخال سجل جديد في جدول Students
INSERT INTO Students (StudentID, FirstName, LastName, DateOfBirth, Email, Major)
VALUES (105, 'Sara', 'Youssef', '2002-06-15', 'sara.youssef@student.edu', 'Chemistry');

-- يحدث خطأ أو يتم اكتشاف مشكلة، لذلك نقرر التراجع عن التغييرات
ROLLBACK;
-- لا يتم إدخال أي بيانات لأن التراجع يلغي كل التغييرات في العملية
```

```
-- بدء عملية وإجراء تغييرات متعددة
BEGIN;

-- تحديث قسم عضو هيئة تدريس
UPDATE Faculty
SET Department = 'Physics'
WHERE FacultyID = 2;

-- إنشاء نقطة حفظ
SAVEPOINT Save1;

-- إجراء تغيير آخر
UPDATE Students
SET Major = 'Biology'
WHERE StudentID = 102;

-- التراجع إلى نقطة الحفظ
ROLLBACK TO Save1;
-- سيتم التراجع عن التغيير الثاني فقط، بينما يظل التغيير الأول
```

```
-- بدء عملية وإنشاء نقاط حفظ
BEGIN;

UPDATE Students
SET Major = 'Mathematics'
WHERE StudentID = 103;

SAVEPOINT Save1;

UPDATE Faculty
SET Department = 'Computer Science'
WHERE FacultyID = 3;

-- حذف نقطة الحفظ Save1
RELEASE SAVEPOINT Save1;

-- ستؤدي إلى خطأ لأنها حُذفت Save1 محاولة التراجع إلى
ROLLBACK TO Save1; -- هذا سيؤدي إلى خطأ
```

```
-- تعيين عملية بحيث تكون قراءة فقط
SET TRANSACTION READ ONLY;

-- استعلام قراءة بيانات الطلاب
SELECT * FROM Students;

-- عند محاولة إجراء عملية كتابة، ستحدث مشكلة لأن العملية محددة كقراءة فقط
UPDATE Students
SET Major = 'Engineering'
WHERE StudentID = 104; -- هذا سيؤدي إلى خطأ بسبب إعداد READ ONLY

-- تعيين مستوى العزل للمعاملة إلى SERIALIZABLE
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

-- إجراء الاستعلامات المطلوبة داخل العملية
```