National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Fall 2024, Lab Manual – 03

Course Code: CL-2005	**Course: Database Systems Lab**
Instructor(s):	**Sohail Ahmed Malik, Mr. Mubashir, Mr. Sameer Faisal, Ms. Mehak Mazhar, Ms, Fatima, Ms. Mahnoor Javed, Ms, Filza Akhlaq, Ms. Bushra Sattar, Ms. Syeda Ravia Ejaz, Ms. Yumna Asif**

Contents:

Constraint and its Types

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*, *column3 datatype constraint*, ...);

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table.

The following constraints are commonly used in SQL:

- **NOT NULL -** Ensures that a column cannot have a NULL value

  Example Syntax:
  1. CREATE TABLE Table1 (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column3 datatype NOT NULL, Column4 datatype);
  2. ALTER TABLE Table1 MODIFY Column_Name datatype NOT NULL;

- **UNIQUE -** Ensures that all values in a column are different.

  Example Syntax:
  1. CREATE TABLE Table1 (Column1 datatype NOT NULL UNIQUE, Column2 datatype NOT NULL, Column3 datatype, Column4 int);
  2. CREATE TABLE Table1 (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column3 datatype, Column4 datatype, CONSTRAINT constraint_name UNIQUE (column1, column2));
  3. ALTER TABLE table1 ADD UNIQUE (column1);
  4. ALTER TABLE table1 ADD CONSTRAINT const_N UNIQUE (column1, column2);
  5. ALTER TABLE table1 DROP CONSTRAINT constraint_name;

- **PRIMARY KEY -** A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.

  Example Syntax:
  1. CREATE TABLE Table1 (Column1 datatype NOT NULL PRIMARY KEY, Column2 datatype, Column3 datatype);

2. CREATE TABLE Table1 (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column3 datatype, PRIMARY KEY (Column1, Column2);
3. ALTER TABLE table1 ADD PRIMARY KEY (column1);
4. ALTER TABLE table1 ADD CONSTRAINT const_N PRIMARY KEY(column1, column2);
5. ALTER TABLE table1 DROP CONSTRAINT const_Name;

Note: If you use ALTER TABLE to add a primary key, the primary key column(s) must have been declared to not contain NULL values (when the table was first created).

- **FOREIGN KEY -** Prevents actions that would destroy links between tables. A **FOREIGN KEY** is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

Example Syntax:

1. CREATE TABLE Table1 (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column3 datatype, PRIMARY KEY (Column1), FOREIGN KEY (COLUMN NAME) REFERENCES Table2(ColumnName));

NOTE: While altering the column to add FK that named column need be present in the table which is later altered as FK.

2. ALTER TABLE Table1 ADD FOREIGN KEY (ColumnName) REFERENCES Table2 (ColumnName);
3. ALTER TABLE Table1 ADD CONSTRAINT CONSTRAIT_Name FOREIGN KEY (ColumnName) REFERENCES Table2 (ColumnName);
4. ALTER TABLE Orders DROP CONSTRAINT CONSTRAINT_Name;

- **CHECK -** Ensures that the values in a column satisfies a specific condition

Example Syntax:

1. CREATE TABLE Table1 (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column3 datatype, PRIMARY KEY (Column1), ColumnName datatype CHECK (column condition);
2. CREATE TABLE Table1 (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column3 datatype, PRIMARY KEY (Column1), CONSTRAINT CONSTRAINT_Name CHECK (column condition1 AND column condition2);
3. ALTER TABLE Table1 ADD CHECK (column condition);
4. ALTER TABLE Table1 ADD CONSTRAINT CONSTRAINT_Name CHECK (condition);
5. ALTER TABLE Table1 DROP CONSTRAINT_Name;

- **DEFAULT -** Sets a default value for a column if no value is specified

Example Syntax:

1. CREATE TABLE Table1 (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column3 datatype, PRIMARY KEY (Column1), ColumnName datatype DEFAULT any ANY_VALUE;
2. ALTER TABLE Table1 MODIFY columnName DEFAULT Value;
3. ALTER TABLE Table1 MODIFY columnName DEFAULT NULL;

- **CREATE INDEX -** Used to create and retrieve data from the database very quickly

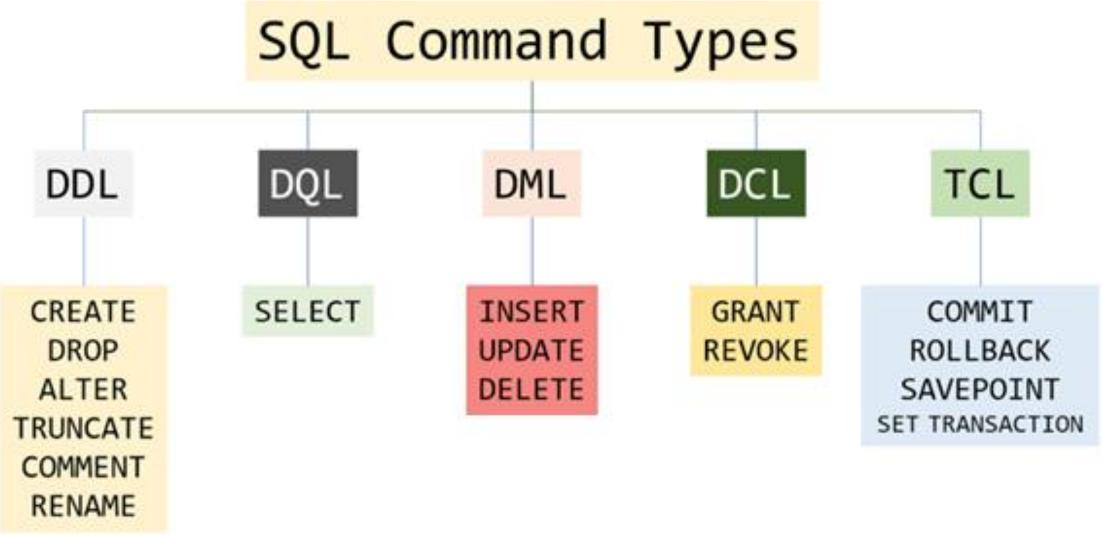
Example Syntax:

1. CREATE UNIQUE INDEX *index_name* ON *table_name* (*column1*, *column2*,..);
2. DROP INDEX *index_name*;

Deferred Constraint Checking (Chicken Egg Problem):
Detailed explanation on deferred constraint and chicken-egg problem was given in the below mention link:
Link: http://oracleexperiments.blogspot.com/2008/11/sometimes-it-is-very-important-to-defer.html

SQL Commands Categories:
Structured Query Language (SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also, we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks. These SQL commands are mainly categorized into following categories:

1. DDL – Data Definition Language
2. DQL – Data Query Language
3. DML – Data Manipulation Language
4. DCL – Data Control Language
5. TCL- TCL – Transaction Control Language



DML (Data Manipulation Language):
The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

- **INSERT** – is used to insert data into a table.
- **UPDATE** – is used to update existing data within a table.
- **DELETE** – is used to delete records from a database table.

1. Insert

The INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:

- **Only values:** 0
  First method is to specify only the value of data to be inserted without the column names.

Example Syntax:

```
INSERT INTO table_name VALUES (value1, value2, value3,…);
 table_name: name of the table.
 value1, value2..: value of first column, second column,… for the new record.
```

- **Column names and values both:**
  In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:

Example Syntax:

- INSERT INTO table_name (column1, column2, column3,..) VALUES (value1, value2, value3,..);

table_name: name of the table.

column1: name of first column, second column …

value1, value2, value3 : value of first column, second column,… for the new record.

2. Update

The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

Example Syntax:

UPDATE table_name SET column1 = value1, column2 = value2,...WHERE condition;

table_name: name of the table

column1: name of first , second, third column....

value1: new value for first, second, third column....

condition: condition to select the rows for which the values of columns needs to be updated.

3. Delete

The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

Example Syntax:

DELETE FROM table_name WHERE some_condition;

table_name: name of the table

some_condition: condition to choose particular record.

DDL (Data Definition Language):

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

- **CREATE -** to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- **ALTER -** alters the structure of the existing database
- **DROP -** delete objects from the database
- **TRUNCATE -** remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT -** add comments to the data dictionary
- **RENAME -** rename an object

1. Create

There are two CREATE statements available in SQL:

- CREATE DATABASE

Example Syntax:

- CREATE DATABASE database_name;

database_name: name of the database.

- CREATE TABLE

Example Syntax:
CREATE TABLE table_name (column1 data_type(size), column2 data_type(size), column3 data_type(size),...);
table_name: name of the table.
column1 name of the first column.
data_type: Type of data we want to store in the particular column.
size: Size of the data we can store in a particular column.

2. Alter

ALTER TABLE is used to add, delete/drop or modify columns in the existing table. It is also used to add and drop various constraints on the existing table.

Example Syntax:
ADD

- ALTER TABLE table_name ADD (Columnname_1 datatype, Columnname_2 datatype, … Columnname_n datatype);

Drop

- ALTER TABLE table_name DROP COLUMN column_name;

Modify

- ALTER TABLE table_name MODIFY column_name column_type;

3. Drop

DROP is used to delete a whole database or just a table. The DROP statement destroys the objects like an existing database, table, index, or view. A DROP statement in SQL removes a component from a relational database management system (RDBMS).

Example Syntax:

- DROP TABLE table_name;

table_name: Name of the table to be deleted.

- DROP DATABASE database_name;

database_name: Name of the database to be deleted.

4. Truncate

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table

Example Syntax:

- TRUNCATE TABLE table_name;

table_name: Name of the table to be truncated.

Drop VS Truncate

- Truncate is normally ultra-fast and its ideal for deleting data from a temporary table.
- Truncate preserves the structure of the table for future use, unlike drop table where the table is deleted with its full structure.
- Table or Database deletion using DROP statement **cannot** be rolled back, so it must be used wisely.

5. Comment

As is any programming languages comments matter a lot in SQL also. In this set we will learn about writing comments in any SQL snippet.

Comments can be written in the following formats:

1. **Single line comments**: Comments starting and ending in a single line are considered as single line comments. Line starting with ‘–‘ is a comment and will not be executed.

Example Syntax:

```
-- single line comment
-- another comment
```

2. **Multi line comments:** Comments starting in one line and ending in different line are considered as multi line comments. Line starting with ‘/*’ is considered as starting point of comment and are terminated when ‘*/’ is encountered.

Example Syntax:

```
/* multi line comment
another comment */
```

6. Rename

Sometimes we may want to rename our table to give it a more relevant name. For this purpose, we can use **ALTER TABLE** to rename the name of table.

Example Syntax:

- ALTER TABLE table_name RENAME TO new_table_name;
- ALTER TABLE table_name RENAME COLUMN old_name TO new_name;

Lab#03 TASKS:

1. Create a table named Departments with the following fields: Department_ID (Primary Key), Department_Name (NOT NULL, Unique), Manager_ID (Foreign Key referencing Employees table), and Location_ID.
2. Alter the Departments table to add a new column Established_Date of type DATE with a default value of the current date.
3. Rename the Departments table to Company_Departments.
4. Drop the table Company_Departments after ensuring it has no dependent constraints.
5. Create a new table Project_Allocation similar to the Job_History table. Insert 3 rows, then truncate the table.
6. Create two tables, Parent and Child, where Child has a foreign key referencing Parent. Use deferred constraint checking to insert a row into Child first, followed by Parent.
7. Alter the Employees table to add a unique constraint on the Email column.
8. Create a table Employee_Salaries with fields Employee_ID, Salary, and a check constraint that ensures Salary is greater than 3000 but less than 10000.
9. Insert a new row into the Employees table, ensuring all constraints are satisfied. Then update the Salary of an employee to 8500 where the Employee_ID is 106.
10. Create a Projects table with a foreign key constraint referencing Department_ID from Departments. Insert related data into both tables, then try to delete a Department row that is being referenced by a Projects row, and observe the constraint violation.
11. Create a new user using SQL command Line and grant privileges. We are using this user to create our own database with related tables, which we are working on in lab#03.
12. Create table Jobs and job_History (ignore foreign keys relations) same fields as given in HR Schema in which job_ID is considered as primary key in jobs table.
13. Change the data type of 'job_ID' from character to numeric in Jobs table.(Like IT_Prog->101).
14. Write a SQL statement to add job_id column in job_history table as foreign key referencing to the primary key job_id of jobs table.
15. Insert a new row in jobs table having all the attributes and the job_ID should update in job_History table as well.
16. Add Column Job_Nature in Jobs table.
17. Create replica of employee table.
18. Write a SQL statement to add employee_id column in job_history table as foreign key referencing to the primary key employee_id of employees table.
19. Drop column Job_Nature.
20. ALTER table EMPLOYEE created in question 5 and apply the constraint CHECK on First_Name attribute such that ENAME should always be inserted in capital letters.
21. ALTER table EMPLOYEE created in question 5 and apply the constraint on SALary attribute such that no two salaries of the employees should be similar. (Hint Unique)
22. ALTER table Employee created in question 5 and apply constraint on Phone_No such that Phone_No should not be entered empty (Hint modify).
23. Write a SQL statement to insert one row into the table employees.
24. Write a SQL statement to change the salary of employee to 8000 who's ID is 105, if the existing salary is less than 1+000.
25. Write a SQL statement to add a primary key for a combination of columns employee_id and job_id in employees table, give the reason why this command is showing error.
26. Write a SQL statement to add an index named indx_job_id on job_id column in the table job_history.
27. Write a SQL statement to remove employees table.