DCIT 55A – Advanced Database System

LESSON 1 – REVIEW OF THEORY OF DATABASE

What is **Database**?

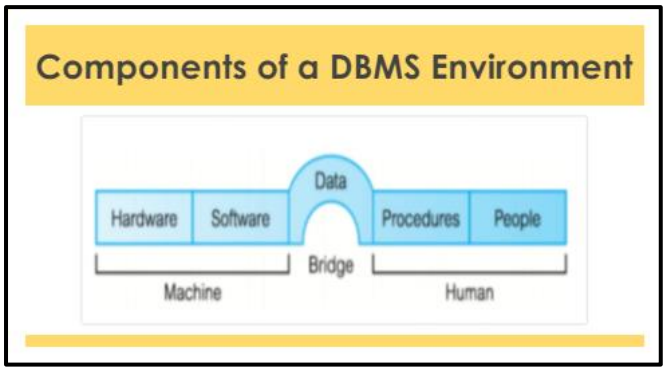
- A database is a collection of logically related data.
- Example: Personal information such as name, address, birthday and gender.

What is **Schema**?

- A schema or a database schema is the over-all design of a database.

What is a **Database System**?

- A database system is an automated system that enables users to define, create, maintain and control access to the database.



Different Facilities that DBMS provide:

- Users can define the database through a Data Definition Language (DDL).
- Users can manipulate data through a Data Manipulation Language (DML).
- DBMS can provide security system.
- DBMS can provide an integrity system.
- DBMS can provide a concurrency control system.
- DBMS can provide a recovery control system.

Entity Relationships

- One to One
- One to Many
- Many to One
- Many to Many

What is a **Relational Database**?

- A relational database is divided into logical units called table which is composed of rows and columns of data.

Other Database Concepts

- What is a **Relation**?
  - A relation is composed of rows and columns of data.
- What is an **Attribute**?
  - In relational database, the table columns correspond to attributes.
- What is a **Domain**?
  - A domain refers to a set of valid atomic values for a given attribute.
- What is a **Primary Key**?
  - A primary key refers to an attribute or field that serves as a unique identifier for a particular record within a relation.

LESSON 2 – INTRODUCTION TO SQL

SQL – A standard language for storing, manipulating and retrieving data in databases.

What is **SQL**?

- **SQL** stands for Structured Query Language.
- **SQL** lets you access and manipulate databases.
- **SQL** is an ANSI (American National Standards Institute) standard.

Objective of **SQL**

- create the database and relation structures
- perform basic data management tasks, such as the insertion, modification, and deletion of data from the relations
- perform both simple and complex queries

DATABASE LANGUAGES

- **DDL** (**DATA DEFINITION LANGUAGE**) - for defining the database structure and controlling access to the data
  - ✓ **CREATE**
  - ✓ **DROP**
  - ✓ **ALTER**

- **DML** (**DATA MANIPULATION LANGUAGE**) – for retrieving and updating data
  - ✓ **INSERT INTO**
  - ✓ **SELECT**
  - ✓ **UPDATE**
  - ✓ **DELETE**

SQL CREATE STATEMENT

- creates an object (a table, for example) in the database
  - **CREATE DATABASE** database_name;
  - **CREATE TABLE** table_name (column1 datatype, column2 datatype);
  - **CREATE TABLE** new_table_name **AS SELECT** column1, column2,… **FROM** existing_table_name **WHERE** condition;

SQL DROP STATEMENT

- **DROP** deletes an object in the database, usually irretrievably.
  - **DROP DATABASE** database_name;
  - **DROP TABLE** table_name;

SQL ALTER STATEMENT

- **ALTER** modifies the structure an existing object in various ways - for example, adding a column to an existing table.
  - **ALTER TABLE** table_name ADD column_name datatype;
  - **ALTER TABLE** table_name DROP **COLUMN** column_name;

SQL BASIC DATA TYPES

- Each column in a database table is required to have a name and a datatype.

SQL String Data Types

- **CHAR(size)** - Fixed length (0-255 characters) Default is 1
- **VARCHAR(size)** - Variable length (0-65535 characters)
- **TEXT** - Holds a string with a maximum length of 65,535 bytes

SQL Numeric Data Types

- **INT(size)** - Signed range is from 2147483648to2147483647. Unsigned range is from 0 to 4294967295
- **FLOAT** - Range is from - 1.79E + 308 to 1.79E + 308
- **BOOLEAN** - Zero is considered as false, nonzero values are considered as true

SQL Date and Time Data Types

- **DATE** - A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
- **DATETIME** - A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
- **TIME** - A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
- **YEAR** - A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000

SQL CONSTRAINTS

- Used to specify rules for the data in a table.
    - **NOT NULL**
    - **UNIQUE**
    - **PRIMARY KEY**
    - **FOREIGN KEY**
    - **CHECK**
    - **DEFAULT**

SQL COMMAND:

SQL SELECT STATEMENT

- The **SELECT** statement is used to select data from a database.
    - **SELECT** * **FROM** table_name;
    - **SELECT** column_name, column_name **FROM** table_name

SQL DISTINCT STATEMENT

- The **DISTINCT** keyword can be used to return only distinct (different) values.
    - **SELECT DISTINCT** column_name1, column_name2 **FROM** table_name

SQL WHERE CLAUSE

- The **WHERE** clause is used to filter records. (numeric fields should not been closed in quotes)
    - **SELECT** column_name, column_name **FROM** table_name **WHERE** condition

OPERATORS IN WHERE CLAUSE

Operator	Description
=	Equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range (ie. BETWEEN 1 AND 3)
LIKE	Search for a pattern (ie. LIKE '%jo%')
IN	To specify multiple possible values for a column. ie. IN('California','New York')

SQL AND & OR OPERATORS

- The **AND** operator displays a record if both the first condition AND the second condition are true
- The **OR** operator displays a record if either the first condition OR the second condition is true.
    - **SELECT** column_name, column_name **FROM** table_name **WHERE** condition **AND**/**OR** condition

SQL ORDER BY KEYWORD

- The **ORDER BY** key word is used to sort the result-set by one or more columns.
    - **SELECT** column_name1, column_name2 **FROM** table_name **ORDER BY** column_name **ASC**|**DESC**

SQL LIKE OPERATORS

- The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column.
    - **SELECT** column_name(s) **FROM** table_name **WHERE** column_name **LIKE** pattern;

SQL WILDCARDS

- A wildcard character can be used to substitute for any other character(s) in a string.

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character
[charlist]	Sets and ranges of characters to match
[^charlist] or [!charlist]	Matches only a character NOT specified within the brackets

SQL IN OPERATOR

- The **IN** operator allows you to specify multiple values in a WHERE clause.
    - **SELECT** column_name(s) **FROM** table_name **WHERE** column_name **IN** (value1,value2,...)

SQL BETWEEN OPERATOR

- The **BETWEEN** operator selects values within a range.
    - **SELECT** column_name(s) **FROM** table_name **WHERE** column_name **BETWEEN** value1 **AND** value2

SQL UPDATE OPERATOR

- The **UPDATE** statement is used to update existing records in a table.
    - **UPDATE** table_name **SET** column1 = value1, column2=value2 **WHERE** some_column = some_value

SQL DELETE STATEMENT

- The **DELETE** statement is used to delete rows in a table.
    - **DELETE FROM** table_name **WHERE** some_column = some_value

SQL COUNT FUNCTION

- The **COUNT**() function returns the number of rows that matches a specified criteria.
    - **SELECT COUNT** (column_name) **FROM** table_name **WHERE** condition;

SQL AVG FUNCTION

- The **AVG**() function returns the average value of a numeric column.
    - **SELECT AVG** (column_name) **FROM** table_name **WHERE** condition;

SQL SUM FUNCTION

- The **SUM**() function returns the total sum of a numeric column.
    - **SELECT SUM** (column_name) **FROM** table_name **WHERE** condition;

SQL MIN % MAX FUNCTION

- The **MIN**() function returns the smallest value of the selected column.
- The **MAX**() function returns the largest value of the selected column.
    - **SELECT MIN**/**MAX**(column_name) **FROM** table_name **WHERE** condition;

LESSON 6 – TRANSACTIONS

Transaction

- Units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

SQL TRANSACTION COMMANDS

- **BEGIN TRANSACTION**
  - This command is used to start a new transaction
  - Syntax: **BEGIN TRANSACTION**;
- **COMMIT**
  - This command is the transactional command used to save changes invoked by a transaction to the database.
  - Syntax: **COMMIT**;
- **ROLLBACK**
  - This command is the transactional command used to undo transactions that have not already been saved to the database.
  - Syntax: **ROLLBACK**;

EXAMPLE:

BEGIN TRANSACTION;

UPDATE Accounts **SET** Balance = Balance – 100
WHERE id = 1;

UPDATE Accounts **SET** Balance = Balance + 100
WHERE id = 2;

IF @@ERROR = 0

COMMIT;

ELSE

ROLLBACK;

SAVEPOINT

- It is a way of implementing sub transactions (nested transactions) within a relational database management system by indicating a particular point within a transaction that a user can “roll back” to in case of failure.
  - Syntax: **SAVEPOINT** savepoint_name;

Use of **SAVEPOINT**

START TRANSACTION

INSERT INTO Table1 (Column1) **VALUES** (“Value1”);

SAVEPOINT SP1;

INSERT INTO Table1 (Column1) **VALUES** (“Value2”);

ROLLBACK TO SP1;

COMMIT;

Release savepoint

- Deletes a savepoint within a transaction.
  - Syntax: **SAVEPOINT** savepoint_name;

Remove Savepoint

- It removes a savepoint within a transaction.
  - Syntax: **ROLLBACK TRANSACTION TO** savepoint_name;

ACID

- Stands for **Atomocity, Consistency, Isolation, and Durability**.
- The four properties of relational database systems that help in making sure that we are able to perform the transactions in a reliable manner.

TRANSACTION ISOLATION LEVELS

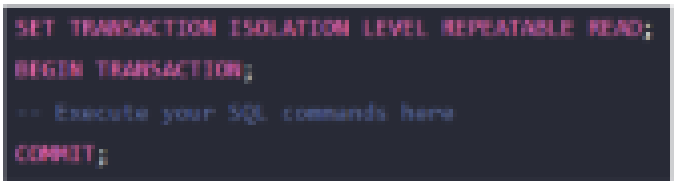
READ UNCOMMITED - This is **the lowest level of isolation**. One transaction may read not yet committed changes made by other transaction, also known as **“Dirty Reads”**.

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
BEGIN TRANSACTION;
-- Execute your SQL commands here
COMMIT;
```

READ COMMITED - A transaction only sees data changes committed before it started, averting “Dirty Reads”. However, it may experience **“Nonrepeatable Reads”**.

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;
-- Execute your SQL commands here
COMMIT;
```

REPEATABLE READ – Once a transaction reads a row, any other transaction's writes (changes) onto those rows are blocked until the first transaction is finished. Preventing "Non-repeatable Reads". However, **"Phantom Reads"** may still occur.



SERIALIZABLE - This is **the highest level of isolation**. It avoids "Dirty Reads", "Non-repeatable Reads" and "Phantom Reads". This is done by fully isolating one transaction from others.