ADVANCED INFORMATION TECHNOLOGY INSTITUTE

GHANA-INDIA KOFI ANNAN CENTRE OF EXCELLENCE IN ICT

DATABASE MANAGEMENT SYSTEM (DBMS)

DATABASE MANAGEMENT SYSTEM (DBMS)

DBMS is a software system that enables users to define, create, maintain, and control access to the database. The **DBMS** is the software that interacts with the users' application programs and the database. Typically, a **DBMS** provides the following facilities:

- It allows users to define the database, usually through a Data Definition Language (**DDL**).The **DDL** is a computer language used to create and modify the structure of database objects in a database. These database objects include views, schemas, tables, indexes, etc.
- It allows users to insert, update, delete, and retrieve data from the database, usually through a Data Manipulation Language (**DML**). Having a central repository for all data and data descriptions allows the **DML** to provide a general inquiry facility to this data, called a query language. The provision of a query language alleviates the problems with file-based systems where the user has to work with a fixed set of queries or there is a proliferation of programs, giving major software management problems. The most common query language is the Structured Query Language (**SQL**).

DATA DEFINITION LANGUAGE (DDL)

The Data Definition Language(**DDL**) is a computer language used to create and modify the structure of database objects in a database. These database objects include views, schemas, tables, indexes, etc. It is used to establish and modify the structure of objects in a database by dealing with descriptions of the database schema. Unlike data manipulation language (**DML**) commands that are used for data modification purposes, **DDL** commands are used for altering the database structure such as creating new tables or objects along with all their attributes (data type, table name, etc.). Commonly used **DDL** in **SQL** querying are **CREATE**, **ALTER**, **DROP**, and **TRUNCATE**.

Create: This command builds a new table and has a predefined syntax. The **CREATE** statement syntax is:

- CREATE **SCHEMA** [schema name];
- CREATE **TABLE** [table name] ([column definitions]) [table parameters];

For example, You need first create a Database or Schema:

```
Create schema school;


CREATE table Students(
Student_Id int Auto_increment,
Full_name varchar(50) ,
Course_Id varchar(20) ,
Gender varchar(10),
Phone int(20),
Marks decimal(10,2),
Address varchar(100)
);
```

The mandatory semi-colon at the end of the statement is used to process every command before it. In this example, the string **VARCHAR** is used to specify the data type. Other data types can be **DATE**, **NUMBER**, or **INTEGER**.

ALTER: An alter command modifies an existing database table. This command can add up additional column, drop existing columns and even change the data type of columns involved in a database table. Example;

ALTER **TABLE Students** ADD PRIMARY KEY (Student_Id);

In this example, we added a unique primary key to the table to add a constraint and enforce a unique value. The constraint "**Student_Id**" is a primary key and is on the **Students** table.

DROP: A drop command is used to delete objects such as a table, index or view. A **DROP** statement cannot be rolled back, so once an object is destroyed, there's no way to recover it. In this example, we're deleting the **Students** table;

DROP TABLE **Students**;

TRUNCATE: Similar to **DROP**, the **TRUNCATE** statement is used to quickly remove all records from a table. However, unlike DROP that completely destroys a table, **TRUNCATE** preserves its full structure to be reused later. In this example, we're marking all the extents of the **Students** table for deallocation, so they're considered empty for reuse.

TRUNCATE TABLE Students ;

DATA MANIPULATON LANGUAGE (DML)

A Data Manipulation Language (**DML**) is a family of computer languages including commands permitting users to manipulate data in a database. This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data. DML is mostly incorporated in SQL databases.

DML resembles simple English language and enhances efficient user interaction with the system. The functional capability of **DML** is organized in manipulation commands like **SELECT**, **UPDATE**, **INSERT INTO** and **DELETE FROM**, as described below:

- **SELECT**: This command is used to retrieve rows from a table. The syntax is SELECT [column name(s)] from [table name] where [conditions]. SELECT is the most widely used DML command in SQL. Example;

```
select * from Students;
```

❖ **INSERT:** This command adds one or more records to a database table. The insert command syntax is INSERT INTO [table name] [column(s)] VALUES [value(s)].

Insert into Students(**Student_Id**, **Full_name**, **Course_Id**, **Gender**, **Phone**, **Marks**, **Address**)
Values ('stud001', 'John Doe', 'csd001', 'Male', 0577983546, 77.8, 'Bolagatanga High Court');

❖ **UPDATE:** This command modifies data of one or more records. An update command syntax is UPDATE [table name] SET [column name = value] where [condition]. Example;

Update Students set phone = 0203874523 **where student_id** = 'stud001';

❖ **DELETE:** This command removes one or more records from a table according to specified conditions. Delete command syntax is DELETE FROM [table name] where [condition]. Example;

Delete from Students where student_id = 'stud001';

OTHER STATEMENTS

- **SELECT DISTINCT:** The Select Distinct statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values. Example;

```
select distinct * from Students;
```

- **OR OPERATOR:** The OR operator displays a record if any of the conditions separated by OR is true. Example;

```
select * from Students where student_id = 'stud001' or  full_name = 'John';
```

- **AND OPERATOR:** The AND operator displays a record if all the conditions separated by AND are true. Example;

```
select * from Students where student_id = 'stud001' and  full_name = 'John';
```

OTHER STATEMENTS

- **ORDER BY:** The Order by keyword is used to sort the result-set in ascending or descending order. The Order by keyword sorts the records in ascending order by default. To sort the records in descending order, use the **DESC** or **ASEC** keyword. Example;

```
select * from Students order by full_name desc limit 1;
```

- **COUNT:** The Count() function returns the number of rows. Example;

```
select count(*) from Students;
```

- **SUM:** The Sum function returns the total sum of a numeric column. Example;

```
select sum(marks) from Students;
```

OTHER STATEMENTS

❖ **LIKE OPERATOR:** The Like operator is used in a Where clause to search for a specified pattern in a column. The following SQL statement selects all customers with a Full_name starting with "a";

```
select * from Students where full_name like ‘a%’;
```

LIKE OPERATORS	DESCRIPTION
WHERE Full_name LIKE 'a%'	Finds any values that start with "a"
WHERE Full_name LIKE '%a'	Finds any values that end with "a"
WHERE Full_name LIKE '%or%'	Finds any values that have "or" in any position
WHERE Full_name LIKE '_r%'	Finds any values that have "r" in the second position
WHERE Full_name LIKE '_r%'	Finds any values that have "r" in the second position
WHERE Full_name LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE Full_name LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

OTHER STATEMENTS

- **BETWEEN:** The Between operator selects values within a given range. The values can be numbers, text, or dates. The Between operator is inclusive: begin and end values are included. Example;

```
select * from Students where marks Between 50 and 90;
```

- **JOIN:** The Join clause is used to combine rows from two or more tables, based on a related column between them. Example, The following SQL statement is selecting from three tables;

```
select Students.full_name, Students.phone, Students.marks from Students

inner join Courses on Students.course_id = courses.id

inner join Professor on Students.course_id = Professor.course_id;
```