SS2 Data Processing Lesson Note First Term

NAME:

SUBJECT: DATA PROCESSING CLASS: SS 2

SCHEME OF WORK

WEEK TOPIC

1. Revision of last term's work
2. Types of Data Models (a) Definition of Data model (b) Type of data models
3. Data Modelling (a) Creating Tables, Forms, Queries and Reports (b) Significance of data model (c) Examples of standard data models
4. Normal Forms: (a) Examples of Tables in first normal forms (b) Tables ion second and third normal forms
5. Normal Forms: (c) Problems of table in first normal form (d) Determinant of normal forms (e) Foreign keys
6. Entity Relationship Model: (a) Entities, attributes and relationship (b) Additional features of entity model
7. Mid term break
8. Relational Model: (a) Creating and modifying relations using SQL (b) Integrity constraints over relations.
9. Relational Model: (c) Enforcing Integrity constraints (d) Querying relational data
10. Revision

11-13 Examination

Reference Book

A textbook of Data Processing for SSS 2 by Adedapo F O and Mitchell A. S

WEEK TWO

DATE:..

TOPIC: TYPES OF DATA MODELS

CONTENT:

1. Definition of Data Model
2. Type of data model

SUBTOPIC 1: Definition of Data Model

Data Models are fundamental entities to introduce abstraction in a DBMS. **Data models** define how **data** is connected to each other and how they are processed and stored inside the system.

A **data model** is an abstract model that organizes elements of data and standardizes how they relate to one another and to properties of the real world entities. For instance, a data model may specify that the data element representing a car be composed of a number of other elements which, in turn, represent the color and size of the car and define its owner.

THREE LEVELS OF DATA MODEL

The ANSI/SPARC three level architecture

1. Conceptual Data model
2. Logical Data Model

iii. Physical Data Model

Conceptual Data Model: is a technology independent specification of the data to be held in the database. It is the focus of communication between the data modeler and business stakeholders, and it is usually presented as a diagram with supporting documentation.

Logical Data Model: is the translation of the conceptual model into structure that can be implemented using a database management system (DBMS).

This model specifies tables and columns. These are the basic building blocks of relational databases, which are implemented using a relational database management system (RDBMS)

Physical Data Model: this incorporates any changes necessary to achieve adequate performance and is also presented in terms of tables and columns, together with a specification of physical storage and access mechanisms.

EVALUATION:

1. What is conceptual data model?
2. What does data model focus on?

iii. What is logical data model?

Sub-topic 2

Types of Data Model

There are six major types of data model

1. Database Model
2. Data Structured Diagram
3. Entity-relationship model
4. Geographic data model
5. Generic data model
6. Semantic data model

Database Model: A database model is a specification describing how a database is structured and used.

Types of Database Model

1. **Flat model**: This may not strictly qualify as a data model. The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another.

1. **Hierarchical model:**The hierarchical model is similar to the network model except that links in the hierarchical model form a tree structure, while the network model allows arbitrary graph.

iii. **Network model:** This model organizes data using two fundamental constructs, called records and sets. Records contain fields, and sets define one-to-many relationships between records: one owner, many members. The network data model is an abstraction of the design concept used in the implementation of databases.

1. **Relational model:**is a database model based on first-order predicate logic. Its core idea is to describe a database as a collection of predicates over a finite set of predicate variables, describing constraints on the possible values and combinations of values. The power of the relational data model lies in its mathematical foundations and a simple user-level paradigm.

1. **Object-relational model:**Similar to a relational database model, but objects, classes and inheritance are directly supported in database schemasand in the query language.

1. **Star Schema:**The simplest style of data warehouse schema. The star schema consists of a few "fact tables" (possibly only one, justifying the name) referencing any number of "dimension tables". The star schema is considered an important special case of the snowflake schema.

Data Structure Diagram

A data structure diagram (DSD) is a diagram and data model used to describe conceptual data models by providing graphical notations which document entities and their relationships, and the constraints that bind them.

Data structure diagrams are an extension of the entity-relationship model. In DSDs, attributes are specified inside the entity boxes rather than outside of them, while relationships are drawn as boxes composed of attributes which specify the constraints that bind entities together.

Entity-relationship model

An entity-relationship model (ERM), sometimes referred to as an entity-relationship diagram (ERD), is an abstract conceptual data model (or semantic data model) used in software engineering to represent structured data. There are several notations used for ERMs

Geographic data model

A data model in Geographic information systems is a mathematical construct for representing geographic objects or surfaces as data.

Generic data model

Generic data models are generalizations of conventional data models. They define standardized general relation types, together with the kinds of things that may be related by such a relation type. Generic data models are developed as an approach to solve some shortcomings of conventional data models. For example, different modelers usually produce different conventional data models of the same domain

Semantic data model

A semantic data model in software engineering is a technique to define the meaning of data within the context of its interrelationships with other data. A semantic data model is an abstraction which defines how the stored symbols relate to the real world. A semantic data model is sometimes called a conceptual data model.

READING ASSIGNMENT:

Study the topic 'Data modelling" using students' textbook

WEEKEND ASSIGNMENT:

OBJECTIVE TEST:

1. In a database data is organized into simple (a) tables (b) rows (c) columns (d) type
2. A data model is a ___ representation of the data structure that are required by a database. (a) logical (b) conceptual (c) physical (d) dynamic
3. The following is not a step of design process: (a) planning logical design (c) survey (d) implementation

WEEK THREE

DATE:.......................................

TOPIC: Data Modelling

CONTENT:

1. Creating Tables, Forms, Queries and Reports
2. Significance of data model

SUB-TOPIC 1: Creating Tables, Forms, Queries and Reports

Data Modelling in software engineering is the process of creating a data model by applying formal data model descriptions using data modeling techniques.

Data modeling is a method used to define and analyze data requirements needed to support the business processes of an organization. The data requirements are recorded as a conceptual data model with associated data definitions. Actual implementation of the conceptual model is called a logical data model.

Adding tables to a database

1. In the database window, click on tables in the object bar. Double click create table in design view. Table design view opens.
2. Type a field name on the first empty line of the Field Name column. Then press Tab to move to the data Type column.

3. When you move to the Data Type column, an arrow appears for a drop down list. Open the Data Type drop-down list and select field type.
4. Press Tab to move to Description column and type a description of the field. (Optional)
5. In the bottom half of the dialog box, you see Filed Properties for the field type you selected. Make any changes to them that you want.
6. If you have more fields to enter, repeat steps 2 through 5
7. Click the Table Design window's Close (x) button
8. When you are asked if you want to save your changes to the table, click Yes. The Save As dialog box appears.
9. Type a name for the table in the Table Name text box, and then click OK

Creating Query

The easiest way to create a query is with the Simple Query wizard, which enable you to select fields you want to display.

1. Open the database you want to work with and click the Queries tab
2. Double-click Create Query by Using wizard. The first dialog box of the simple query wizard appears. This dialog box might look familiar, it is similar to the first screen of the Table Wizard described earlier.
3. Choose the table from which you want to select field from the Table/Queries drop-down list.
4. Click a field name in the Available Fields list: then click the > button to move it to the Selected Fields list. Repeat to move all the fields you want. Or move them all at once with the >> button. Then click on Next.
5. Select type of Query whether Detailed (showing every field of the every record) or Summary (showing selected SUM, AVG, MIN OR MAX of the selected fields) and click on NEXT
6. Enter a title for the Query in the What Title Do You want for your Query? Text box. I will call it Student's record.
7. Click Finish to view the query results.

Designing Report

Report wizard offers a good compromise between ease-of use and flexibility. With the Report Wizard, you can use multiple tables and queries and choose a layout and format for your report.

1. Open the database containing the table or query on which you want to report
2. Click the Reports tab in the Database window.
3. Double click Create Report by using wizard to start the Report wizard.
4. From the Table/Queries drop-down list, select a table or query from which you want to include fields.
5. Click a field in the Available Fields list, and then click the > button to move it to the Selected Fields list.
6. If desired, select another table or query from the Table/Queries list and repeat step 5.
7. If you want the records grouped by any of the fields you selected, click the field, and then click the > button. You can select several grouping levels in order you want then.

8. The wizard asks you what sort order you want to use. If you want sorted records, open the top drop-down list and select a field by which to sort.
9. In the next dialog box, choose a layout option from the Layout section. When you click an option button, the sample in the box changes to show your selection.
10. Choose the orientation for your printed report: Portrait (across the narrow edge of the paper) or Landscape
11. In the next wizard dialog box, choose a report style.
12. You are asked for a report title. Enter one in the Report textbox and click Finish to see your report in Print Preview.

Sub-topic 2

Significance of data model

E/R modeling was revolutionary in that – for the first time in data processing – data, not processes were at the center of both business analysis and system design. The implications were enormous; data could now become a reusable commodity, which meant that every unique data element was identified and inventoried once and only once. That provided the ability to track every person or program using the unique data elements for any purpose.

The concepts of data-driven analysis, and much later of data-driven methodologies, were born as business analyst and data modeling practitioners realized that they could finally create a business model of their organization that would logically portray a non-redundant 'single version of the truth' of their enterprise in terms of its data resources. Companies created departments called data resource management (DRM) and information resource management (IRM) to manage their business data as a corporate asset, just like they managed their financial asset, fixed asset, real estate, or human resources.

EVALUATION:

1. What is data modeling?
2. What is the significance of data modeling?

iii. What is importance of data modeling?

Examples of standard data model

A standard data model or industry standard data model (ISDM) is a data model that is widely applied in some industry, and shared amongst competitors to some degree. They are often defined by standards bodies, database vendors or operating system vendors.

When in use, they enable easier and faster information sharing due to the fact that heterogeneous organization have a standard vocabulary and pre-negotiated semantics, format, and quality standards for exchange data. The standardization has an impact on software architecture as solutions that vary from the standard may cause data sharing issues and problems if data is out of compliance with the standard.

The more effective standard models have developed in the banking, insurance, pharmaceutical and automotive industries, to reflect the stringent standards applied to customer information gathering, customer privacy, consumer safety, or just in time manufacturing.

The most complex data models known are in military use, and consortia such as NATO tend to require strict standards of their members' equipment and supply database. However, they typically do not share these with non-NATO competitors, and so calling these 'standard' in the same sense as commercial software is probably not very appropriate.

EVALUATION:

1. What are the example of standard data model
2. Explain standard data model.

READING ASSIGNMENT:

Study the topic 'Data modelling" using students' textbook

WEEKEND ASSIGNMENT:

OBJECTIVE TEST:

1. In a database data is organized into simple (a) tables (b) rows (c) columns (d) type
2. A data model is a ___ representation of the data structure that are required by a database. (a) logical (b) conceptual (c) physical (d) dynamic
3. The following is not a step of design process: (a) planning logical design (c) survey (d) implementation
4. Creating a Query, choose the following from New Query dialog box. (a) creating query wizard (b) design view (c) simple query wizard (d) none of the above
5. To select all fields to be added to a generated form click (a) >> (b) < (c) > (d) <<

WEEK FOUR

DATE:..

TOPIC: NORMAL FORMS

CONTENT:

1. Examples of tables in first normal forms
2. Tables in second and third normal forms

Sub-topic 1

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization is used for mainly two purposes:

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Problem Without Normalization

Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anamolies are very frequent if Database is not Normalized. To understand these anomalies let us take an example of **Student** table.

S_id	S_Name	S_Address	Subject_opted
401	Adam	Noida	Bio
402	Alex	Panipat	Maths
403	Stuart	Jammu	Maths
404	Adam	Noida	Physics

- **Updation Anamoly :**To update address of a student who occurs twice or more than twice in a table, we will have to update **S_Address** column in all the rows, else data will become inconsistent.
- **Insertion Anamoly :**Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert **NULL** there, leading to Insertion Anamoly.
- **Deletion Anamoly :**If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

Normalization Rule

Normalization rule are divided into the following normal form.

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

First Normal Form (1NF)

As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

The **Primary key** is usually a single column, but sometimes more than one column can be combined to create a single primary key. For example consider a table which is not in First normal form

Student Table :

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

Student	Age	Subject
Adam	15	Biology
Adam	15	Maths
Alex	14	Maths
Stuart	17	Maths

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Sub-topic 2

Second Normal Form (2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails **Second normal form**.

In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for. While this is searchable, and follows First normal form, it is an inefficient use of space. Also in the above Table in First Normal Form, while the candidate key is {**Student**, **Subject**}, **Age** of Student only depends on Student column, which is incorrect as per Second Normal Form. To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

Student	Age
Adam	15
Alex	14
Stuart	17

In Student Table the candidate key will be **Student** column, because all other column i.e **Age** is dependent on it.

New Subject Table introduced for 2NF will be :

Student	Subject
Adam	Biology

```
Adam    Maths
Alex    Maths
Stuart  Maths
```

In Subject Table the candidate key will be {**Student**, **Subject**} column. Now, both the above tables qualifies for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this *transitive functional dependency* should be removed from the table and also the table must be in **Second Normal form**. For example, consider a table with following fields.

Student_Detail Table :

```
Student_id Student_name DOB Street city State Zip
```

In this table Student_id is Primary key, but street, city and state depends upon Zip. The dependency between zip and other fields is called **transitive dependency**. Hence to apply **3NF**, we need to move the street, city and state to new table, with **Zip** as primary key.

New Student_Detail Table :

```
Student_id Student_name DOB Zip
```

Address Table :

```
Zip Street city state
```

The advantage of removing transtive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anamoly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form

- and, for each functional dependency (X -> Y), X should be a super Key.

EVALUATION:

1. What is normalization?
2. What is 1NF?

READING ASSIGNMENT:

Study the topic 'Normal Forms" using students' textbook

WEEKEND ASSIGNMENT:

OBJECTIVE TEST:

1. A table always meets the requirement of its ____ form.
2. The normal forms are applicable to individual ______

WEEK FIVE

DATE:.......................................

TOPIC: NORMAL FORMS

CONTENT:

1. Problems of table in first normal form
2. Determinant of normal forms

Sub-topic 1

Problems of table in first normal form

The basic problem is that department names and addresses are really data about departments rather than employees, and belong to a separate *Department* table. We therefore establish a third table for department data, resulting in the three-table model below:

Employee table

Department Table

Qualification Table

We leave Department number in the Employee table to serve as a cross-reference, in the same way that we retained Employee Number in the Qualification table. Our data now normalized.

Sub-topic 2

Determinant of normal forms

It is important to understand that this whole procedure of separating hospital data relied on the fact that for a given hospital number there could be only one hospital name, contact person, hospital type and teaching status. In fact we could look at the dependency of hospital data on hospital number as the cause of the problem. Every time a particular hospital number appeared in the Operation table, the hospital name, contact person, hospital type and teaching status were the same.

Hospital data removed to separate table.

Formally, we say Hospital Number is a determinant of the other four columns. We can show this as:

Hospital Number   Hospital Name, Contact Person, Hospital Type, Teaching Status

Where we read "   " as "determines" or "is a determinant of".

Determinants need not consist of only one column; they can be a combination of two or more columns, in which case we can use a + sign to indicate such a combination.

E.g. Hospital Number + Operation Number   Surgeon Number.

This leads us to a more formal description of the procedure:

1. Identify any determinants, other than the primary key, and the columns they determine.
2. Establish a separate table for each determinant and the columns it determines. The determinant becomes the key of the new table.
3. Name the new tables.
4. Remove the determined columns from the original table. Leave the determinants to provide links between tables.

Primary Keys

A primary key is a nominated column or combination of columns that has a different value for every row in the table. Each table has one (and only one) primary key. When checking this with a business person, we would say, "if I nominated, say, a particular account number, would you be able to guarantee that there was never more than one account with that number?"

Candidate Keys

Sometimes more than one column or combination of columns could serve as a primary key. E.g, we could have chosen Drug Name rather than Drug Short Name as the primary key of the Drug table (assuming, of course, that no two drugs could have the same name). we refer to such possible primary keys, whether chosen or not, as *candidate keys*. From the point of view of normalization, the important thing is that candidate keys that have not been chosen as the primary key, such as Drug Name, will be determinants of every column in the table, just as the primary key is.

Foreign Keys

Recall that when we removed repeating groups to a new table, we carried the primary key of the original table with us, to cross-reference or "point back" to the source. In moving from first to third normal form, we left determinants behind as cross-reference to the relevant rows in the new tables.

These cross-referencing columns are called foreign keys, and they are our principal means of linking data from different tables.

EVALUATION:

1. What is primary key?
2. What is the use of foreign keys?

iii. What are candidate keys?

READING ASSIGNMENT:

Study the topic 'Entity Relational Model" using students' textbook

WEEKEND ASSIGNMENT:

OBJECTIVE TEST:

1. Foreign key is a field in a relational table that matches a ___ of another table.

(a) candidate key (b) row (c) field (d) none of the above

1. Each value of the primary key uniquely identifies one ___ of the table. (a) column (b) row (c) field (d) none of the above
2. The ___ the normal form applicable to a table, the less vulnerable it is to inconsistencies and anomalies. (a) better (b) lower (c) higher (d) none of the above

WEEK SIX

DATE:.......................................

TOPIC: ENTITY RELATIONSHIP MODEL

CONTENT:

1. Entities, attributes and relationship
2. Additional features of entity model

Sub-topic 1

Entities, attributes and relationship

The rectangle in the diagram above are called entity type and the ovals are called attributes. The entities are the 'things' in business environment about which we want to store data. The attributes provide us with a means of organizing and structuring the data.

Entities

Entities are drawn as rectangular boxes containing a noun in singular form.

You will see later that each entity you draw ultimately becomes a table in your database. You might want to keep this transformation from entity to table in mind when selecting the names of your entities. E.g. your entity names should be short but descriptive.

Relationship

A relationship between entities is drawn as a line bisected by a diamond. The diamond contains a verb (or short verb phrase) that describes the nature of the relationship between the entities.

Named relationship are used to make the ERDs more readable. However, unlike entity names, relationship names never show up in the final database.

Attributes

Attributes are properties or characteristics of a particular entity about which we wish to collect and store data. In addition, there is typically one attribute that uniquely identifies particular instances of the entity. E.g. each of your customers may have a unique customer ID. Such attributes are known as *Key attributes*.

Sub-topic 2

Additional features of entity model

Associative entities

Given the number and importance of the attributes attached to the 'buys' relationship, it makes sense to treat the relationship as an entity as an entity in its own right. To transform a relationship into an entity on an ERD, we use a special symbol called an *associative entity*. The notation for an associative entity is a relationship diamond nested inside of an entity rectangle.

EVALUATION:

1. What are entities?
2. What are attributes?

iii. What are associative entities?

READING ASSIGNMENT:

Study the topic 'Relational Model" using students' textbook

WEEKEND ASSIGNMENT:

OBJECTIVE TEST:

1. The ___ provide us with a means of organizing and structuring the data.
2. A set of tools and procedures for applying the tools that specifies the notation used within the organization is called ______
3. To transform a relationship into an entity on an ERD, we use a special symbol called an (a) Entity (b) Attribute (c) associative entity (d) none of the above

WEEK SEVEN : MID TERM BREAK

WEEK EIGHT

DATE:.......................................

TOPIC: RELATIONAL MODEL

CONTENT:

1. Creating and modifying relations using SQL
2. Integrity constraints over relations

Sub-topic 1

The relational model for database management is a database model based on first – order predicate logic, first formulated and proposed in 1969 by E.F. Codd.

Diagram of a sample database according to the relational model

The purpose of the relational model is to provide a declarative method for specifying data and queries.

Creating and modifying relations using SQL

Creating and modifying relations simply is the process of creating and modifying a database. Data-definition queries can be very convenient. You can regularly delete and re-create parts of your database schema merely by running some queries. Consider using a data-definition query if you are familiar with SQL statements and you plan to delete and re-create particular tables, constraints, indexes or relationships.

A relation is a table structure definition (set of column definitions) along with the data appearing in that structure. The structure definition is the heading and the data appearing in it is the body, a set of rows. A database **relvar** (relation variable) is commonly known as **base table.** Operators called the Update operators are used to interact with the database. Example, INSERT, UPDATE or DELETE.

SQL

SQL (often referred to as structured query language) is a programming language designed for managing data in relational database management system(RDBMS) it is use for data insert, query, update and delete, schema creation and modification, and data access control. SQL can achieve the following in a database:

1. SQL can execute queries against a database
2. SQL can retrieve data from a database
3. SQL can insert records into a database
4. SQL can update records in a database
5. SQL can delete records from a database
6. SQL can create new databases
7. SQL can create new tables in a database
8. SQL can create store procedures in a database
9. SQL can create views in a database
10. SQL can set permissions on tables, procedures and views.

SQL can be divided into two parts:

1. Data Manipulation Language(DML)
2. Data Definition Language(DDL)

DML is the part that supports query and update commands

KEYWORD	USE
SELECT	Extract data from a database
UPDATE	Updates data in a database
DELETE	Deletes data from a database
INSERT	Insert new data into a database

DDL part of SQL permits creating and deleting of Databases and database tables.

KEYWORDS	USE
CREATE DATABASE	Creates a new database
ALTER DATABASE	Modifies a database
CREATE TABLE	Creates a new table
ALTER TABLE	Modifies a table
DROP DATABASE	Deletes a database
DROP TABLE	Deletes a table
CREATE INDEX	Creates an index(search key)it is always an integer
DROP INDEX	Deletes an index
ADD	Add a column or a constraint to a table

We will be looking at Microsoft Access as an example of a Database application.

SQL CREATE STATEMENT:

The SQL create statement is used for creating database and tables. Take note that SQL syntax are case sensitive.

1. The CREATE DATABASE Statement: This is used to create a new database. The syntax(command) is as follows:

CREATE DATABASE database_name

(That is type CREATE then a spacebar, DATABASE then another spacebar then type the database name then hit Enter).

***NOTE**

The syntax CREATE DATABASE **must** be in Upper case, while the database name is optional i.e lower or upper case depending on your choice. Then the underscore between database and name is to make it one word because a database name must be one word no space in between.

Example:

Write an SQL syntax to create a database called dlhs.

SOLUTION:

CREATE DATABASE dlhs

1. The CREATE TABLE Statement: This is used to create a table in a database. The syntax(command) is as follows:

CREATE TABLE table_name

(That is, type CREATE then a spacebar, TABLE then another spacebar then types the table name then hit Enter).

The only required elements of a CREATE TABLE command are the CREATE TABLE command itself and the name of the table, but usually you will want to define some fields or other aspects of the table. Consider this simple example.

Example:

Suppose that you want to create a table called student to store the names, age, class and the scores of students offering ICT in your school.

Solution

CREATE TABLE student

(

Student_id,

FirstName,

LastName,

Age,

Class,

Scores

)

The empty student table will now look like this:

```
Student_id  FirstName   lastName   age      class     scores
```

The empty table can now be filled with INSERT INTO statement.

1. Go to all program and navigate to Microsoft Access
2. Click on blank Database
3. Enter the name dlhs on the column for database name
4. On the main menu click on Create tab
5. On the **Create**tab, in the **Macros & Code** group, click **Query Design**.
6. Close the **Show Table**dialog box.
7. On the **Design**tab, in the **Query Type** group, click **Data Definition**.

The design grid is hidden, and the SQL view object tab is displayed.

1. Type the following SQL statement:
2. **CREATE TABLE student (student_id int(10), FirstName varchar(40), LastName varchar(40), Age int(10), Class varchar(40), Scores int(5))**
3. On the **Design**tab, in the **Results** group, click **Run**.

MODIFY A TABLE

To modify a table, you use an ALTER TABLE command. You can use an ALTER TABLE command to add, modify, or drop (remove) columns or constraints. An ALTER TABLE command has the following syntax:

ALTER TABLE *table_name predicate*

where *predicate* can be any one of the following:

ADD COLUMN *field type*[(*size*)] [NOT NULL] [CONSTRAINT *constraint*]

ADD CONSTRAINT *multifield_constraint*

ALTER COLUMN *field type*[(*size*)]

DROP COLUMN *field*

DROP CONSTRAINT *constraint*

Suppose that you want to add a 10-character text field to store information about the grades of each student. You can do the following:

1. On the **Create**tab, in the **Macros & Code** group, click **Query Design**.

2. Close the **Show Table**dialog box.
3. On the **Design**tab, in the **Query Type** group, click **Data Definition**.

The design grid is hidden, and the SQL view object tab is displayed.

1. Type the following SQL statement:

ALTER TABLE student ADD COLUMN grades TEXT(10)

1. On the **Design**tab, in the **Results** group, click **Run**.

SUB-TOPIC 2:

INTEGRITY CONSTRAINTS OVER RELATIONS

Create a constraint or a relationship

A constraint establishes a logical condition that a field or combination of fields must meet when values are inserted. Constraints are used to limit the type of data that can go into a table or a particular field. For example, a UNIQUE constraint prevents the constrained field from accepting a value that would duplicate an existing value for the field.

A relationship is a type of constraint that refers to the values of a field or combination of fields in another table to determine whether a value can be inserted in the constrained field or combination of fields. You do not use a special keyword to indicate that a constraint is a relationship.

To create a constraint, you use a CONSTRAINT clause in a CREATE TABLE or ALTER TABLE command. There are two kinds of CONSTRAINT clauses: one for creating a constraint on a single field, and another for creating a constraint on multiple fields.

Single-field constraints

A single-field CONSTRAINT clause immediately follows the definition of the field that it constrains, and has the following syntax:

CONSTRAINT *constraint_name* {PRIMARY KEY | UNIQUE | NOT NULL |

REFERENCES *foreign_table* [(*foreign_field*)]

[ON UPDATE {CASCADE | SET NULL}]

[ON DELETE {CASCADE | SET NULL}]}

Multiple-field constraints

A multiple-field CONSTRAINT clause can be used only outside a field-definition clause, and has the following syntax:

CONSTRAINT *constraint_name*

{PRIMARY KEY (*pk_field1*[, *pk_field2*[, ...]]) |

UNIQUE (*unique1*[, *unique2*[, ...]]) |

NOT NULL (*notnull1*[, *notnull2*[, ...]]) |

FOREIGN KEY [NO INDEX] (*ref_field1*[, *ref_field2*[, ...]])

REFERENCES *foreign_table*

[(*fk_field1*[, *fk_field2*[, ...]])] |

[ON UPDATE {CASCADE | SET NULL}]

[ON DELETE {CASCADE | SET NULL}]}

EVALUATION:

1. What is a constraint?
2. What is Data manipulation Language(DML)?

iii. Data Definition Language(DDL)?

READING ASSIGNMENT:

Study the topic 'Integrity constraint" using students' textbook

WEEK NINE

DATE:..

TOPIC: RELATIONAL MODEL

CONTENT:

1. Enforcing integrity constraints
2. Querying relational data

Sub-topic 1

SUB-TOPIC 1: Enforcing integrity constraints

We will consider the following constraints:

Constraints	Meaning
NOT NULL	Enforces a column not to accept NULL values, that is the field cannot be empty.
UNIQUE KEY	Prevents duplicate data
PRIMARY KEY	This constraint uniquely identifies each record in a database table. It always carries a unique value
FOREIGN KEY	This enforces a constraint that prevents actions that would destroy links between tables, and prevents invalid data being inserted.
CHECK	If you **define a CHECK constraint** on a single column it allows only certain values for this column. If you **define a CHECK constraint** on a table it can limit the values in certain columns based on values in other columns in the row.
DEFAULT	allow you to specify a value that the **database** will use to populate fields that are left blank in the input source.

1. SQL NOT NULL Constraint:

The following SQL enforces student_id and Firstname column to not accept NULL values:

CREATE TABLE student

(

Student_id int NOT NULL,

FirstName varchar(255) NOT NULL,

```
LastName,

Age,

Class,

Scores

)
```

1. SQL UNIQUE KEY Constraint:

The following SQL enforces gsm_no column not to accept duplicate values on the teachers table below. Meaning two teachers cannot be sharing one gsm_no

```
CREATE TABLE teachers

(

teacher_id int NOT NULL,

FirstName varchar(255) NOT NULL,

LastName,

Age,

email_address,

gsm_no int(11) UNIQUE KEY

)
```

1. SQL PRIMARY KEY Constraint:

The following SQL constraint uniquely identifies each record in a database table. It must contain unique values. Each table should have a primary key and cannot be NULL. In most cases, the id is used as the primary key so we will enforce it on the teacher_id in the SQL syntax below.

```
CREATE TABLE teachers

(

teacher_id int  NOT NULL  PRIMARY KEY,
```

FirstName varchar(255) NOT NULL,

LastName,

Age,

email_address,

gsm_no int(11) UNIQUE KEY

)

1. SQL FOREIGN KEY Constraint:

The FOREIGN KEY in one table points to a PRIMARY KEY in another table. Let us use a student and teachers table to illustrate this

Student_id	FirstName	Lastname	Age	Class	Scores
1	John	Hassan	15	ss1	20
2	Mary	Adebayo	14	ss1	18
3	Agwu	Ifeoluwa	16	ss2	21

Sub_id	subject	Student_id
1	ICT	2
2	Geography	1
3	Chemistry	2
4	Yoruba	3

CREATE TABLE subject

(

teacher_id int NOT NULL,

FirstName varchar(255) NOT NULL,

LastName,

Age,

email_address,

gsm_no int(11) UNIQUE KEY

)

SUB-TOPIC 2:

QUERYING RELATIONAL DATA

Definition: Queries are the primary mechanism for retrieving information from a database and consist of questions presented to the database in a predefined format. Many database management systems use the Structured Query Language (SQL) standard query format.

There are various SQL query statements which we will look at briefly and what they are meant for.

Fetching Data: SQL SELECT Queries

It is a rare database application that doesn't spend much of its time fetching and displaying data. Once we have data in the database, we want to "slice and dice" it every which way. That is, we want to look at the data and analyze it in an endless number of different ways, constantly varying the filtering, sorting, and calculations that we apply to the raw data. The SQL **SELECT** statement is what we use to choose, or select, the data that we want returned from the database to our application. It is the language we use to formulate our question, or query, that we want answered by the database. We can start out with very simple queries, but the **SELECT** statement has many different options and extensions, which provide the great flexibility that we may ultimately need. Our goal is to help you understand the structure and most common elements of a **SELECT** statement, so that later you will be able to understand the many options and nuances and apply them to your specific needs. We'll start with the bare minimum and slowly add options for greater functionality.

A SQL **SELECT** statement can be broken down into numerous elements, each beginning with a keyword. Although it is not necessary, common convention is to write these keywords in all capital letters. We will focus on the most fundamental and common elements of a **SELECT** statement, namely

- **SELECT**
- **FROM**
- **WHERE**
- **ORDER BY**

The SELECT ... FROM Clause

The most basic SELECT statement has only 2 parts: (1) what columns you want to return and (2) what table(s) those columns come from.

If we want to retrieve all of the information about all of the students in the student table, we could use the asterisk (*) as a shortcut for all of the columns, and our query looks like

SELECT * FROM student

If we want only specific columns (as is usually the case), we can/should explicitly specify them in a comma-separated list, as in; using the student table in the example we had in the previous section, run the query below:

SELECT student_id, FirstName, LastName, Class FROM student

Result

```
Student_id     FirstName     Lastname      Class
1              John          Hassan        ss1
2              Mary          Adebayo       ss1
3              Agwu          Ifeoluwa      ss2
```

Explicitly specifying the desired fields also allows us to control the order in which the fields are returned, so that if we wanted the last name to appear before the first name, we could write it this way:

SELECT student_id, LastName, FirstName, Class FROM student

And the result will be:

```
Student_id     LastName      Firstname     Class
1              John          Hassan        ss1
2              Mary          Adebayo       ss1
3              Agwu          Ifeoluwa      ss2
```

The WHERE Clause

The next thing we want to do is to start limiting, or filtering, the data we fetch from the database. By adding a **WHERE** clause to the **SELECT** statement, we add one (or more) conditions that must be met by the selected data. This will limit the number of rows that answer the query and are fetched. In many cases, this is where most of the "action" of a query takes place.

We can continue with our previous query, and limit it to only those students in ss1:

SELECT student_id, FirstName, LastName, Class FROM student

WHERE Class= 'ss1'

This query will return the following result:

```
Student_id    FirstName     Lastname      Class
1             John          Hassan        ss1
2             Mary          Adebayo       ss1
```

Only students in ss1 will be echoed. If you want to get other classes you declare it in the query and the result will be displayed.

The ORDER BY Clause

Until now, we have been discussing filtering the data: that is, defining the conditions that determine which rows will be included in the final set of rows to be fetched and returned from the database. Once we have determined which columns and rows will be included in the results of our **SELECT** query, we may want to control the order in which the rows appear—sorting the data.

To sort the data rows, we include the **ORDER BY** clause. The **ORDER BY** clause includes one or more column names that specify the sort order. If we return to one of our first **SELECT** statements, we can sort its results by Class with the following statement:

SELECT student_id, FirstName, LastName, Class FROM student

ORDER BY Class

The result of the query will arrange the student in order of their class.

And the result will be:

```
Student_id     LastName      Firstname     Class
1              John          Hassan        ss1
2              Mary          Adebayo       ss1
3              Agwu          Ifeoluwa      ss2
```

EVALUATION:

1. Define Query.
2. What elements can we break SELECT statement into?

iii. Differentiate between **The WHERE Clause** and **The ORDER BY Clause**

Week end Project;

Prepare a student database with the following fields, Firstname, Lastname, Class, scores and run queries on it.