RQF LEVEL 4


GENDD401

NETWORKING AND INTERNET TECHNOLOGIES

# Database Development

TRAINEE'S MANUAL

*October, 2024*

# DATABASE DEVELOPMENT

KOICA Korea International Cooperation Agency

TQUM TVET Quality Management Project

2024

# AUTHOR'S NOTE PAGE (COPYRIGHT)



*Copies available from:*

- *HQs: Rwanda TVET Board-RTB*
- *Web: www.rtb.gov.rw*
- **KIGALI-RWANDA**

Original published version: October 2024

# ACKNOWLEDGEMENTS

The publisher would like to thank the following for their assistance in the elaboration of this training manual:

Rwanda TVET Board (RTB) extends its appreciation to all parties who contributed to the development of the trainer's and trainee's manuals for the TVET Certificate IV in Networking and internet technology, specifically for the module **"GENDD401: Database Development."**

We extend our gratitude to KOICA Rwanda for its contribution to the development of these training manuals and for its ongoing support of the TVET system in Rwanda

We extend our gratitude to the TQUM Project for its financial and technical support in the development of these training manuals.

We would also like to acknowledge the valuable contributions of all TVET trainers and industry practitioners in the development of this training manual.

The management of Rwanda TVET Board extends its appreciation to both its staff and the staff of the TQUM Project for their efforts in coordinating these activities.

**This training manual was developed:**

Under Rwanda TVET Board (RTB) guiding policies and directives

Under Financial and Technical support of

## COORDINATION TEAM

RWAMASIRABO Aimable

MARIA Bernadette M. Ramos

MUTIJIMA Asher Emmanuel

## PRODUCTION TEAM

### Authoring and Review

UZAMUKUNDA Judith

AGIRANEZA Benitha

### Validation

MUKANYANDWI Leoncie

TUYIZERE Alice

### Conception, Adaptation and Editorial works

HATEGEKIMANA Olivier

GANZA Jean Francois Regis

HARELIMANA Wilson

NZABIRINDA Aimable

DUKUZIMANA Therese

NIYONKURU Sylvestre

MINANI Aloys

### Formatting, Graphics, Illustrations, and infographics

YEONWOO Choe

SUA Lim

SAEM Lee

SOYEON Kim

WONYEONG Jeong

MANIRAKORA Alexis

### Financial and Technical support

KOICA through TQUM Project

# TABLE OF CONTENT

**B-tree:** Balanced tree

**CODASYL:** Conference on Data Systems Languages.

**CPU** Power: Central Processing Unit Power

**DBMS:** Database Management System

**DCL:** Data Control Language

**DDL:** Data Definition Language

**DML:** Data Manipulation Language

**DQL:** Data Querry Language

**ERD:** Entity Relationship Diagram

**I/O:** Input/Out put

**MD5:** Message Digest Algorithm 5

**NFRs:** Non-Functional Requirements

**OODBMS:** Object-Oriented Database Management Systems

**RTB:** Rwanda TVET Board

**SHA:** Secure Hash Algorithm

**SQL:** Structured Querry Language

**SSMS:** SQL Server Management Studio

**TCL:** Transaction Control Language

**TQUM** Project**:** TVET Quality Management Project

# INTRODUCTION

This trainee's manual includes all the knowledge and skills required in software development specifically for the module of **"Database Development".** Trainees enrolled in this module will engage in practical activities designed to develop and enhance their competencies. The development of this training manual followed the Competency-Based Training and Assessment (CBT/A) approach, offering ample practical opportunities that mirror real-life situations.

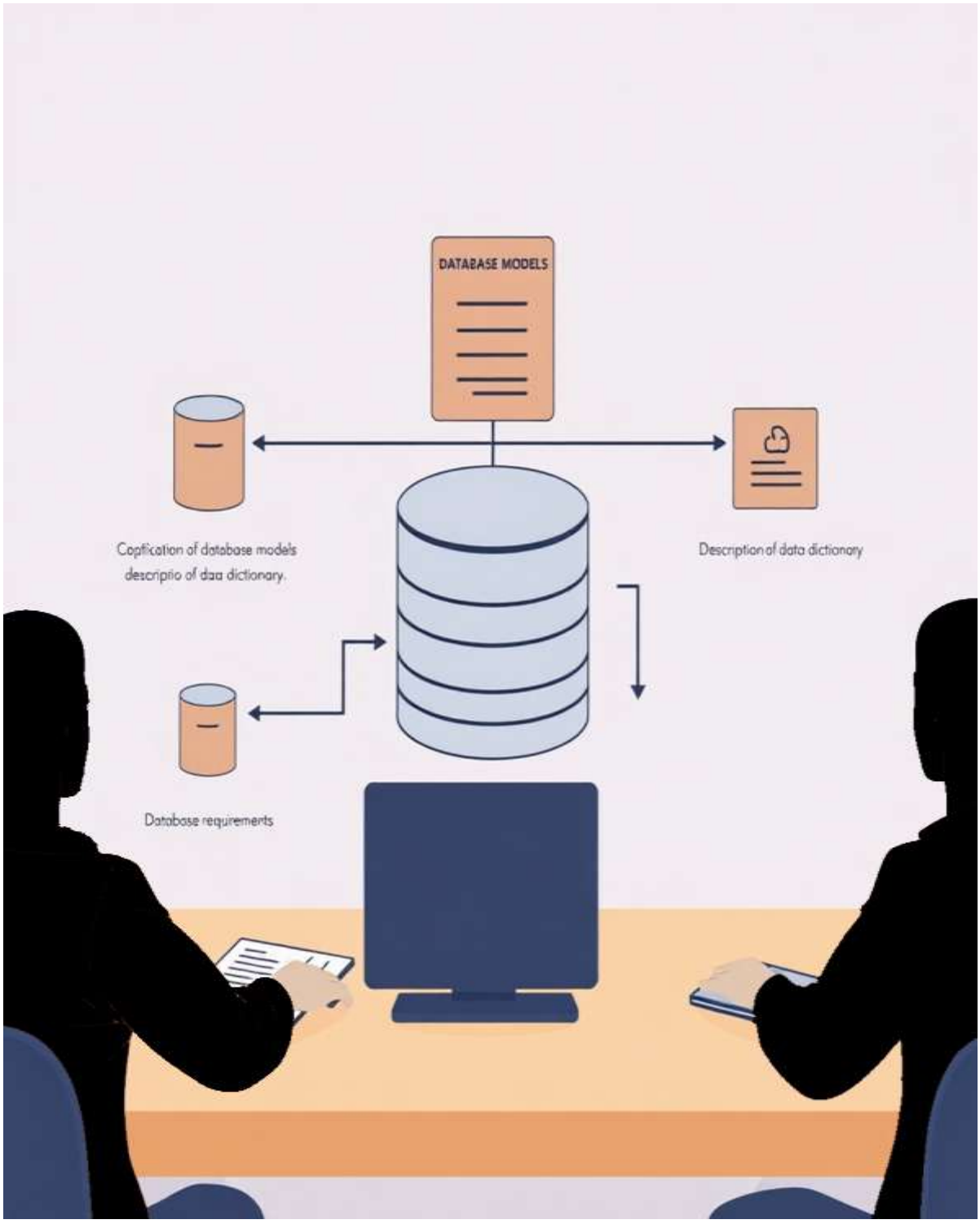
The trainee's manual is organized into Learning Outcomes, which is broken down into indicative content that includes both theoretical and practical activities. It provides detailed information on the key competencies required for each learning outcome, along with the objectives to be achieved.

As a trainee, you will start by addressing questions related to the activities, which are designed to foster critical thinking and guide you towards practical applications in the labor market. The manual also provides essential information, including learning hours, required materials, and key tasks to complete throughout the learning process.

All activities included in this training manual are designed to facilitate both individual and group work. After completing the activities, you will conduct a formative assessment, referred to as the end learning outcome assessment. Ensure that you thoroughly review the key readings and the 'Points to Remember' section.

# MODULE CODE AND TITLE: GENDD401 DATABASE DEVELOPMENT

| |
|---|
| **Learning Outcome 1: Analyse Database** |
| **Learning Outcome 2: Design Database** |
| **Learning Outcome 3: Implement Database** |
| **Learning Outcome 4: Secure Database** |


DATABASE MODELS
Coptication of database models
descriptio of daa dictionary.
Description of data dictionary
Database requirements

**Indicative contents**

**1.1 Description of database fundamental**
**1.2 Description of data dictionary**
**1.3 Identification of database requirements**

## Key Competencies for Learning Outcome 1: Analyze Database

| Knowledge | Skills | Attitudes |
|---|---|---|
| • Description of database Concepts<br>• Identification of database models<br>• Identification of database relationships<br>• Description of determination for database datatypes<br>• Description of data dictionary<br>• Identification database requirements<br>• Identification of data collection methods | • Gathering data based on user requirement<br>• Analysing database requirements | • Having self-confidence while gathering data<br>• Being Organizer while analysing collected data<br>• Having attention to details |

**Duration: 15hrs**

**Learning outcome 1 objectives**:

By the end of the learning outcome, the trainees will be able to:

1. Describe properly Database concepts based on database standards.
2. Identify properly Database models based on database standards
3. Identify properly Database relationship based on database standards
4. Determine correctly Datatypes based on database standards
5. Describe clearly Data dictionary based on database models.
6. Prepare correctly data dictionary according to the user requirements
7. Identify properly database requirements based on user requirements
8. Gather properly data based on customer needs
9. Analyse effectively database requirements based on user requirement.

**Resources**

| Equipment | Tools | Materials |
|---|---|---|
| • Computer<br>• Digital voice recorders (audio or video) | • Video conferencing tools:( Zoom, Microsoft Teams, Skype) | • Internet<br>• Papers<br>• pens |

## Indicative content 1.1: Description of Database Fundamental.

**Duration: 7 hrs**

**Theoretical Activity 1.1.1: Description of database concepts**

**Tasks:**

1: Answer the following questions related to introduction to database

I. Define the following terms:

a. Database
b. Data
c. Information
d. Entities
e. Attributes/Field
f. Records
g. Table
h. Database schema
i. DBMS
j. SQL

II. Where a database can be applied?

2: Write answers on paper, flipchart, blackboard or white board

3: Present your findings to your classmates and trainer

4: Ask questions where necessary

5: Read key reading 1.1.1.

**Key readings 1.1.1: Introduction to database Concepts**

- ✓ **database Concepts**
  - ✓ **Definition of key terms**

**Database**

A database is a structured collection of data that is organized and stored in a way that can be easily accessed, managed, and updated.

Databases are essential components of many software applications, enabling them to store, retrieve, and manipulate data efficiently.

They are used in various scenarios, such as websites, mobile apps, business applications, and more, to handle large volumes of information.

- **Data**

Data refers to raw, unorganized facts and figures. It can be in the form of numbers, text, symbols, or multimedia files. Data has little meant on its own and requires interpretation to become useful.

**Example:** In a database, numbers 1, 2, 3, 4, and 5 are data. By themselves, these numbers do not convey any specific meaning.

- **Information**

Information is data that has been processed, organized, or structured to make it meaningful, relevant, and useful. It is the result of processing raw data to reveal patterns, relationships, or context.

**Example:** If the numbers 1, 2, 3, 4, and 5 represent the sales figures for a company over five months, the statement "The company's sales have been steadily increasing over the past five months" is information derived from the data. Here, the data (numbers) has been processed and contextualized to provide meaningful information about the company's sales performance.

- **Entities**

An entity refers to a distinct object, concept, or thing in the real world that is represented in a database and can be uniquely identified.

Each entity typically corresponds to a table in a relational database or a document in a NoSQL database, and each instance of an entity is represented as a row in the table or a document in the collection.

**For example,** consider an e-commerce system. In this system, entities could include:

Customer, order, product are entities

- **Attributes/Field**

An attribute is a property or characteristic of an entity.

It describes a specific piece of information that can be associated with the entity. Attributes provide details about entities and define the type of data that can be stored in the database.

**Examples:**

- **Customer entity may have attributes like:** Customer ID, Name, Email, Phone Number
- **Product entity may have attributes like:** Product ID, Name, Description, Price, Quantity in Stock

### Records

**A record, also known as a row or tuple,**

Record is a collection of related data items treated as a single unit.

Each record in a database table represents a unique entity, instance, or occurrence of the data being modeled. In simpler terms, a record is a horizontal entry in a database table that contains values for each attribute defined in the table's schema.

| Customer ID | Name | Email | Phone Number |
|---|---|---|---|
| 45 | John Doe | johndoe@example.com | +250788812345 |

record

### Table

a table is a data structure used to organize and store data in a relational database management system (RDBMS).

It consists of rows and columns where each row represents a record and each column represents an attribute or field of the record. Tables are the fundamental building blocks of a relational database.

Here is an example of a simple "Customer" table:

| CustomerID | Name | Email | Phone Number |
|---|---|---|---|
| 1 | John Doe | johndoe@example.com | +250788812345 |
| 2 | Jane Smith | janesmith@example.com | +250788867891 |

### Database schema

A database schema is a blueprint or design that defines the structure of a database system, including its tables, columns, data types, relationships, and constraints.

It provides a logical view of the entire database, outlining how data is organized and how relationships between data elements are handled. The schema is essential for ensuring data integrity, consistency, and efficiency in a database.

Tables, Columns, Data Types, Constraints, Relationships, Indexes.

- **DBMS**

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

  - MySQL
  - PostgreSQL
  - Oracle
  - Microsoft SQL Server
  - MongoDB
  - Cassandra
  - SQLite

✓ **Applications of database**

Databases have numerous applications across various industries and domains. Here are some common applications of databases:

- **Business:** Databases are widely used in businesses for managing and storing large amounts of structured data. They help in organizing and retrieving data efficiently, enabling businesses to make informed decisions, analyze trends, and improve operations.
- **E-commerce:** Databases are essential for online retail platforms to store product information, customer details, and transaction records. They enable efficient inventory management, order processing, and personalized customer experiences.
- **Banking and Finance:** Databases are crucial for managing financial transactions, customer accounts, and regulatory compliance. They ensure data integrity, security, and enable real-time processing of financial transactions.
- **Healthcare:** Databases are used in healthcare systems to store patient records, medical histories, and diagnostic data. They facilitate efficient management of patient information, enable quick access to medical records, and support research and analysis.
- **Education:** Databases are utilized in educational institutions to store student data, course information, and academic records. They help in managing student information, tracking progress, and generating reports.

- **Logistics and Supply Chain:** Databases play a vital role in tracking inventory, managing logistics operations, and optimizing supply chain processes. They enable efficient order management, inventory control, and forecasting.
- **Social media:** Databases are the backbone of social media platforms, storing user profiles, posts, comments, and interactions. They enable quick retrieval of content, personalized recommendations, and targeted advertising.
- **Research and Science**: Databases are used in scientific research to store experimental data, research findings, and scholarly articles. They facilitate data sharing, collaboration, and enable efficient analysis and discovery.

**Theoretical Activity 1.1.2: Identification of database models**


**Tasks:**

1: you are asked to answer the following questions:

I. What do you understand by term database model?

II. Explain types of database model

2: Present the findings to the whole class

3: Participate in grouping and agreeing on correct answers

4: Ask questions where necessary.

5: For more clarification, read the key readings 1.1.2.

**Key readings 1.1.2.: Identification of database Models**

- **Database model**

A database model refers to the logical structure, representation or layout of a database and how the data will be stored, managed and processed within it. It helps in designing a database and serves as blueprint for application developers and database administrators in creating a database.

There are several commonly used database models, each with its own approach to organizing and structuring data. Here are four major database models:

✓ **Types of Database models**

- **Relational Model**

  The relational model is the most widely used database model. It organizes data into tables with rows and columns, where each table represents an entity or relationship. The tables are connected through keys, enabling efficient querying

and manipulation of data. Relational database management systems (RDBMS) like MySQL, PostgreSQL, and Oracle Database are based on this model.

- **Hierarchical Model**

  The hierarchical model organizes data in a tree-like structure, where each record has a parent-child relationship. It is suitable for representing one-to-many relationships. However, it can be inflexible when dealing with complex or changing data relationships. IBM's Information Management System (IMS) is an example of a hierarchical database model.

- **Network Model**

  The network model is an extension of the hierarchical model that allows for more complex relationships. It represents data as a collection of records connected through pointers, forming a network-like structure. This model is more flexible than the hierarchical model but can be complex to implement and maintain. Integrated Data Store (IDS) and Integrated Database Management System (IDMS) are examples of network database systems.

- **Object-Oriented Model**

  The object-oriented model represents data as objects, combining data and behavior into a single entity. It supports encapsulation, inheritance, and polymorphism, allowing for complex data structures and relationships. Object-oriented database management systems (OODBMS) like MongoDB and Apache Cassandra are based on this model.

**Theoretical Activity 1.1.3: Identification of database Relationships**


**Tasks:**

1: you are asked to answer the following questions related to database relationship

I. What do you understand by database relationship?

II. Identify types of database relationship

2: Present your findings to your classmates and trainer

3: Ask questions where necessary

4: For more clarification, read the key readings 1.1.3.

**Key readings 1.1.3.: Identification of database Relationships**

- **Database Relationship**

In a relational database, relationships are established between tables to define how the data in one table is related to the data in another table. There are three main types of relationships commonly used in relational databases:

✓ **Types of database relationship**

- **One-to-One (1:1) Relationship**:

  In a one-to-one relationship, each record in one table is associated with exactly one record in another table, and vice versa. This type of relationship is typically used when the related data is distinct and can be separated into two tables for better organization.

  For example, a "Person" table may have a one-to-one relationship with an "Address" table, where each person has a unique address.

- **One-to-Many (1: N) Relationship:**

  In a one-to-many relationship, a record in one table can be associated with multiple records in another table, but each record in the second table is associated with only one record in the first table. This is the most common type of relationship.

  For example, in a "Customer" table, each customer can have multiple orders in an "Order" table, but each order is associated with only one customer.

- **Many to One (M:1) Relationship:**

  A many-to-one relationship in a database refers to a type of association between two entities where many instances of one entity (often referred to as the "child" entity) are related to one instance of another entity (often referred to as the "parent" entity).

  **A many-to-one relationship in a database refers to a type of association between two entities where many instances of one entity (often referred to as the "child" entity) are related to one instance of another entity (often referred to as the "parent" entity).**

  **Example:** Consider a school database:

  Students (Child Entity)

  Classrooms (Parent Entity)

  Here, many students can be assigned to one classroom. Thus, the relationship between Students and Classrooms is many-to-one.

- **Many-to-Many (N: N) Relationship:** In a many-to-many relationship, multiple records in one table can be associated with multiple records in another table. To

represent this relationship, an intermediate table, known as a junction or join table, is used. This table contains foreign keys from both tables, establishing the relationship. For example, in a bookstore database, a "Book" table can have a many-to-many relationship with an "Author" table through a "Book Author" junction table, as multiple books can have multiple authors, and authors can write multiple books.

**Theoretical Activity 1.1.4: Description of determination for data types**


**Tasks:**

1: Read carefully and answer the following questions:

I. What do you understand by datatype?
II. What are main categories of datatype?

2: Present the findings to the whole class

3: Participate in grouping and agreeing on correct answers

4: Ask questions where necessary.

5: For more clarification, read the key readings 1.1.4.

**Key readings 1.1.4.: Determination of datatypes**

- **Datatype**

Datatype is a keyword used in database to identify type of data a column will store

✓ **Categories of datatypes**

**Character**

| Data type | Description |
|---|---|
| CHAR (size) | A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1 |
| VARCHAR (size) | A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535 |
| BINARY(size) | Equal to CHAR (), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1 |

| | |
|---|---|
| VARBINARY(size) | Equal to VARCHAR (), but stores binary byte strings. The *size* parameter specifies the maximum column length in bytes. |
| TINYBLOB | For BLOBs (Binary Large Objects). Max length: 255 bytes |
| TINYTEXT | Holds a string with a maximum length of 255 characters |
| TEXT(size) | Holds a string with a maximum length of 65,535 bytes |
| BLOB(size) | For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data |
| MEDIUMTEXT | Holds a string with a maximum length of 16,777,215 characters |
| LONGTEXT | Holds a string with a maximum length of 4,294,967,295 characters |
| ENUM(val1, val2, val3, ...) | A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them |
| SET(val1, val2, val3, ...) | A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list |

- **Numeric data types:**

| Data type | Description |
|---|---|
| BIT(*size*) | A bit-value type. The number of bits per value is specified in *size*. The *size* parameter can hold a value from 1 to 64. The default value for *size* is 1. |
| TINYINT(*size*) | A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255) |
| BOOL | Zero is considered as false, nonzero values are considered as true. |
| BOOLEAN | Equal to BOOL |
| SMALLINT(*size*) | A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255) |
| MEDIUMINT(*size*) | A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255) |

| | |
|---|---|
| INT(*size*) | A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255) |
| INTEGER(*size*) | Equal to INT(size) |
| BIGINT(*size*) | A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255) |
| FLOAT(*size*, *d*) | A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions |
| FLOAT(*p*) | A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE() |
| DOUBLE(*size*, *d*) | A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter |
| DECIMAL(*size*, *d*) | An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0. |
| DEC(*size*, *d*) | Equal to DECIMAL (size,d) |

**Note:** All the numeric data types may have an extra option: UNSIGNED or ZEROFILL. If you add the UNSIGNED option, MySQL disallows negative values for the column. If you add the ZEROFILL option, MySQL automatically also adds the UNSIGNED attribute to the column.

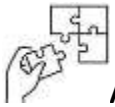
- **Date and Time data types:**

| Data type | Description |
|---|---|
| DATE | A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31' |

| | |
|---|---|
| DATETIME(*fsp*) | 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'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time |
| TIMESTAMP(*fsp*) | A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition |
| TIME(*fsp*) | 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. MySQL 8.0 does not support year in two-digit format. |

**Points to Remember**

- Database always used to store structured data for easy access, management, and updating.
- You should apply databases in various fields like business, e-commerce, banking, healthcare, and education to ensure organized data handling.
- To ensure the safety Store and process of information in a relational database you are recommended to use SQL (Structured Query Language).
- Processed data turns into useful information.
- While interacting with database, Database Management System (DBMS) is always used to create and manage data efficiently.
- You are recommended to use different database models: Relational, Hierarchical, Network, and Object-Oriented.
- You should associate related entities using relationships in relational databases
- Always identify clearly the type of relationship between entities to determine whether it is one-to-one, one-to-many, many-to-one or many-to-many.
- Data type must be assigned to each column to specify what kind of data it will store.
- The main categories of datatype in database should focus on numeric, character, data and time.



**Application of learning 1.1.**

XYZ University is a large institution with thousands of students, faculty members, and administrative staff. To efficiently manage the records of students, faculty, and courses, the university decided to implement a student information management system (SIMS). The goal of the system is to store, organize, and retrieve relevant data related to student enrolment, faculty assignments, and course offerings. As a database analyst you are asked to analyse the above case study and find out elements that will be stored in database.

## Indicative content 1.2: Description of Data Dictionary

**Duration: 7 hrs**

### Theoretical Activity 1.2.1: Description of data dictionary

**Tasks:**

1: Read carefully and answer the following questions related to the data dictionary

1) What is a data dictionary?
2) What are elements of data dictionary?

2: Present your findings to your classmates and trainer

3: Ask questions where necessary

4: For more clarification, read the key readings 1.2.1.

**Key readings 1.2.1.: Description of data dictionary**

- **Data dictionary**

A data dictionary is a structured collection of metadata that provides detailed information about the data stored within the database. It serves as a reference guide for understanding the structure, organization, and characteristics of the data.

✓ **Elements of data dictionary**

- **Field Name:** Specifies the name of the column in the table.
- **Data Type:** Indicates the type of data that can be stored in the column (e.g., INT for integers, VARCHAR for variable-length strings, DATE for dates).
- **Length:** Represents the maximum length of the data element, applicable to string data types.
- **Constraints:** Describes any constraints applied to the column (e.g., Primary Key, Foreign Key, Not Null, and Check).
- **Description:** Provides a brief description of the data element for better understanding.

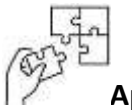
A data dictionary would include additional details such as relationships with other tables, indexes, default values, and more specific descriptions. The level of detail in a data dictionary can vary based on the complexity of the database and the specific needs of the organization.

**Sample data dictionary**

| Field Name | Data Type | Length | Constraints | Description |
|---|---|---|---|---|
| Employee ID | INT | 4 | Primary Key | Unique identifier for each employee |
| First Name | VARCHAR | 50 | Not Null | First name of the employee |
| Last Name | VARCHAR | 50 | Not Null | Last name of the employee |
| Birthdate | DATE | 8 | Not Null | Date of birth of the employee |
| Gender | CHAR | 1 | Not Null | Gender of the employee (M/F) |
| Is Active | BOOLEAN | 1 | Not Null | Indicates if the employee is currently active (1 for active, 0 for inactive) |

**Points to Remember**

- A data dictionary is a structured collection of metadata that provides detailed information about the data stored within the database.
- While display data dictionary ensure that it contains its essential elements, including table names, attribute names, data types (with their sizes), and constraints.



**Application of learning 1.2.**

The data dictionary below is generated by the MySQL DBMS from the database of A&U Trading Company, which is located in Gasabo district, Giti sector, Muhuro village. As a newly hired database analyst for the company, you are asked to identify and analyze the key

elements within this data dictionary produced by the DBMS.

**a&u_trading_company_database**

## customer_table

| Column | Type | Null | Default | Links to |
|---|---|---|---|---|
| Customerid *(Primary)* | int(7) | No | | |
| CustomerFname | varchar(35) | No | | |
| CustomerLname | varchar(25) | No | | |
| CustomerPhone | varchar(14) | No | | |
| CustomerSex | varchar(1) | No | | |

## employee_table

| Column | Type | Null | Default | Links to |
|---|---|---|---|---|
| EmployeeId *(Primary)* | int(7) | No | | |
| EmployeeFname | varchar(25) | No | | |
| EmployeeLname | varchar(35) | No | | |
| EmployeeEmail | varchar(18) | No | | |
| EmployeeSex | varchar(1) | No | | |
| EmployeeUserName | varchar(38) | No | | |
| EmployeePassword | varchar(34) | No | | |
| EmployeePhone | varchar(13) | No | | |

## orders_table

| Column | Type | Null | Default | Links to |
|---|---|---|---|---|
| OrdersId *(Primary)* | int(7) | No | | |
| OrdersDate | date | Yes | *NULL* | |
| Customerid | int(5) | Yes | *NULL* | customer_table -> Customerid |
| Employeeid | int(6) | Yes | *NULL* | employee_table -> EmployeeId |

## Indicative content 1.3: Identification of Database Requirements

**Duration: 5 hrs**

### Theoretical Activity 1.3.1: Identification of data collection methods

**Tasks:**

1: You are requested to answer the following questions related to the description of key concepts:

a) How does data collection contribute to the successful development of a database?
b) What are the primary methodologies for data collection in database design?

2: Provide the answer for the asked questions

3: For more clarification, read the key readings 1.3.1.1. In addition, ask questions where necessary.

**Key readings 1.3.1.: Identification of data collection methods**

- **data collection methods**

✓ **Data collection**

Data collection is the cornerstone of database development. It provides the essential blueprint by identifying the specific information to be stored, the relationships between data elements, and the intended use of the database. Through careful analysis of collected data, database designers can create a structure that accurately represents the real-world entities and processes, ensuring data integrity, efficiency, and effective retrieval.

✓ **Methods to collect data**

- **Interview**

**Interview:** Interviews involve direct conversations with users, stakeholders, or developers to gather detailed information about their experiences, needs, or perspectives. Interviews can be structured or unstructured and can be conducted in person, over the phone, or through video conferencing.

- **Types of interview**

**Structured interviews** involve a predetermined set of questions asked in a standardized manner. The questions are formulated in advance, and the interviewer asks the same set of questions to all respondents.

**Unstructured interviews** are informal and open-ended. There is no predetermined set of questions, allowing the conversation to flow naturally. The interviewer explores topics in-depth based on the participant's responses.
**Semi-structured interviews**: in this type of interview; The interviewer has a predefined list of questions but can deviate from the script to explore specific points in more detail based on the participant's responses.

- **Tools used in perform interview**

- Interview guides
- Digital voice recorders (audio or video)
- Video conferencing tools: Zoom, Microsoft Teams, Skype

**Documentation**

It involves gathering and analyzing existing documentation to understand the system architecture, requirements, design decisions, or codebase. Documentation is essential for knowledge transfer, maintaining codebases, and ensuring proper understanding of the software system.

- **Tools used to carry out Documentation**
  - Data Extraction Tools: Microsoft Word, Excel (for summaries)
  - Adobe Acrobat (PDF annotations)
  - Spreadsheet Software (Microsoft Excel, Google Sheets)
  - Google Drive

**Questionnaire**

A questionnaire is a structured set of questions designed to collect specific information from respondents. Questionnaires can be administered in written or electronic form, and respondents provide answers to predefined questions.

Analyst may use open questions or close questions
**Open questions** are questions that do not restrict respondents to specific answer choices. Instead, they allow respondents to provide detailed, free-form responses in their own words.
**Closed questions** are questions that offer respondents specific answer options to choose from. These options can include multiple-choice selections, yes/no responses, or other predefined categories.

- **Tools used to carry out Questionnaire**
  - Survey Software: Google Forms, Survey Monkey, Type form.
  - Distribution Channels: Email, social media, or paper forms
  - Excel, Google Sheets

- **Observation**

Observation is a data collection technique where researchers or developers directly observe and analyze how people or processes behave in real-life situations. This can involve watching users interact with software applications, observing workflow processes, or studying user behavior in specific contexts.

- **Tools used to carry out observation**
  - Notebooks
  - tablets
  - Digital cameras
  - smartphones,
  - Specialized observation software: Noldus Observer, Mangold INTERACT

**Practical Activity 1.3.2: Collecting data**


**Task:**

1: Read the read key readings 1.3.2 about creation of database requirements.

2: Answer the question below on a paper:

> your school need to enhance the technology that will help to manage internet accessibility in different trades. To implements that technology school needs an application that will store and executes the user information. As database analyst, you are requested to collect data that will guide in database development by using interview, observation, documentation and Questionnaire.

3: Present your findings to the class.

4: Ask the trainer for clarification if any.

**Key readings 1.3.2: Collecting data**

- **Collecting data**

✓ **Steps followed to collect data using interview as selected method.**

- **Define Objectives**: Identify the purpose of the interview (e.g., gather insights on user experience).
- **Select Participants:** Choose a representative sample of individuals (e.g., network administrators).
- **Prepare Questions:** Develop open-ended questions that encourage detailed responses.
- **Conduct Interviews:** Schedule and conduct the interviews, ensuring a comfortable environment.
- **Record Responses:** Take notes or use audio recording (with permission) for accuracy.

**The following Tools may be used to carry out interview:**

- **Interview Guide:** A structured list of questions.
- **Recording Device:** Audio recorder or smartphone app.
- **Note-taking App:** Software like Evernote or OneNote.

✓ **Steps followed to collect data using Questionnaires as selected method.**

- **Define Objectives:** Clarify what information you want to collect.
- **Design Questionnaire:** Create questions that are clear and concise, mixing multiple-choice and open-ended formats.
- **Pilot Test:** Test the questionnaire with a small group to identify issues.
- **Distribute Questionnaire:** Use online platforms or paper forms to reach participants.
- **Collect Responses:** Monitor the response rate and follow up as needed.

**The following Tools may be used to carry out Questionnaire:**

- **Survey Software:** Google Forms, Survey Monkey, or Type form.
- **Distribution Channels**: Email, social media, or paper forms
- Excel, Google Sheets

✓ **Steps followed to collect data using observation as selected method.**

- **Define Objectives:** Determine what behaviors or processes you want to observe (e.g., how users interact with network equipment).

- **Select Setting:** Choose the location where observations will take place (e.g., server room).
- **Develop an Observation Checklist:** Create a list of specific behaviors or events to observe.
- **Conduct Observations:** Spend time observing the environment and recording findings.
- **Analyze Data:** Summarize and interpret the observations.

**The following Tools may be used to carry out observation:**

- **Observation Checklist:** A structured list to guide observations.
- **Field Notes:** Notebook or digital tool for recording observations.
- **Video Recording:** Cameras (with consent) to capture interactions.

✓ **Steps followed to collect data using Documentation as selected method.**

- **Identify Sources:** Determine what documents are relevant (e.g., network configurations, policies).
- **Collect Documents:** Gather existing materials from different sources (e.g., internal databases, manuals).
- **Review and Analyze:** Assess the documents for relevant information related to your research objectives.
- **Summarize Findings:** Extract key insights and organize them for reporting.

**The following Tools may be used to carry out Documentation:**

- Data Extraction Tools: Microsoft Word, Excel (for summaries)
- Adobe Acrobat (PDF Readers)
- Spreadsheet Software (Microsoft Excel, Google Sheets)
- Google Drive

**Points to Remember**

- Always use interviews as the primary method of data collection to obtain efficient data for the database system.
- To ensure the accurate information You should conduct interview with key users of the current system or future users who will manage the information in the new application/system.
- Ensure that the analyst conducts interviews with both main users and those responsible for managing the database system.
- You are recommended to use questionnaires to collect data from a large number of people who will use the services provided by the database system.

- Prepare clear questions that will guide interviewer.
- It is mandatory to organize tools and materials that relate with the methods to be used in data collection.

**Theoretical Activity 1.3.3.: Identification of database requirements**


**Tasks:**

1: You are requested to answer the following questions related to identification of database requirement

I. What do you understand by database requirements?
II. What are the types of database requirement?

2: Provide the answer for the asked questions

3: For more clarification, read the key readings 1.3.3.

4: In addition, ask questions where necessary.

**Key readings 1.3.3: Identification of database requirements**

- **Database requirements**

Database requirements are the detailed specifications and criteria that a database must meet to fulfil the needs of a particular application, organization, or project.

✓ **Types of database requirements**

**Functional requirement**

Functional requirements define the specific actions and operations the database system must perform. They outline the system's capabilities and functionalities. Examples of functional requirements include:

- **Data storage:** Defining data types, structures, and formats for efficient storage.
- **Data retrieval:** Specifying how data can be accessed, searched, and extracted.
- **Data update:** Determining how data can be modified, corrected, or replaced.
- **Data deletion:** Establishing procedures for removing outdated or unnecessary data.
- **Data processing:** Indicating calculations, transformations, or manipulations required on the data.
- **Reporting:** Specifying the format and content of information to be generated from the database.

**Non-Functional Requirements (NFRs)**

Non-functional requirements describe the qualities and characteristics of a database system, rather than its specific functions. They focus on how well the system performs its functions. Key NFRs include:

- **Performance:** Defining response times, throughput, and scalability expectations.
- **Security:** Specifying access controls, data encryption, and protection measures.

- **Reliability:** Determining system availability, fault tolerance, and recovery procedures.
- **Maintainability:** Indicating ease of modification, upgrade, and support.
- **Usability:** Specifying user interface requirements and ease of use.
- **Compatibility:** Defining compatibility with other systems and software.
- **Portability:** Indicating the ability to transfer the database to different platforms.
- **Scalability:** Determining the system's ability to handle increasing data volumes and user loads.

**Points to Remember**

- Clearly define the database requirements to ensure that the system meets the necessary specifications for the intended application.
- All Specified Database requirements must be classified into functional and non-functional categories.
- to guide the design and implementation process requires to Identify and classify the database requirements into functional (what the system should do) and non-functional (performance, scalability, etc.)

**Practical Activity 1.3.4: Analysing database requirement**


**Task:**

1: Read the read key readings 1.3.4 about creation of database requirements.

2: Answer the question below on a paper:

A university is planning to develop a new Student Information System (SIS) to manage student records, academic performance, and enrolment. The system will be used by students, faculty, and administrative staff.

**You are tasked to:**

Identify the potential database requirements for this SIS. Consider both functional and non-functional requirements.

Hint: Think about the data that needs to be stored, the operations that need to be performed, and the system's overall performance and security needs.

3: Present your findings to the class or your colleagues.

4: Ask the trainer for clarification if any.

**Key readings 1.3.4: Analysing database requirements**

- **Database requirements Analysis**
  - ✓ **Steps followed while identifying requirements.**

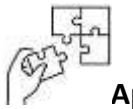
- **Define the System's Purpose:** Clearly articulate the goals and objectives of the database system.
- **Identify Stakeholders:** Determine who will use the system and what their needs are.
- **Analyze Gathered Information:** Collect data about the system's functionalities, processes, and data elements.
- **Define Functional Requirements:** Specify the actions the system must perform.
- **Define Non-Functional Requirements:** Specify the system's qualities and characteristics.
- **Prioritize Requirements:** Rank requirements based on importance and feasibility.
- **Validate Requirements:** Ensure requirements are clear, complete, and consistent.

Throughout the process, keep the following in mind:

- **User-centric approach:** Focus on user needs and experiences.
- **Flexibility and scalability:** Design the database to accommodate future growth.
- **Data quality and integrity:** Ensure data accuracy and consistency.
- **Security and privacy:** Protect sensitive information.
- **Performance optimization:** Consider factors affecting system speed and responsiveness.
- **Cost-effectiveness:** Balance requirements with budget constraints.
- **Maintainability:** Design for ease of modification and updates.

**Points to Remember**

- Ensure that all requirements are clear, complete, and consistent before proceeding with the design and implementation.
- Express clearly the goals and objectives of the database system before implementation to check if it meets with requirement.
- Always Determine who will use the system (the users of the system) to identify the target users and their roles.
- Analyze the data collected during the research phase to understand the needs.
- Before deigning database, you should Specify the actions and operations the system will perform.



**Application of learning 1.3.**

A new online clothing store XYZ is launching and requires a robust database to manage its operations. The store needs to store information about products, customers, orders, and inventory.

**Task:**

You are tasked to design the database structure for the online clothing store. Identify the **critical data elements** to be stored, the **essential database operations** required, and the **performance and security requirements** necessary to ensure a successful online shopping experience.

## Learning outcome 1 end assessment

**Written assessment**

1) Match the terms in column A to their description in column B.

| Answers | Column A | Column B |
|---|---|---|
| 1…… | 1. Database | A) A person, place, thing, or event. |
| 2…… | 2. Data | B) A structured collection of related data. |
| 3…… | 3. Information | C) The logical structure of a database, defining entities, attributes, and relationships |
| 4…….. | 4. Entity | D) A characteristic or property of an entity. |
| 5…….. | 5.Attribute/Field | E) A collection of related data organized in a structured format |
| 6…….. | 6. Record | F) A row in a table, representing an instance of an entity. |
| 7…….. | 7. Table | G) Software used to manage databases. |
| 8…….. | 8. Database schema | H) A language used to interact with databases. |
| 9…….. | 9. DBMS | I) A collection of related records. |
| 10…….. | 10. SQL | J) Raw facts and figures. |

2) Read the following statements and circle the letter of the correct answers:

I. Which database model is based on a hierarchical structure?
   a. Relational database
   b. Hierarchical database
   c. Network database
   d. Object-oriented database

II. What is the relationship between a department and its employees?
   a. One-to-one
   b. One-to-many
   c. Many-to-one
   d. Many-to-many

III. Which data type would be suitable for storing a person's age?
   a. Character
   b. Number
   c. Date
   d. None

3) Determine the type of database relationship below:
   a) A table of employees and a table of their corresponding addresses.
   b) A table of customers and a table of their orders.
   c) A table of books and a table of their authors.
4) What are the different methods used to collect data for database design?
5) What are the key elements of a data dictionary?
6) Identify the two types of database requirements and explain how they differ. Provide examples of each type.

**Practical assessment**

A hospital ADB is planning to develop a new Hospital Information System (HIS) to manage patient records, appointments, treatments, and billing. The system will be used by doctors, nurses, administrative staff, and patients.

**Task:**

1. Identify the key database terms involved in this scenario, Determine the applications of the HIS database, analyze the solution gained from using a HIS database, Identify the most suitable database model for this scenario, what are the relationships between key entities in the HIS, Identify the data types required for storing various information in HIS database.
2. **Identify and categorize the functional and non-functional requirements** for the HIS.
3. **Identify a data dictionary** for the HIS, defining the entities, attributes, and data types to be used in the database.

**END**

## References


Abraham Silberschatz, H. F. (2010). Database System Concepts. New York: McGraw-Hill Education.

Authors. (2000). Design Database. Indianapolis, Indiana, USA: Sams Publishing.

Carlos Coronel, S. M. (2016). Database Systems: Design, Implementation, & Management. Boston: Cengage Learning.

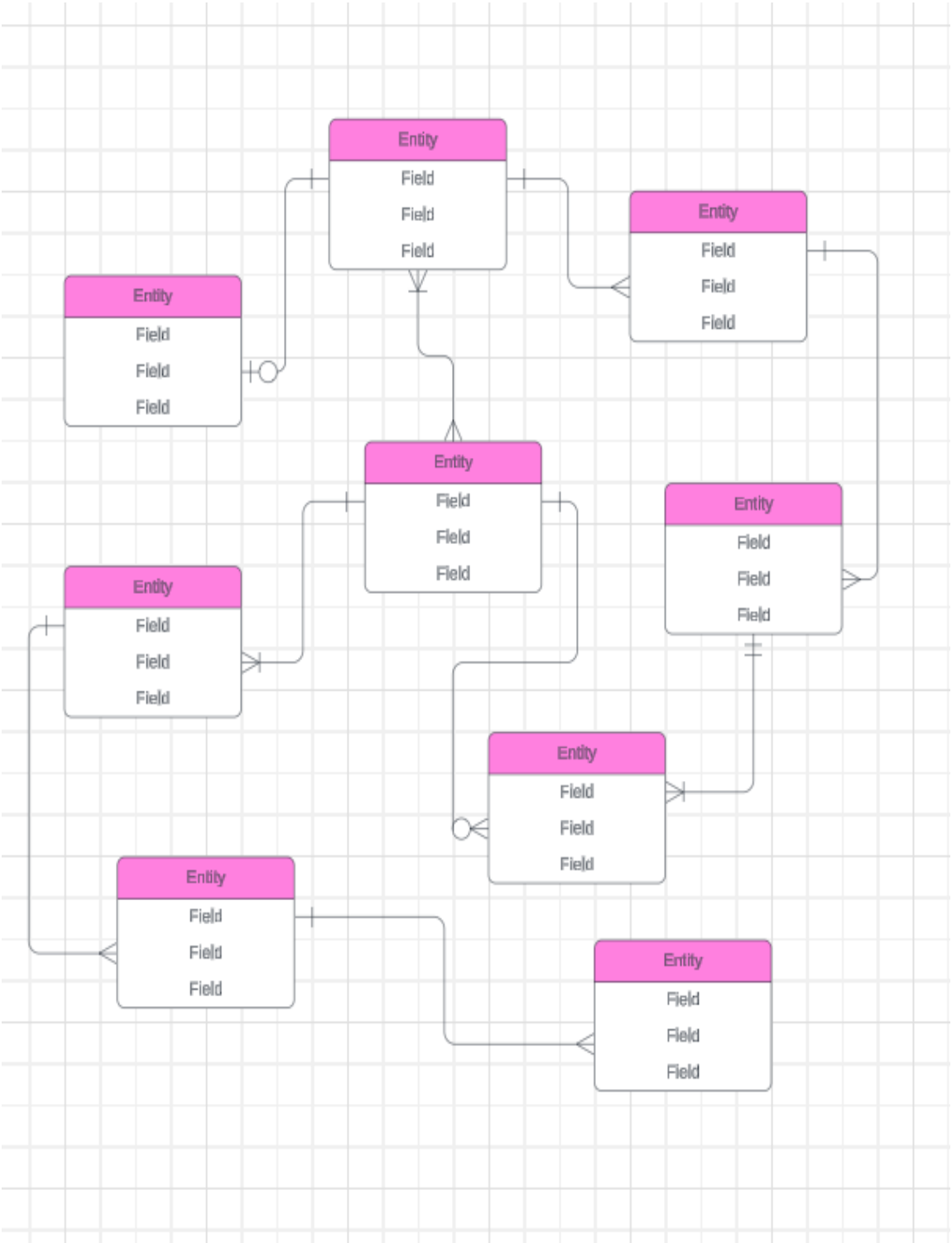
https://www.relationaldbdesign.com/database-design/module4/introrequirementsanalysis.php#:~:text=Requirements%20Analysis%20is%20the%20stage,data%20needs%20to%20be%20accessed.

https://study.com/academy/lesson/what-is-an-entity-in-a-database.html

Ramez Elmasri, S. B. (2015). Fundamentals of Database Systems. Boston: Pearson.

Sciore, E. (2008). Database Design and Implementation. Hoboken, New Jersey, USA: Wiley.

Watt, A. (2014). Database Design – 2nd Edition. Victoria, British Columbia, Canada: BCcampus Open Education.


Entity
Field
Field
Field
Entity
Field
Field
Field
Entity
Field
Field
Field
Entity
Field
Field
Field
Entity
Field
Field
Field
Entity
Field
Field
Field
Entity
Field
Field
Field
Entity
Field
Field
Field
Entity
Field
Field
Field

| Indicative contents |
|---|
| **2.1 Description of database schema**<br>**2.2 Design of conceptual database schema**<br>**2.3 Design of logical database schema**<br>**2.4 Optimization of database**<br>**2.5 Design of Physical database schema** |

## Key Competencies for Learning Outcome 2 : Design Database

| Knowledge | Skills | Attitudes |
|---|---|---|
| • Description of database schema<br>• Description of conceptual database schema<br>• Description of logical database schema<br>• Description of physical database schema | • Designing conceptual database schema<br>• Designing logical database schema<br>• Optimizing database<br>• Designing Physical database schema | • Having creativity skills in designing database schemas<br>• Being Innovative in designing database schemas<br>• Having adaptability skills in designing database schemas<br>• Being Organizer while designing database schemas<br>• Being Flexible while optimizing database<br>• Having self-confidence while optimizing database |

**Duration: 25 hrs**

**Learning outcome 2 objectives**:

By the end of the learning outcome, the trainees will be able to:

1. Describe properly database schema based on DBMS
2. Describe properly conceptual database schema based on DBMS
3. Design correctly ERD based on system requirements
4. Design properly logical database schema based on system requirements
5. Implement effectively Database optimization based on database schema
6. Create appropriately Physical Database Schema based on the Physical Data Model

 **Resources**

| Equipment | Tools | Materials |
|---|---|---|
| • Computer | • E-Draw max<br>• MySQL set up | • Internet |

## Indicative content 2.1: Description of Database Schema

**Duration: 2 hrs**

**Theoretical Activity 2.1.1: Description of database schema**

**Tasks:**

1: You are requested to answer the following questions related to database schema

i. What do you understand with database schema?
ii. Identify types of database schema
iii. What are three levels of database abstraction
iv. Identify types of data independence

2: Present your findings to your classmates and trainer

3: Ask questions where necessary

4: For more clarification, read the key readings 2.1.1.

**Key readings 2.1.1.: Description of database schema**

- **Introduction of database schema**

A database schema is a logical blueprint or design that defines the structure, organization, and relationships of a database. It describes the tables, columns, data types, constraints, and relationships between the data elements within the database.
The database schema provides a conceptual representation of how the data is organized and stored. It defines the entities (tables) and attributes (columns) that make up the database, as well as the relationships between these entities.

The schema defines the rules and constraints that govern the data stored in the database, such as primary key constraints, foreign key relationships, unique constraints, and data validation rules. It also specifies the data types and lengths of each attribute, ensuring data integrity and consistency.
In addition to the structure and constraints, the schema may also include views, indexes, stored procedures, and other database objects that facilitate data retrieval, manipulation, and management.

### ✓ Types of database schema

While the term schema is broadly used, it is commonly referring to three different schema types a conceptual database schema, a logical database schema, and a physical database schema.

#### Conceptual database schema

Conceptual schemas offer a big-picture view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements.

**Features of Conceptual database schema are:**

- Entity Names
- Relationships

#### Logical database schema

Logical database schemas are less abstract, compared to conceptual schemas. They clearly define schema objects with information.

The logical schema defines the structure of the data itself and the relationships between the various attributes, tables, and entries.

**Features of logical database schema are:**

- Entity Names
- Entity Relationships
- Attributes
- Primary Keys
- Foreign Keys

#### Physical database schema

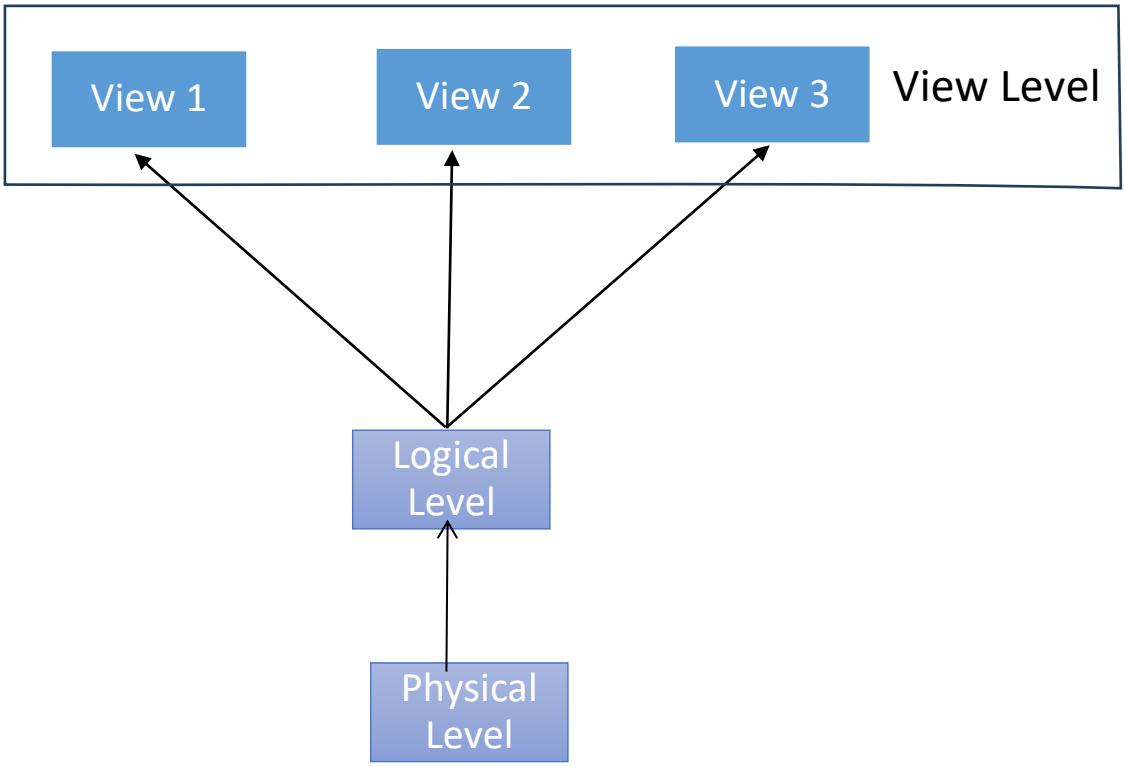
Physical database schemas represent how data is stored on the device hard disk. This is the actual code that will be used to create the structure of your database which includes any tables and how they relate to each other.

**Features of physical database schema are:**

- Primary Keys
- Foreign Keys
- Table Names
- Column Names
- Column Data Types

### ✓ Data abstraction levels

In database management, data abstraction refers to the process of simplifying complex data structures and operations into more manageable and understandable forms. There are three commonly recognized levels of data abstraction:




- **Physical Level**

This is the lowest level of data abstraction, which deals with the physical storage and representation of data. It focuses on the details of how data is stored on the storage media, such as hard disks or solid-state drives. It involves aspects like file organization, indexing methods, and data compression techniques.

- **Logical Level**

The logical level of data abstraction is concerned with the overall structure and organization of data in the database. It defines the conceptual schema or the logical view of the database. At this level, the emphasis is on defining the tables, columns, relationships, constraints and other logical structures that make up the database. It provides a high-level representation of the data without concerning itself with the physical storage details.

- **View Level**

The view level of data abstraction focuses on the user's perspective or the specific requirements of applications. It involves creating customized views or subsets of data from the logical level to meet the specific needs of different users or applications. Views present a tailored and simplified representation of data, hiding unnecessary details and providing a more intuitive and user-friendly interface.

### ✓ Types of data independence

Data independence in databases refers to the ability to make changes to the database schema or organization without affecting the applications that use the data. There are two main types of data independence: logical data independence and physical data independence.

#### Logical Data Independence

**Definition:** Logical data independence allows you to make changes to the logical structure (schema) of the database without affecting the external schema or the application programs that interact with the data.

**Example:** Suppose you have a database with a certain table structure, and you decide to add a new attribute (column) to one of the tables. If the applications interacting with the database are not affected by this change and can still function without modification, then logical data independence is maintained.

#### Physical Data Independence

**Definition:** Physical data independence allows you to make changes to the physical storage structures or devices without affecting the logical schema or the application programs.

**Example:** Let's say you decide to change the storage structure of a table from one type of file organization to another (e.g., from sequential files to indexed files). If the applications accessing the data are not impacted by this change and can still retrieve and manipulate data as before, then physical data independence is maintained.

**Points to Remember**

- A database schema is a logical blueprint or design that defines the structure, organization, and relationships of a database.
- types of database schema are conceptual database, logical database and physical database schema.
- Three levels of database abstraction are Physical level, Logical level, and View level.
- Types of data independence are logical data independence and physical data independence.

**Application of learning 2.1.**

Bright Future University is developing a new **Student Information System (SIS)** to manage student records, academic performance, and enrolment. The university wish to hire a database designer who will helps it to take a decision about a database structure to be used by identifying elements of each structure, as technician, Suggest the database structure of this system.

## Indicative content 2.2: Design of conceptual Database Schema

**Duration: 6hrs**

### Theoretical Activity 2.2.1: Description of conceptual database schema

**Tasks:**

1: Answer the following questions:

I. Explain why conceptual database schema is required in database design?
II. What is an Entity Relationship Diagram (ERD)
III. Describe ERD by explaining its components

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 2.2.1.

### Key readings 2.2.1: Description of conceptual database schema

- **Conceptual database schema**

A conceptual database schema is a high-level representation of the structure and organization of a database. It provides a conceptual view of the database, focusing on the entities, their attributes, and the relationships between them.

The purpose of a conceptual schema is to provide a clear and abstract representation of the database design, independent of any specific database management system or implementation details. It serves as a blueprint for designing the logical and physical schemas that will be used to implement the database.

A conceptual database schema helps in understanding the overall structure and relationships within a database, facilitating effective database design and development. Typically, a conceptual schema is represented using entity-relationship (ER) diagrams.

✓ **Entity relationship diagram (ERD)**

An Entity-Relationship Diagram (ERD) is a visual representation of the data model that depicts the entities (objects or concepts), attributes (properties of entities), relationships (associations between entities), and cardinality (how many instances of one entity are related to another) within a system. ERDs are widely used in database design and software engineering to understand and document the structure and behaviour of a system's data.

- ER Model in DBMS stands for an Entity-Relationship model

- The ER model is a high-level data model diagram
- ER diagrams are a visual tool which is helpful to represent the ER model
- ER diagrams in DBMS are blueprint of a database
- Entity relationship diagram DBMS displays the relationships of entity set stored in a database
- ER diagrams help you to define terms related to entity relationship modeling
- ER Model in DBMS is based on three basic concepts: Entities, Attributes & Relationships
- An entity can be place, person, object, event or a concept, which stores data in the database (DBMS)
- Relationship is nothing but an association among two or more entities
- A weak entity is a type of entity which doesn't have its key attribute
- It is a single-valued property of either an entity-type or a relationship-type
- It helps you to defines the numerical attributes of the relationship between two entities or entity sets
- ER- Diagram DBMS is a visual representation of data that describe how data is related to each other
- While Drawing ER diagrams in DBMS, you need to make sure all your entities and relationships are properly labelled.

✓ **Components of ERD**

- **Entity**

An entity type in DBMS is a classification used to define and generate a set of entities that have common characteristics. For example, a company database may include entity types such as "customers," "products," and "sales." Each entity type in DBMS typically has its own set of attributes, or properties, that describe the Entity.

- **Types of entities**
  - **Strong Entity Type**

A strong entity in DBMS is an independent table that doesn't rely on any other tables for its existence.
A simple example of a strong entity type would be "customer" in a customer relational database table.
The entity relationship diagram represents a strong entity type with the help of a single rectangle.

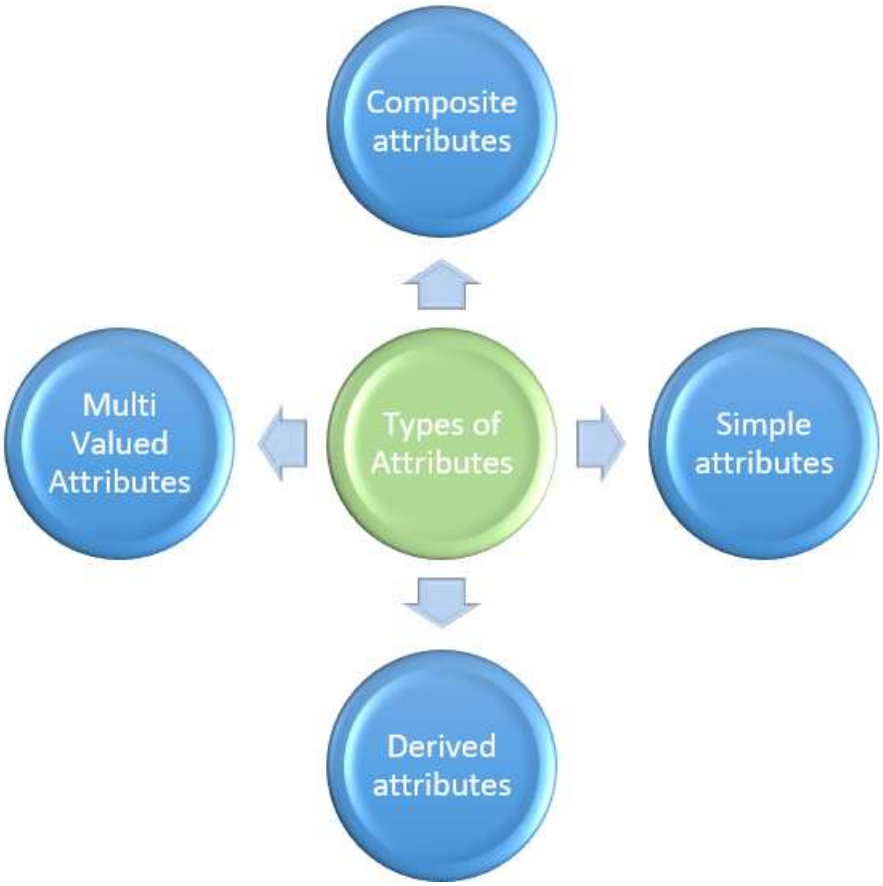

- **Weak Entity Type**

A weak entity type is a dependent table that relies on another table for its existence; it has no meaningful attributes of its own except for the foreign key from the parent table. A typical example of a weak entity type is an "order." It has no meaning by itself
–it must be placed by some customer so it has a foreign key relation to the "customer" table
–but it has several attributes of its own such as orderID, productID, etc.
–The Entity Relationship Diagram represents the weak entity type using double rectangles.


- **Attributes**

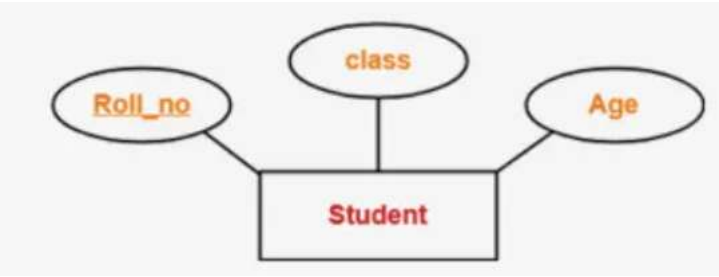
An attribute is a piece of data that describes an entity. For example, in a customer database, the attributes might be name, address, and phone number.

- **Types of Attributes in DBMS**



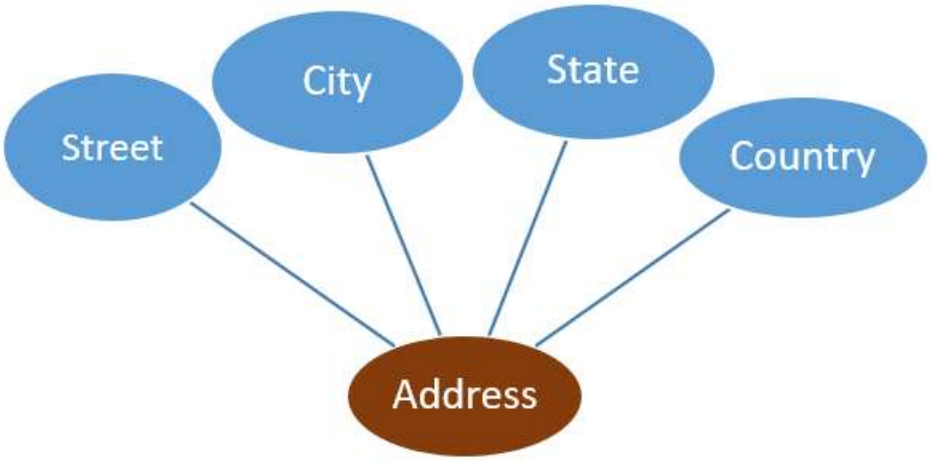

- **Simple Attributes/ Single Valued Attributes**

Simple attributes are those that cannot be further divided into sub-attributes.
For example, A student's roll number of a student or the employee identification number.




- **Composite Attributes**

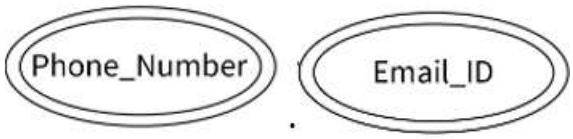
Composite attributes are made up of two or more simple attributes. For example, a person's address may be a composite attribute that is made up of the person's street address, city, state, and zip code.




Notes: from the simple and composite attributes there can derive other type of attributes known as **Key Attributes. Key Attributes:** are used to uniquely identify each row in a table. Usually, there is more than one key attribute in a table (primary key and foreign key).

- **Multivalued Attributes**

Multivalued attributes can have more than one value. For example, a person may have multiple email addresses or phone numbers. Multivalued attributes in DBMS are often used to store information about relationships between entities.
In ERD a multivalued attribute is represented by double ovals




- **Derived Attributes**

Derived attributes are based on other attributes and are not stored directly in the database.
For example: Consider a database of employees. Each employee has a date of birth, and we might want to calculate their age. However, age is a derived attribute because it can be determined from the date of birth.
In ERD derived attribute is represented in dash-dash oval or dot-dot oval


- **Relationship**

In the context of databases and Entity-Relationship Models (ERMs), relationships refer to the associations between different entities. These relationships define how data is stored and retrieved in a structured manner within a database.
In ERD a relationship is represented in Diamond


- **One to One Relationship**

A One-to-one relationship means a single record in Table A is related to a single record in Table B and vice-versa.
For example, a person has only one passport and a passport is given to one person.


- **One to Many Relationship**

Such a relationship exists when each record of table A can be related to one or more records of another table, i.e., table B. However, a single record in table B will have a link to a single record in table A. This is the most common relationship you will find that is widely used. A one-to-many relationship in DBMS can also be named a many-to-one relationship, depending on how we view it.

**For example:** a customer can place many orders but an order cannot be placed by many customers.


- **Many to Many Relationship**

A many-to-many relationship exists between the tables if a single record of the first table is related to one or more records of the second table and a single record in the second table is related to one or more records of the first table.

Consider the tables A and B. In a many-to-many relationship, each record in table A can be linked to one or more records in table B and vice-versa. It is also represented as an N: N relationship.

**For example,** a can be assigned to many projects and a project can be assigned to many students.


**Practical Activity 2.2.2: Designing ERD**


**Task:**

1: Read key reading 2.2.2.

2: Read the task described below:

> The management team of "Book Nook," a small independent bookstore, is seeking to develop a simple database system to manage its core operations. They require a database that can efficiently handle the essential aspects of their business with

a focus on simplicity. The system should manage customer information, including basic contact details and a record of their past purchases. It should also track the inventory of books, including details such as the title, author, genre, and stock levels. As a database designer you are requested to create an ERD that includes up to four tables, representing customers, books, sales transactions, and transaction details.

3: By referring to the key readings, perform the task 2 provided above

4: Present your work to the trainer and whole class

**Key readings 2.2.2: Designing ERD**

- **Step-by-step guide to Design ERD**

Now, let's dive into the step-by-step process of creating an ER diagram, which will enable you to create a well-structured and visually appealing representation of the data relationships.

✓ **Defining entities**

Start by identifying the main entities in your system. Entities are objects or concepts that have data to be stored. Define entities by adding entity names in rectangles

✓ **Establishing relationships**

Establishing relationships between entities is a crucial aspect of an ER diagram. Relationships define how entities are connected or associated with each other.
To identify the relationships between data entities, perform the following steps:

- Using the list of business functions, identify relationships between entities as verbs. In those instances where no verb adequately expresses the relationship, join the two entity names to form a name for the relationship. For example, the DEPARTMENT and EMPLOYEE entities could be connected through the relationship BELONGS TO or through the relationship DEPT-EMPLOYEE.
- List these key verbs between the entities they connect and draw a diamond around each one.
- Associate entities with the appropriate relationships by connecting them with lines.
- Label each relationship to show whether it is 1-1, 1-M, or M-M.

✓ **Adding attributes**

Attributes provide additional information about entities. Add relevant attributes to the entities identified in the previous steps. Based on the value an attribute will store add

attributes names in Ovals and link them to the entities

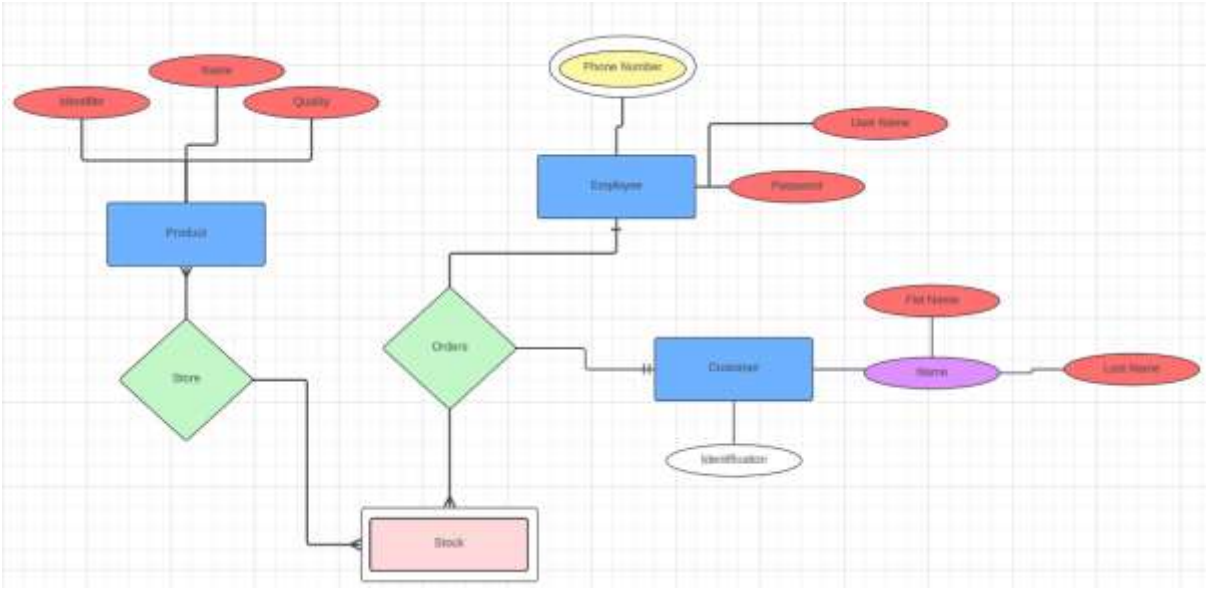
- ✓ **Defining the Relationship type by using cardinality symbols**
- **Refining the Diagram**

In this final step, we focus on refining the ER diagram to enhance clarity and readability. Organize the entities and relationships in a logical and intuitive manner. Group related entities together and arrange them in a way that reflects their connections.

**Practical Activity 2.2.3: Drawing an ERD using E-Draw max**


**Task:**

1: You are requested to go in the computer lab to draw ERD by using E-draw max referring to the ERD provided




2: Draw ERD Using E-draw max

3: Present your work to the trainer and whole class

Key readings 2.2.3: **Drawing an ERD using E-Draw max**

Edraw Max Online is a multi-purpose graphics tool that can be used to create a wide range of diagrams, charts, and other visual content. Create a basic ER diagram in Edraw by the following steps:

- **Steps to draw an ERD using E-Draw max**

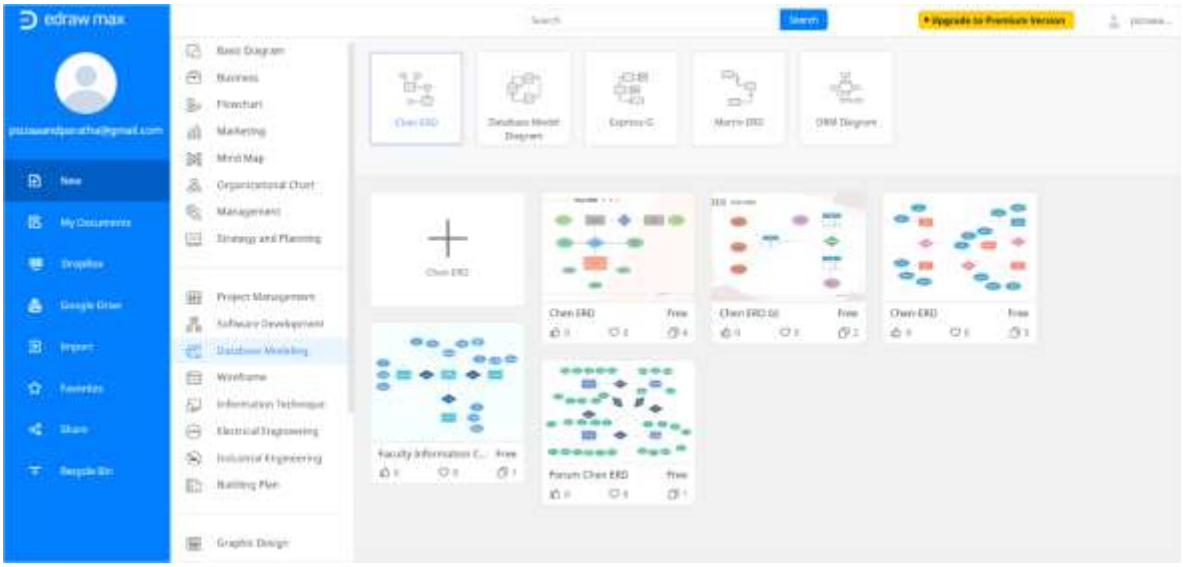
✓ **Open Edraw Max Online**

Launch Edraw Max online on the browser through this link: https://www.edrawmax.com/online/ to open the Edraw online diagramming tool.

✓ **Choose the ER Diagram**

In the navigation pane on the left side of the screen, click on Database Modeling and then click on the Chen ERD option or other options of ER Diagrams. You will get some predesigned templates and choose your favorite. Click on the one you prefer to create an ER diagram using a predesigned template.

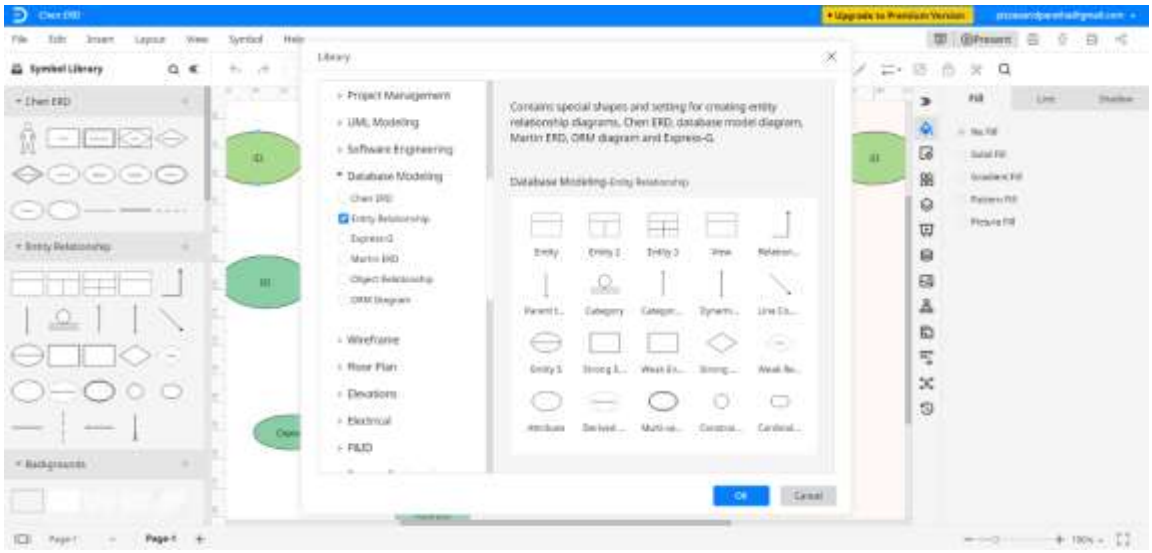
You can also create an ER Diagram from the start all by yourself. For this, you need to select the blank template.



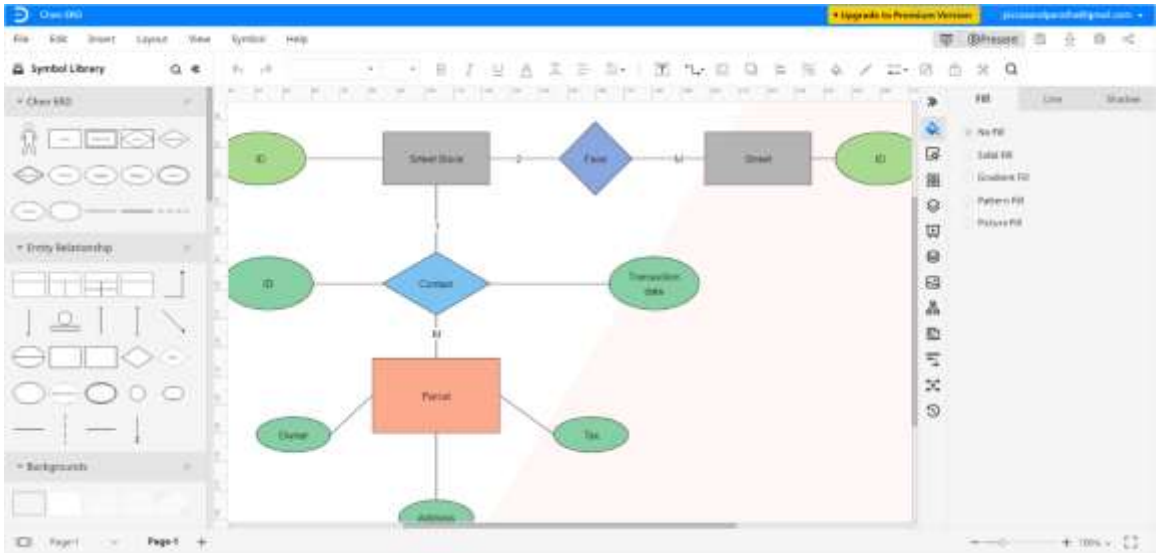

✓ **Create an ER Diagram**

Now Edraw Online will launch a new tab on your browser with the pre-made or blank template once you click on it. To create an ER diagram by yourself, add symbols in the Symbol Library. Click on the icon next to Symbol Library and wait for the pop-up window. Now scroll down to Database Modeling and click OK. ER diagram symbols will appear on

the left side under the Symbols Library tab. Now use the shapes, stickers, and symbols to create an ER diagram from scratch.



Use the navigation panes on either side of the screen to customize and edit your ER diagrams if you are using a premade free template.

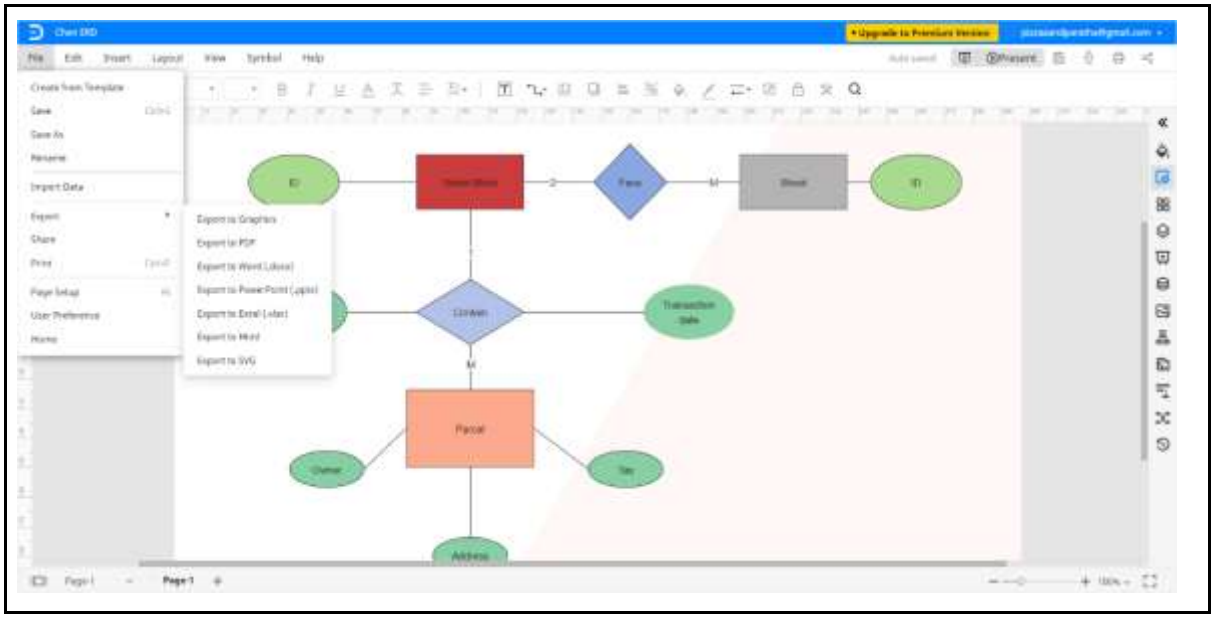


- ✓ **Customize Your ER Diagram**

Customize and edit your ER Diagram with the help of different formatting tools available in Edraw. Adjust the size, font, shape, color, alignment, and other details according to your preference.

- ✓ **Save File**

You can save your Edraw file for further editing or sharing. Click on File to save it. You can also export Edraw documents in the standard file formats and edit them in the corresponding software. For this, you need to click on File > Export and select the desired file type.



**Points to Remember**

- To describe the ERD, you should focus on Entities, Attributes, Relationships, and Cardinality symbols.
- Arrange entities and connect them using relationship before assigning the attributes to make a diagram clear
- End by add attributes and defining the type of relationship between entities to avoid confusion to the database developers
- Use an appropriate symbol for each element of ERD to avoid confusion in the diagram.
- Use E-draw Max either online or by downloading and installing it on your computer to draw ERD.
- Select the appropriate ERD format to optimize drawing time.
- Begin by arranging entities and connecting them with relationships to maintain clarity in the diagram, then add attributes afterward.

**Application of learning 2.2.**

The owner of "Fresh food Market," a local grocery store, wants to implement a new database system to streamline the store's inventory and sales operations. The system should manage product information, including product name, category, price, and stock quantity. It should also handle supplier details, such as supplier name, contact information, and the products they supply. The store needs to keep track of customer

purchases, including the date of purchase, items bought, and the total amount spent. Additionally, the database should support inventory management, enabling the store to monitor stock levels and automatically reorder products when they fall below a certain threshold. As a database designer design ERD and draw it by using Edraw max

## Indicative content 2.3: Design of Logical Database Schema

**Duration: 6hrs**

**Theoretical Activity 2.3.1: Description of logical database schema**

**Tasks:**

1: You are requested to answer the following questions related to logical database schema

I. What do you understand with logical database schema?

II. List and explain constraints in SQL

2: Present your findings to your classmates and trainer

3: Ask questions where necessary

4: For more clarification, read the key readings 2.3.1.

**Key readings 2.3.1: Description of logical database schema**

- **logical database schema**
  - ✓ **Definition**
    - A logical database schema defines all the logical constraints that need to be applied to the stored data, and also describes tables, views, entity relationships, and integrity constraints.
    - The Logical schema describes how the data is stored in the form of tables & how the attributes of a table are connected.
    - Using **ER modelling** the relationship between the components of the data is maintained.
    - In logical schema different integrity constraints are defined in order to maintain the quality of insertion and update the data.
  - ✓ **Table constraints**

Table constraint is a rule or condition that is applied to the data in a table.

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. If there is any violation between the constraint and the data action, the action is aborted.

- **NOT NULL Constraint**

A NOT NULL constraint is a type of table constraint in a relational database that ensures a specific column does not contain any NULL values. When a NOT NULL constraint is

applied to a column, it means that every row in the table must have a valid, non-null value in that column.

- **UNIQUE Constraint**

A unique constraint ensures that the values in a specific column or a set of columns are unique across all the rows in the table. Unlike a primary key, unique constraints allow null values.

Unique Ensures the uniqueness of values in a specified column or columns.

- **DEFAULT Constraint**

A DEFAULT constraint is a feature in relational databases that allows you to specify a default value for a column. When a new row is inserted into the table and a value for that column is not provided, the default value specified in the DEFAULT constraint will be used. If a value is provided during the insertion, the provided value takes precedence over the default value.

- **CHECK Constraint**

A check constraint specifies a condition that must be true for every row in the table. If a row does not satisfy the condition, it is not allowed to be inserted or updated.

Check Enforces business rules or specific conditions on the data stored in the table.

- **PRIMARY KEY Constraint**

A primary key constraint ensures that a specific column or a set of columns in a table contains unique and non-null values. This means that every row in the table is uniquely identified by the values in the primary key column(s).

Primary key Enforces data integrity by preventing duplicate or null values in the primary key column(s).

- **FOREIGN KEY Constraint**

A foreign key constraint establishes a link between two tables by specifying that the values in a specific column (or columns) in one table must correspond to the values in a primary key column in another table.

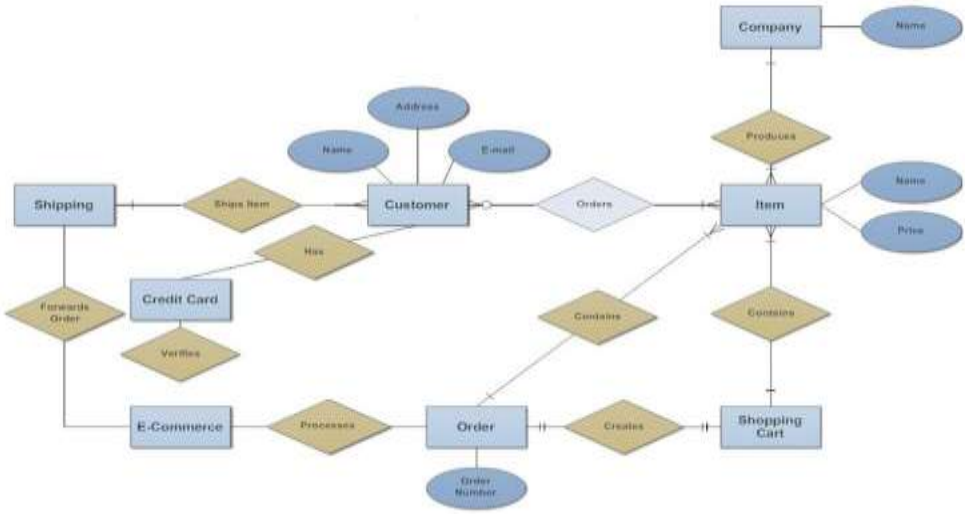
Foreign key Enforces referential integrity by ensuring that relationships between tables are maintained.

**Practical Activity 2.3.2: Converting conceptual database schema to logical database schema**

**Task:**

1: You are requested to perform the task described below:

As a database designer you are requested to go in computer lab to transform the below ERD into logical database schema:




2: Design logical database schema

3: Present your work to the trainer and whole class

4: Ask clarification where necessary

**Key readings 2.3.2: Converting conceptual database schema to logical database schema**

- **Steps to convert a conceptual database schema to a logical database schema in MySQL**
  - ✓ **Review the Conceptual Schema**

Understand the entities, attributes, and relationships defined in the conceptual schema.

- ✓ **Identify Entities and Attributes**

Identify each entity and its corresponding attributes. Determine the data types, lengths, and constraints for each attribute.

- ✓ **Design Tables**

Create a table for each entity in the conceptual schema. Map the attributes of each entity to columns in the table. Consider the appropriate data types for each column based on the attribute definitions.

- ✓ **Define Primary Keys**

Identify the primary key for each table. The primary key uniquely identifies each record in the table. Specify the primary key constraint for the corresponding column(s) in each table.

- ✓ **Establish Relationships**

Determine the relationships between entities and translate them into foreign keys. Identify the primary key of the parent table and create a foreign key column in the child table that references it. Use the FOREIGN KEY constraint to establish the relationship between the tables.

- ✓ **Define Constraints**

Specify any additional constraints or rules that need to be enforced on the data. This includes defining unique constraints, check constraints, and any other business rules that need to be upheld. Use the appropriate constraints in MySQL to enforce these rules.

- ✓ **Review and Refine**

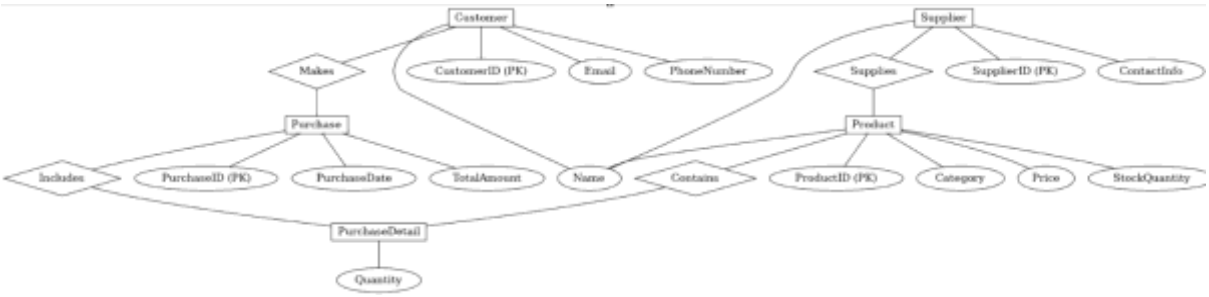
Review the logical schema for completeness, accuracy, and efficiency. Ensure that it meets the requirements and performance goals of the MySQL database. Make any necessary refinements or adjustments based on your analysis.

**Points to Remember**

- You should describe table constraint through NOT NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY and FOREIGN KEY constraints
- "In a logical database design, if a relationship in the ERD involves a many-to-many connection between two entities, it should typically be represented as a separate table.
- Select a primary key for each entity
- Create a relationship between entities by migrating a primary key of a parent entity in the child entity
- Arrange entities properly to maintain clarity of a schema.
- Incorporate Entities, relationships between entities, Attributes and constraints in the logical database schema.

**Application of learning 2.3.**

The owner of "Fresh Harvest Market," provide the below ERD and he is seeking for a database developer to develop to transform it into Logical database schema.




You as a database designer you are requested to transform it in Logical database schema.

## Indicative content 2.4: Optimization of Database

**Duration: 6hrs**

**Theoretical Activity 2.4.1: Description of database optimisation**

**Tasks:**

1: You are requested to answer the following question:

I. What do you understand by database optimisation?
II. Explain the ways that can be used to optimise database

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 2.4.1.

**Key readings 2.4.1: Description of database optimisation**

- **Database optimization**

Database optimization refers to a variety of strategies for reducing database system response time.

Database performance depends on several factors at the database level, such as tables, queries, and configuration settings. Here let's focus on Optimizing at the Database Level The most important factor in making a database application fast is its basic design.

Are the tables structured properly?

Are the right indexes in place to make queries efficient?

Are you using the appropriate storage engine for each table, and taking advantage of the strengths and features of each storage engine, you use?

Now let's focus on table structure by checking if the structure of a table is proper and index

✓ **Method of optimising database**

- **Data Normalization**
  - **Definition**

Data Normalization is a process in database design that aims to eliminate data redundancy and improve data integrity by organizing data into well-structured and normalized tables. It involves breaking down a large table into smaller, more manageable tables and establishing relationships between them.

- Normalization is the process of organizing the data in the database.

- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Here are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As with many formal rules and specifications, real world scenarios don't always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

- **Types of Normal Forms**

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

| **Normal Form** | **Description** |
|---|---|
| 1NF | A relation is in 1NF if it contains an atomic value. |
| 2NF | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| 3NF | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |

- **Advantages of Normalization**
  - Normalization helps to minimize data redundancy.
  - Greater overall database organization.
  - Data consistency within the database.
  - Much more flexible database design.
  - Enforces the concept of relational integrity.

### Indexing

indexing refers to the process of selecting and defining indexes on the appropriate columns of database tables to improve query performance. It involves identifying the columns that are frequently used in search conditions or join operations and creating indexes on those columns.

The goal of indexing in logical design is to improve the efficiency of data retrieval operations by reducing the number of disk I/O operations and minimizing the amount of data that needs to be scanned. By using indexes, the database can quickly locate the desired data rows based on the indexed columns, rather than performing a full table scan.

**How Does Database Indexing Works**

Imagine we have a table of users which includes their name and salary:

| id | name | salary |
|---|---|---|
| 1 | Alice | 50000 |
| 2 | Bob | 60000 |
| 3 | Sam | 55000 |
| 4 | Eve | 70000 |
| 5 | Carol | 45000 |
| 6 | Dave | 58000 |

**Querying without an index**

Suppose we want to execute the query SELECT * FROM users WHERE name='Sam' and retrieve all records for the user Sam. In normal cases where we do not have indexing, the database system would need to scan through each row in the table sequentially to find all occurrences of "Sam." This process is called a full table scan. For our sample table with 6 records, this may not seem like a big deal, but imagine what would happen if we had millions of records. The time it takes to scan through each row would increase significantly, resulting in slower query performance.

Plus, just like to SELECT statements, UPDATE and DELETE queries also benefit from index optimization techniques.

**Querying with an index: B-Tree DBMS**

An index is an additional data structure that is added to the database to speed up queries. When you add an index to a database column, the indexed structure's data is typically stored separately from the actual data in the database. In MySQL indexes are stored in a separate file on disk as a B-tree data structure, but they are managed and accessed by the DBMS (Database Management System). The index file is kept in sync

with the data in the table. When you insert, update, or delete rows in the table, the DBMS automatically updates the index to reflect the changes.

Let's visualize how the column "name" looks in each step of the B-tree DBMS process:

**Initial index:**

| id | name | salary |
|---|---|---|
| 1 | Alice | 50000 |
| 2 | Bob | 60000 |
| 5 | Carol | 45000 |
| 6 | Dave | 58000 |
| 4 | Eve | 70000 |
| 3 | Sam | 55000 |

B-tree (balanced tree) is a way of organizing data in a special tree structure., which makes searching for specific information in the database really quick and efficient, whether you're looking for exact matches or ranges of values.

**Step 1**

The database system looks at the sorted index's middle value, "Dave." Since "Dave" comes before "Sam" alphabetically, the system knows that all names after "Dave" in the index are also greater than "Sam." Thus, it eliminates the second half of the index.

| id | name | salary |
|---|---|---|
| 1 | Alice | 50000 |
| 2 | Bob | 60000 |
| 5 | Carol | 45000 |

**Step 2**

The system focuses on the first half of the index and repeats the process. It looks at the middle value, which is "Bob." Since "Bob" comes before "Sam," it eliminates the portion of the index before "Bob."

| id | name | salary |
|---|---|---|
| 5 | Carol | 45000 |

**Step 3**

The system focuses on the remaining portion of the index and repeats the process until it finds the occurrence of "Sam." This process is called a "B-tree" search and it significantly reduces the number of comparisons needed compared to scanning through all records sequentially. It results in faster query performance, especially when dealing with large datasets, as the database system can leverage the sorted index to narrow down the search space and retrieve the desired records more efficiently.

- **Types of indexes**

- **Primary Index**

Automatically created on a table's primary key. Ensures each record is unique and helps quickly locate rows based on the primary key.

- **Secondary index**

This is an additional index created on a table to improve data retrieval performance for columns that are not the primary key or clustered index. It allows the database to locate rows more efficiently based on these other columns without scanning the entire table.

- **Clustered Index**

Determines the physical order of data in the table based on the indexed column(s). Each table can have only one clustered index because data rows are stored in this order. Faster for retrieving ranges of data but may slow down insertions, updates, and deletions.

- **Non-Clustered Index**

Does not alter the physical order of the data but creates a separate structure to store the index with pointers to the data rows. Tables can have multiple non-clustered indexes, which are useful for speeding up read queries.

**Practical Activity 2.4.2: Normalising database**


**Task:**

1: You are requested to read the task described below:

A small retail company, "KIGALITechs," maintains a database to keep track of its inventory and sales. The current database structure stores information in a single table called Orders, which includes the following attributes: OrderID, CustomerName, CustomerAddress, Product ID, Product Name, Quantity, Unit Price, and Order Date. Over time, the company has experienced several issues, such as redundant data, difficulty in maintaining accurate customer information, and inconsistencies in product pricing. As a database designer you are tasked with normalizing this table to improve its efficiency and integrity. Check 1NF,2NF, and 3NF

2: Referring to the key reading 2.4.2. perform task 1 described above

3: Present your work to the trainer and whole class

**Key readings 2.4.2: Normalising database**

- **Steps to normalise database**

✓ First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

**Example:** Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

**EMPLOYEE table:**

| EMP_ID | EMP_NAME | EMP_PHONE | EMP_STATE |
|---|---|---|---|
| 14 | John | 7272826385, 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389, 8589830302 | Punjab |

The decomposition of the EMPLOYEE table into 1NF has been shown below:

| EMP_ID | EMP_NAME | EMP_PHONE | EMP_STATE |
|---|---|---|---|
| 14 | John | 7272826385 | UP |
| 14 | John | 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389 | Punjab |
| 12 | Sam | 8589830302 | Punjab |

✓ **Second Normal Form (2NF)**

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

**Example:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

**TEACHER table**

| TEACHER_ID | SUBJECT | TEACHER_AGE |
|---|---|---|
| 25 | Chemistry | 30 |
| 25 | Biology | 30 |
| 47 | English | 35 |
| 83 | Math | 38 |
| 83 | Computer | 38 |

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

**TEACHER_DETAIL table:**

| TEACHER_ID | TEACHER_AGE |
|---|---|
| 25 | 30 |
| 47 | 35 |
| 83 | 38 |

**TEACHER_SUBJECT table:**

| TEACHER_ID | SUBJECT |
|---|---|
| 25 | Chemistry |
| 25 | Biology |
| 47 | English |
| 83 | Math |
| 83 | Computer |

- ✓ **Third Normal Form (3NF)**

  - A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
  - 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
  - If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

**Example:**

**EMPLOYEE_DETAIL table:**

| EMP_ID | EMP_NAME | EMP_ZIP | EMP_STATE | EMP_CITY |
|---|---|---|---|---|
| 222 | Harry | 201010 | UP | Noida |
| 333 | Stephan | 02228 | US | Boston |
| 444 | Lan | 60007 | US | Chicago |
| 555 | Katharine | 06389 | UK | Norwich |
| 666 | John | 462007 | MP | Bhopal |

Super key in the table above:

1. {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

**EMPLOYEE table:**

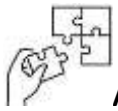
| EMP_ID | EMP_NAME | EMP_ZIP |
|---|---|---|
| 222 | Harry | 201010 |
| 333 | Stephan | 02228 |
| 444 | Lan | 60007 |
| 555 | Katharine | 06389 |
| 666 | John | 462007 |

**EMPLOYEE_ZIP table:**

| EMP_ZIP | EMP_STATE | EMP_CITY |
|---|---|---|
| 201010 | UP | Noida |
| 02228 | US | Boston |
| 60007 | US | Chicago |
| 06389 | UK | Norwich |
| 462007 | MP | Bhopal |

**Points to Remember**

- For better understanding database optimization, use Normalization and indexing descriptions
- Check if the column values are atomic. If they are, proceed to the next step; if not, ensure the column values are made atomic.
- Verify if each non-key column depends on a single prime attribute. If it does, move to the next step; if not, separate the columns based on their dependency on different prime attributes by splitting the table into additional tables.
- Ensure that non-key attributes are mutually independent. If they are not, the process is complete; if they are, separate the non-key columns that depend on one another.
- If you split a table into more tables, establish relationships between the newly created tables.



**Application of learning 2.4.**

Isange restaurant located at Kicukiro District, Mukingi Sector, Mutovu village, has a restaurant management application that is used to store data about the company's employees it is annoyed by the information in that table because of duplications of data. It needs to hire a database designer to help it to change the structure of employee table to remove data duplication. As a database designer normalize the below table:

| employee_id | name | job code | job | state code | home state |
|---|---|---|---|---|---|
| E001 | Alice | J01 | Chef | 26 | Michigan |
| E001 | Alice | J02 | Waiter | 26 | Michigan |
| E002 | Bob | J02 | Waiter | 56 | Wyoming |
| E002 | Bob | J03 | BXartender | 56 | Wyoming |
| E003 | Alice | J01 | Chef | 56 | Wyoming |

## Indicative content 2.5: Design of Physical Database Schema

**Duration: 5 hrs**

### Theoretical Activity 2.5.1: Description of DBMS

**Tasks:**

1: You are requested to answer the following questions:

I. What do you understand by DBMS?

II. What are the components of DBMS?

2: Present your answers to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 2.5.1.

### Key readings 2.5.1: Description of DBMS

- **DBMS**

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

✓ **Types of DBMS**

**SQL DBMS (Relational DBMS)**

- Data Structure:
  - Uses structured data stored in tables with predefined schemas.
  - Relationships between tables are established using foreign keys.
- Query Language:
  - Utilizes SQL (Structured Query Language) for data manipulation and retrieval.
  - Supports complex queries, joins, and transactions.
- ACID Compliance:
  - Generally, adheres to ACID properties (Atomicity, Consistency, Isolation, Durability) to ensure data integrity.
- Scalability:
  - Typically scales vertically (adding more power to a single server).
  - May face challenges with horizontal scaling (distributing data across multiple servers).

- Use Cases:
  - ♦ Ideal for applications requiring complex queries, transactions, and structured data (e.g., banking systems, ERP).
    Examples: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

**NoSQL DBMS**

- Data Structure:
  - ♦ Designed for unstructured or semi-structured data; can store various formats (documents, key-value pairs, graphs, etc.).
  - ♦ Schema-less or flexible schema design allows for rapid changes.
- Query Language:
  - ♦ Uses various query languages or APIs specific to the NoSQL database type (e.g., MongoDB uses BSON, Redis uses commands).
- Eventual Consistency:
  - ♦ Often follows an eventual consistency model rather than strict ACID compliance, allowing for higher availability and partition tolerance.
- Scalability:
  - ♦ Generally designed to scale horizontally, allowing for easy distribution of data across multiple servers.
- Use Cases:
  - ♦ Suitable for applications with large volumes of data, real-time analytics, and those requiring flexibility (e.g., social media, big data applications).
- Examples:
  - ♦ MongoDB (document-oriented), Cassandra (column-family), Redis (key-value), Neo4j (graph).

✓ **Components of DBMS**

- Hardware
- Software
- Data
- Procedures
- Database Access Language
- People

**This module Will Focus on MySQL DBMS**

✓ **MySQL**

- **MySQL is a database management system.**

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling

large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL databases are relational.**

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed.

- **MySQL software is Open Source.**

Open-Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything.

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.

**Practical Activity 2.5.2: Preparing DBMS Environment (MySQL)**


**Task:**

1: You are requested to read the task described below:

As a database designer you are requested to prepare MySQL environment

2: Referring to the key reading2.5.2, prepare MySQL environment

3: Present your work to the trainer and whole class

**Key readings 2.5.2: Preparation of DBMS Environment (MySQL)**

- **Preparation of DBMS Environment (MySQL)**
  - ✓ Steps to prepare MySQL environment

**Install MySQL**

Download and install the MySQL Server from the official MySQL website (https://dev.mysql.com/downloads/). Choose the appropriate version for your operating system.

**Configure MySQL**

During the installation process, you will be prompted to configure MySQL. Set the root password and specify other configuration options as needed.

**Start MySQL Server**

Once the installation is complete, start the MySQL Server. On Windows, you can start it from the Services panel or using the MySQL Command Line Client. On Unix/Linux, you can start it from the terminal using the command `sudo service mysql start` or `sudo systemctl start mysql`.

**Connect to MySQL**

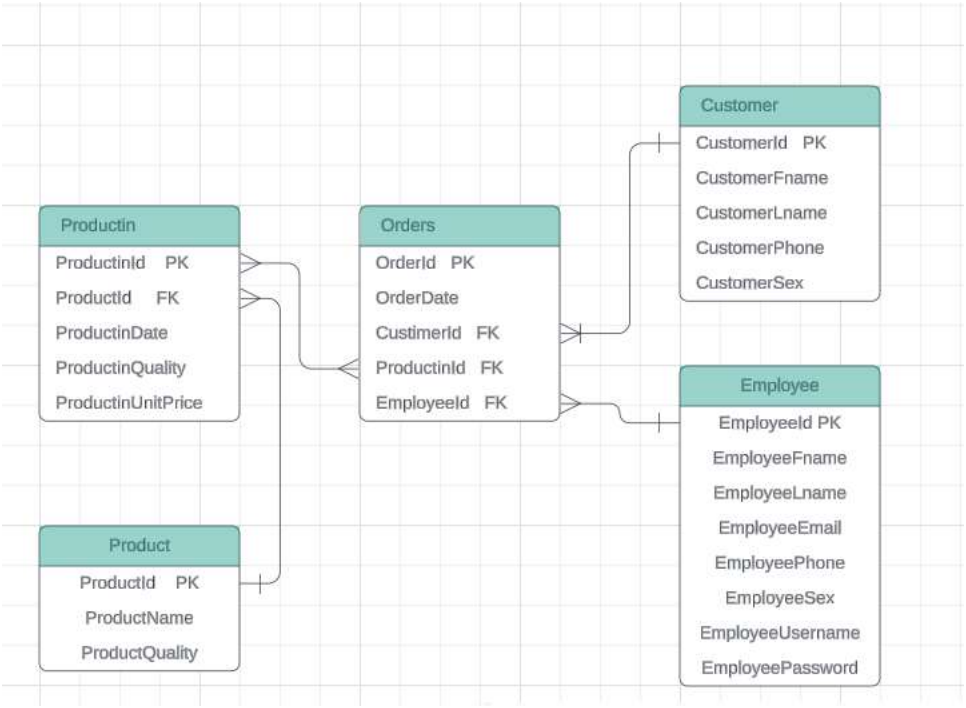
Use a MySQL client to connect to the MySQL Server. The MySQL Command Line Client is a popular option. Open the command prompt or terminal and enter the command `mysql -u root -p` to connect as the root user. Enter the root password when prompted.


**Practical Activity 2.5.3: Converting logic database schema to physical database schema**

**Task:**

1: You are requested to read the task described below:

As a database designer you are requested to go in computer lab to transform the below logical database schema into physical database schema:




2: Referring to the key reading2.5.3, design a physical database schema

3: Present your work to the trainer and whole class

**Key readings 2.5.3: Converting logic database schema to physical database schema**

- **Steps to convert logical database schema into physical database schema**

**Review Logical Schema**

- **Examine the Logical Data Model:** Ensure the logical schema includes all entities, relationships, attributes, and constraints as defined in your ERD or logical data model.

**Select a Database Management System (DBMS)**

- **Choose the DBMS:** Decide on the DBMS that will be used for implementation. Here our DBMS is MySQL.
- **Consider DBMS-Specific Features:** Take into account the specific features and limitations of the chosen DBMS.

**Define Tables**

- **Map Entities to Tables:** Convert each entity in the logical schema into a table in the physical schema.
- **Determine Primary Keys:** Assign primary keys to each table based on the entity's unique identifier.

**Specify Columns and Data Types**

- **Map Attributes to Columns: Convert each attribute of an entity into a column in the corresponding table.**
- **Choose Data Types: Define appropriate data types for each column based on the data that will be stored**

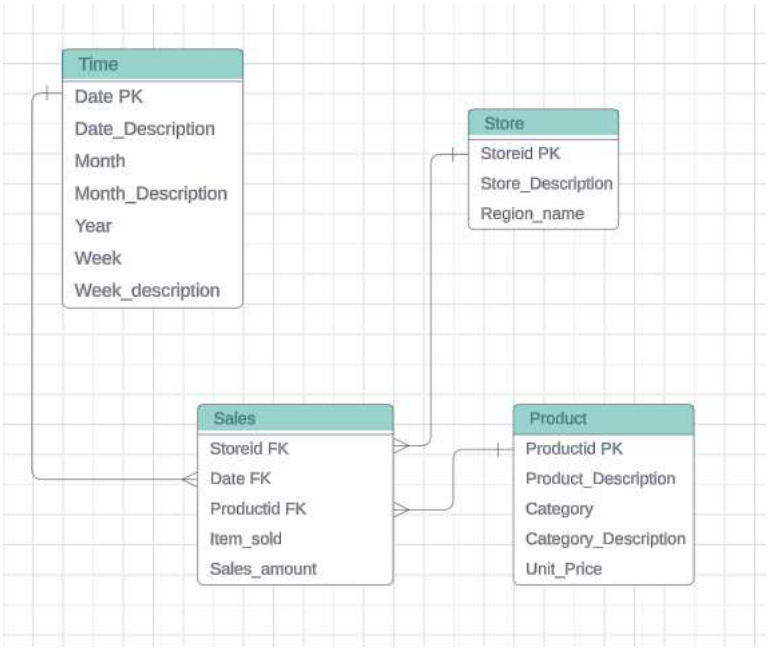
- **Define Constraints**
- **Define relationship**: 1:1, 1:N, N:N
- **Normalize Data (if needed):**
  - Apply Normalization Rules: Ensure the database is normalized to the desired level to minimize redundancy and improve data integrity.

**Points to Remember**

- A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner.
- Components of DBMS: Hardware, Software, Data, Procedures, Database Access Language, People.
- During XAMPP installation, select all necessary components required for your DBMS to ensure optimal functionality
- Understand that the physical database schema does not support many-to-many relationships.
- Use the physical database schema to guide how the database will be created and stored.
- Always organize the table structure to ensure a clear and comprehensible database structure.
- Link the primary key field to its foreign key to provide guidance for database developers.

**Application of learning 2.5.**

RAHABU company LTD is a company that sales its product online, it is located in NYABUHU DISTRICT. It is in the process of designing a database that will helps their employees to manage online sales the previous database designer lost his job after designing logical database schema shown below




you are hired as a database designer to resume the database design by convert the above logical database schema into physical database schema.

## Learning outcome 2 end assessment

**Theoretical assessment**

**Question1: Match the following terms in column A with their correct definitions in column B**

| Answers | Terms (Column A) | Definitions (Column B) |
|---|---|---|
| 1.. | 1. Database Schema | A. A big-picture view of what the system will contain, how it will be organized, and which business rules are involved. |
| 2.. | 2. Conceptual Schema | B. Clearly defines schema objects with table names, field names, relationships, and integrity constraints but lacks technical requirements. |
| 3.. | 3. Logical Schema | C. is the lowest level of data abstraction, which deals with the physical storage and representation of data. |
| 4.. | 4. Physical Schema | D. Provides a high-level representation of the data, defining tables, columns, and relationships without concerning physical storage details. |
| 5... | 5. Physical Level of Data Abstraction | E. Is a process of simplifying the structure of database by removing data duplication |
| | | F. Defines the structure, organization, and relationships of a database, including tables, columns, data types, and constraints. |

**Question2:** Read careful the below statements and encircle the correct answer from the provided options

1) **The following features are included in a database schema Except:**

   A) Tables and columns

   B) Primary key constraints

   C) User interface designs

   D) Data types and lengths

2) **A conceptual schema provides:**

   A) The syntax for creating data structures on disk storage

   B) A high-level view of the system's organization and business rules

   C) Detailed technical requirements for database implementation

   D) A specific view for different database users

3) **The following are table constraints Except…….**

   A) Primary Key

   B) Foreign Key

   C) Data Type

   D) Unique

4) **PRIMARY KEY constraint ensures that…….**

   A) All values in the column(s) are unique.

   B) The column(s) cannot contain NULL values.

   C) The column(s) uniquely identify each row in the table.

   D) All of the above.

5) **A constraint ensures that a column must have a value (i.e., it cannot be NULL) is …….**

   A) PRIMARY KEY

   B) FOREIGN KEY

   C) CHECK

   D) NOT NULL

6) **A constraint that is used to enforce a relationship between two entities is known as a……..**

   A) PRIMARY KEY

   B) FOREIGN KEY

C) UNIQUE

D) CHECK

**7) The role of a UNIQUE constraint is to…..**

A) Ensures that all values in the column(s) are different.

B) Ensures that no two rows have the same values in the specified column(s).

C) Allows only one NULL value in the column.

D) All of the above.

**8) The constraint used to limit the range of values that a column can store is known as…..**

A) PRIMARY KEY

B) CHECK

C) UNIQUE

D) FOREIGN KEY

**Question3:** write T to the correct statement and F to the wrong statement

a) A strong entity is an entity that does not depend on any other entity for its existence.
b) A weak entity can exist independently without being associated with a strong entity.
c) In a one-to-one relationship, each entity in one entity set can be associated with multiple entities in the other entity set.
d) A one-to-many relationship means that one entity in an entity set is associated with many entities in another entity set.
e) In a many-to-many relationship, each entity in one entity set can be associated with multiple entities in another entity set, and vice versa.
f) Attributes are properties or characteristics of entities that help to describe the entity.
g) A composite attribute is an attribute that can be divided into smaller sub-attributes.

**Question 4: Match Normal forms to their descriptions**

| **Answers** | **Forms of Normalisation** | **Description** |
|---|---|---|
| 1.. | 1.1NF | A. Ensures that a table has no repeating groups and that each column contains atomic (indivisible) values. |
| 2.. | 2.2NF | B. Requires that a table be in 2NF and that all transitive dependencies are eliminated, ensuring that non-key attributes are only dependent on the primary key. |
| 3.. | 3.3NF | C. Requires that a table be in 1NF and that all non-key attributes be fully functionally dependent on the primary key. |

**Practical assessment**

KTG is a secondary school located in your village, it needs a database designer to redesign a database for its system that cover all key aspects of the school's operations, including student information, teacher details, class schedules, subjects, and grades. The current database it uses it turn errors in the weekly report because of the gaps in database design. You as a database design you are hired by that school and your tasks are:

- To create an Entity-Relationship Diagram (ERD) and draw it using E-draw Max.
- Design a logical database schema
    - Design physical database schema
    - Prepare MySQL environment

**END**

## References


*https://miro.com/diagramming/how-to-draw-an-er-diagram/ diagram.*

*https://www.edrawsoft.com/entity-relationship-diagrams.html*

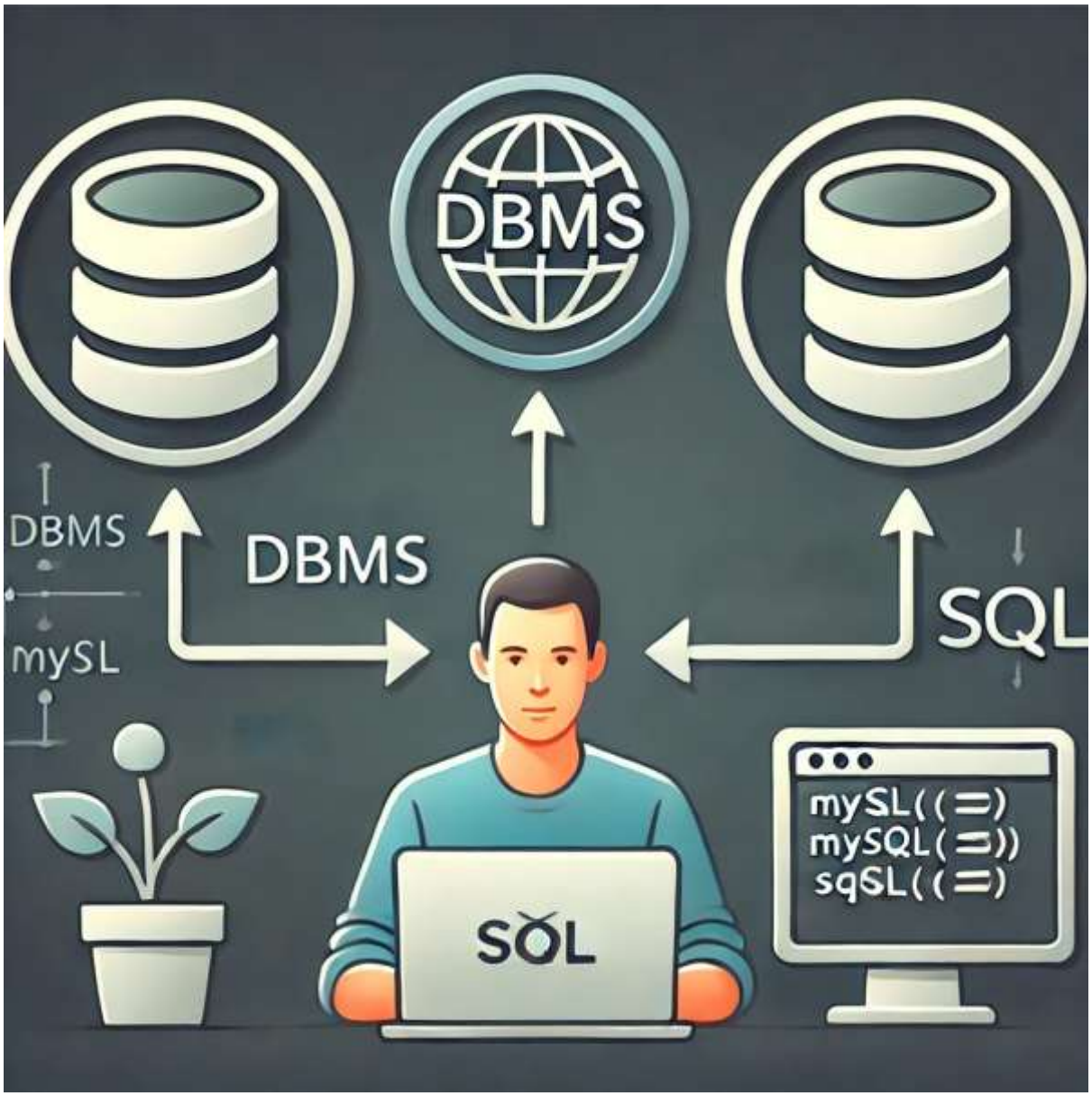
*https://www.ibm.com/topics/database-schema*

*Ramez Elmasri, S. B. (2015). Fundamentals of Database Systems. Boston: Pearson.*

*Sciore, E. (2008). Database Design and Implementation. Hoboken, New Jersey, USA: Wiley.*

*Watt, A. (2014). Database Design – 2nd Edition. Victoria, British Columbia, Canada: BCcampus Open Education.*

*www.geeksforgeeks.org/database-schemas/*


DBMS
DBMS
DBMS
mySL
SQL
SOL
mySL( (=)
mySQL( =))
sqSL( (=)

| Indicative contents |
| --- |
| **3.1 Description to SQL**<br>**3.2 Application of DDL commands**<br>**3.3 Application of DML commands**<br>**3.4 Application of DQL Command**<br>**3.5 Application of DCL commands**<br>**3.6 Application of TCL commands** |

## Key Competencies for Learning Outcome 3: Implement Database

| Knowledge | Skills | Attitudes |
| --- | --- | --- |
| • Description of SQL<br>• Description of DDL Command<br>• Description of DML Commands<br>• Description of DQL Commands<br>• Description of DCL Commands<br>• Description of TCL Commands | • Writing SQL queries<br>• Applying DDL Commands<br>• Applying DML Commands<br>• Applying DQL Commands<br>• Applying DCL Commands<br>• Applying TCL Commands | • Being Flexible in command Query debugging<br>• Having Teamwork error handling<br>• Having self Confidence while executing SQL command.<br>• Being rapid during work execution |

**Duration: 20 hrs**

**Learning outcome 3 objectives**:

By the end of the learning outcome, the trainees will be able to:

1. Describe properly SQL as used in database
2. Describe properly SQL Queries in Database
3. Apply effectively DDL Queries based on database schema
4. Apply effectively DML Queries based on database schema
5. Apply effectively DQL Queries based on database schema
6. Apply effectively DCL Queries based on database schema
7. Apply effectively TCL Queries based on database schema

**Resources**

| Equipment | Tools | Materials |
|---|---|---|
| • Computer | • MySQL<br>• Apache<br>• XAMPP<br>• WAMP<br>• LAMP | • Pen<br>• Papers |

**Advance Preparation:**

Before delivering this learning outcome, you are recommended to:

- Avail computer lab with installed XAMPP and web browser.

## Indicative content 3.1: Description to SQL

**Duration: 5 hrs**

**Theoretical Activity 3.1.1: Description of SQL**

**Tasks:**

1: You are requested to answer the following questions related to the description of SQL

1. What do you understand by SQL?
2. Differentiate SQL sub-Languages
3. What are SQL operators

2: Present your answers to your classmate and your trainer

3: Ask questions where necessary.

4: For more clarification, read the key readings 3.1.1.

**Key readings 3.1.1.: Description of SQL**

- **Introduction of SQL**
  - ✓ **Structured Query Language (SQL)**

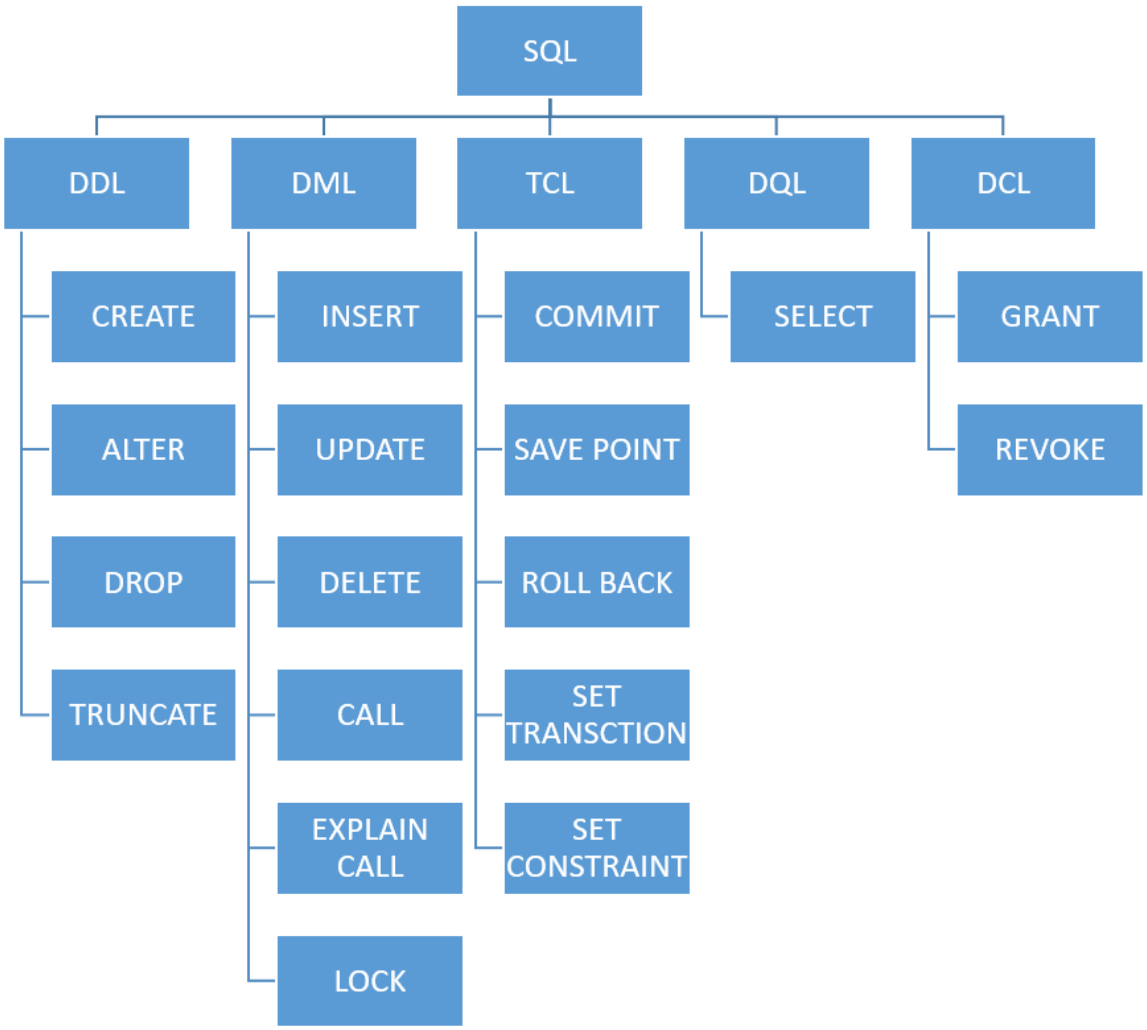
As we all know, is the database language by which we can perform certain operations on the existing database, and we can also use this language to create a database. SQL uses certain commands like CREATE, DROP, INSERT, etc. to carry out the required tasks.

SQL commands are like instructions to a table. It is used to interact with the database with some operations. It is also used to perform specific tasks, functions, and queries of data. SQL can perform various tasks like creating a table, adding data to tables, dropping the table, modifying the table, set permission for users.

These SQL commands are mainly categorized into five categories:

- **DDL** – Data Definition Language
- **DQL –** Data Query Language
- **DML –** Data Manipulation Language
- **DCL –** Data Control Language
- **TCL –** Transaction Control Language

Now, we will see all of these in detail.




✓ **SQL sub-languages**

- **Data Definition Language (DDL):** SQL includes commands for defining and modifying the structure of a database schema. This includes creating tables, modifying table structures, and defining constraints such as primary keys, foreign keys, and indexes.
  - **CREATE**: Is used to create database and its objects (table, index, function, views, store procedure, and triggers)
  - **DROP:** is used to Delete objects from the database or to delete database from DBMS
  - **ALTER:** Is used to modify the structure of the database objects
  - **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 existing in the database

- **Data Manipulation Language (DML):** DML commands are used to retrieve, insert, update, and delete data in the database. Common DML commands include

- **INSERT:** is used to add new records in database object
- **UPDATE:** is used to modify existing records of an object
- **DELETE:** is used to remove records for database table.
- **LOCK** Table control concurrency
- **Call a PL/SQL** or JAVA subprogram
- **EXPLAIN PLAN:** Describe the access path to data

- **Data Control Language (DCL):** DCL commands are used to control access to data within the database. This includes commands such as GRANT (to give users permission to perform certain operations) and REVOKE (to revoke previously granted permissions).

  - **BEGIN TRANSACTION**: Starts a new transaction
  - **COMMIT:** Saves all changes made during the transaction
  - **ROLLBACK** Undoes all changes made during the transaction
  - **SAVEPOINT:** Creates a save point within the current transaction
- **Transaction Control Language:** SQL supports transactions, which are sequences of one or more SQL operations treated as a single unit of work. Transactions ensure data integrity by allowing multiple operations to either succeed or fail together. Common transaction commands include COMMIT (to save changes) and ROLLBACK (to undo changes).
  - **GRANT:** Assigns new privileges to a user account, allowing access to specific database objects, actions, or functions.
  - **REVOKE:** Removes previously granted privileges from a user account, taking away their access to certain database objects or actions.

- **Data Query Language:** One of the primary uses of SQL is querying data from databases. SQL provides a powerful and flexible syntax for filtering, sorting, grouping, and aggregating data. The SELECT statement is used to retrieve data from one or more tables based on specified criteria.
  - **SELECT**: It is used to retrieve data from the database

✓ **SQL Operators**

An operator is a reserved word or a character that is used to query our database in a SQL expression. To query a database using operators, we use a WHERE clause. Operators are necessary to define a condition in SQL, as they act as a connector between two or more conditions. The operator manipulates the data and gives the result based on the operator's functionality.

### SQL Arithmetic Operators:

Arithmetic operators are used to perform mathematical operations on numeric values in SQL.

**Common arithmetic operators include:**

- **Addition (+):** Adds two values.
- **Subtraction (-):** Subtracts one value from another.
- **Multiplication** (*): Multiplies two values.
- **Division (/):** Divides one value by another.
- **Modulus (%):** Returns the remainder of a division operation.

**Example:** SELECT 10 + 5 AS Sum, 10 - 5 AS Difference, 10 * 5 AS Product, 10 / 5 AS Quotient, 10 % 3 AS Remainder FROM dual;

### SQL Bitwise Operators:

Bitwise operators perform bit manipulations between two expressions of any of the data types of the integer data type category.

Bitwise operators convert two integer values to binary bits, perform the AND, OR, or NOT operation on each bit, producing a result. Then converts the result to an integer.

**Common bitwise operators include:**

- **Bitwise AND (&):** Performs a bitwise AND operation.
- **Bitwise OR (|):** Performs a bitwise OR operation**.**
- **Bitwise XOR (^):** Performs a bitwise XOR (exclusive OR) operation.
- **Bitwise NOT (~):** Performs a bitwise NOT operation (inverts the bits).
- **Bitwise left shift (<<):** Shifts the bits to the left by a specified number of positions.
- **Bitwise right shift (>>):** Shifts the bits to the right by a specified number of positions.

**Example:** SELECT 3 & 5 AS Bitwise_AND, 3 | 5 AS Bitwise OR, 3 ^ 5 AS Bitwise_XOR FROM dual;

This SQL statement performs bitwise operations on the decimal representations of the numbers 3 and 5 and then aliases the results as **Bitwise_AND**, **Bitwise_OR**, and **Bitwise_XOR**. Let's break down what each part does:

- **SELECT**: This is the SQL keyword used to retrieve data from a database.
- **3 & 5 AS Bitwise_AND**: The **&** operator performs a bitwise AND operation on the binary representations of the numbers 3 and 5. In binary, 3 is **0011** and 5 is **0101**. The bitwise AND operation compares each bit position and returns 1 if both bits are 1; otherwise, it returns 0. So, **0011 & 0101** results in **0001**, which is 1 in decimal. This result is aliased as **Bitwise_AND**.

- **3 | 5 AS Bitwise_OR**: The **|** operator performs a bitwise OR operation on the binary representations of the numbers 3 and 5. In binary, **3 | 5** results in **0111**, which is 7 in decimal. This result is aliased as **Bitwise_OR**.
- **3 ^ 5 AS Bitwise_XOR**: The **^** operator performs a bitwise XOR (exclusive OR) operation on the binary representations of the numbers 3 and 5. In binary, **3 ^ 5** results in **0110**, which is 6 in decimal. This result is aliased as **Bitwise_XOR**.

**SQL Compound Operators:**

Compound operators combine an assignment operation with another operation, such as arithmetic or bitwise operations.

They provide a shorthand way of performing an operation and assigning the result to a variable or column.

**Common compound operators include:**

- Assignment with addition (+=)
- Assignment with subtraction (-=)
- Assignment with multiplication (*=)
- Assignment with division (/=)
- Assignment with bitwise AND (&=)
- Assignment with bitwise OR (|=)
- Assignment with bitwise XOR (^=)
- Assignment with bitwise left shift (<<=)
- Assignment with bitwise right shift (>>=)

Example: UPDATE table_name SET column_name += 10 WHERE condition;

**SET column_name += 10:**

This part indicates that you want to update the values in the column_name by adding 10 to the existing values.

The += operator is a compound assignment operator that adds the specified value to the current value of the column. If you want to subtract, multiply, or divide, you would use -= (subtract), *= (multiply), or /= (divide) respectively.

**SQL Logical Operators**:

Logical operators are used to combine conditions in SQL queries to form more complex conditions.

These operators are typically used in WHERE clauses to filter rows based on multiple conditions

| Operator | Description |
|---|---|
| ALL | TRUE if all of the subquery values meet the condition |
| AND | TRUE if all the conditions separated by AND is TRUE |
| ANY | TRUE if any of the subquery values meet the condition |
| BETWEEN | TRUE if the operand is within the range of comparisons |
| EXISTS | TRUE if the subquery returns one or more records |

| | |
|---|---|
| IN | TRUE if the operand is equal to one of a list of expressions |
| LIKE | TRUE if the operand matches a pattern |
| NOT | Displays a record if the condition(s) is NOT TRUE |
| OR | TRUE if any of the conditions separated by OR is TRUE |
| SOME | TRUE if any of the subquery values meet the condition |

**Example:** SELECT * FROM table_name WHERE condition1 AND condition2;

- **Comparison operators**

Comparison operators can compare numbers or strings and perform evaluations. Expressions that use comparison operators do not return a number value as do arithmetic expressions. Comparison expressions return either 1, which represents true, or 0, which represents false.

Assume 'variable a' holds 10 and 'variable b' holds 20, then –

| Operator | Description | Example |
|---|---|---|
| = | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (a = b) is not true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (a != b) is true. |
| <> | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (a <> b) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (a > b) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (a < b) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (a >= b) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (a <= b) is true. |
| !< | Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true. | (a !< b) is false. |
| !> | Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true | (a!> b) is true. |

**Points to Remember**

- To interact with relational databases always require SQL programming language.
- You are recommended to use SQL sub-language to ensure the management of different access to the database system.
- To perform various computations and logical operations within queries you should always use SQL operators.

**Application of learning 3.1.**

Bright Future University is developing a new **Student Information Management System** to manage student records, academic performance and enrolment. The database structure that shows how the database will be implemented, was designed. the university needs to hire a database developer to implement database in MySQL by using SQL, to ensure that you are able to implement that database, you are asked to describe SQL Components (Sub-languages), their respective commands and the operators that can be used to Apply SQL commands.

## Indicative content 3.2: Application of DDL Commands

**Duration: 4 hrs**

### Theoretical Activity 3.2.1: Description of DDL Commands

**Tasks:**

1: Answer the following questions related to the DDL commands

1. Describe DDL Create Commands
2. Describe DDL Drop Commands
3. DDL Alter Commands
4. Describe DDL Truncate Commands

2: Present your findings to your classmates and trainer

3: Ask questions where necessary

4: For more clarification, read the key readings 3.2.1.

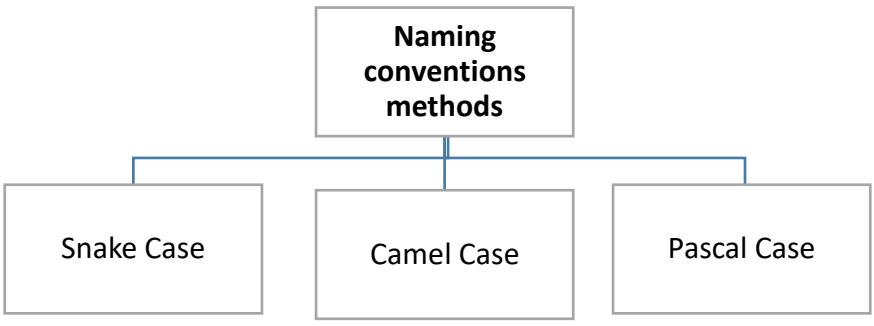
**Key readings 3.2.1.: Description of DDL Commands**

- **Description of DDL Commands**

Before starting DDL commands let's discuss on same important rules that will helps us to implement database.

- ✓ **Naming database object rules**
- **Naming Conventions Methods**

There are three widely adopted naming conventions structuring the names of objects in database design:




- **Snake Case**

Snake case uses underscores to separate word elements of a name, with all letters in lowercase.

For example:

| Employee |
| --- |
| employee_id<br>employee_name |

- **Camel Case**

Camel capitalizes each word in a name except the first one and concatenates the words without a separator:

**For example:**

| Employee |
| --- |
| employeeId<br>employeeName |

- **Pascal Case**

**Pascal case is the same as the camel case except that it capitalizes every word in a name (including the first one):**

| Employee |
| --- |
| EmployeeId<br>EmployeeName |

- **Naming Database**

- we should choose the names that accurately represent the purpose or content of the database objects.
- we should avoid special characters, spaces, and non-alphanumeric characters to prevent compatibility issues and misunderstandings.
- we shouldn't use reserved words for table and column names.

- **Naming Table**

- Table names should precisely convey the content or purpose of the data.
- Opting for singular nouns when naming tables, rather than plural nouns, is a recommended approach.

- **Naming column**

- column names should effectively and precisely define the data they contain
- column names should be singular nouns.
- We should avoid redundant words or information in column names, mainly when such details can be inferred from the broader context of the table or the presence of other columns.

- **Instructions for writing an SQL Commands**

  When writing an SQL statement remember that:
  - SQL Keywords Are Not Case-Sensitive
  - String and date values in statement must be quoted
  - Each SQL statement is ended by semi colon

✓ **Description of CREATE Commands**

Create is used to create database and database objects like; table, view, store procedure, synonym,..

- **Creating Database**

To create database, the **"CREATE DATABASE"** command is used.

**General syntax**: **Create database** name;

The command CREATE DATABASE name; is used in SQL to create a new database. Here's a breakdown of its components:

- CREATE DATABASE: This is the SQL keyword used to initiate the creation of a new database.
- database_name: This is a placeholder for the actual name you want to give to the new database. Replace database_name with the desired name for your database.

**Example**: **Create database** Ikaze_Online_Shop;

- **Creating Table**

To create database, the **"CREATE TABLE"** command is used.

**General syntax: CREATE TABLE** *table_name* (
*column1 datatype*,
*column2 datatype*,
*column3 datatype*,
....
**);**

- **CREATE TABLE table_name:** This specifies the command to create a table and assigns a name to the new table. Replace **table_name** with the name you want for your table.
- (: This opens the list of columns and their definitions for the table.
- **columnN datatype**: This defines the $N^{th}$ column in the table. columnN is the name of the column, and datatype specifies the type of data that the column will hold (e.g., VARCHAR, INT, DATE, etc.).
- );: This closes the list of column definitions.

**Example:**

```
CREATE TABLE Customer(
    CustomerID int(6),
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

**Creating Table with constraints at column level**

**General syntax:**

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    ...............
);
```

**Components**

**table_name**: Replace this with the name you want for your table.

**column1, column2, column3, ...**: These are the names of the columns in your table.

**datatype**: Specify the type of data the column will hold (e.g., INT, VARCHAR(255), DATE).

**constraint**: Add any constraints based on integrity (e.g., NOT NULL, UNIQUE, FOREIGN KEY).

**PRIMARY KEY (column1)**: Example of a constraint that sets column1 as the primary key of the table.

**Example:**

```
CREATE TABLE Customers(
    CustomerID int(6) Primary key,
    LastName varchar(255) Null,
    FirstName varchar(255) Null,
    Address varchar(255) Not Null,
    City varchar(255) Not Null
);
```

**Creating Table with constraints at table level**

**General syntax:**

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    ……..
    columnN datatype constraint,
```

*constraint1*
*constraint2*
*…….*
*constraintN*
*);*

**table_name:** The name of the table to be created. It should be unique within the database schema.

**column1 datatype constraint, column2 datatype constraint, ... columnN datatype constraint,**

- The name of the column. Column names should be unique within the table.
- **datatype:** Specifies the data type for the column (e.g., INT, VARCHAR, DATE, etc.), determining what kind of data can be stored in that column.
- **constraint:** Constraints can be applied to columns to enforce data integrity.

**Table-level constraints: Unlike column-level constraints, these apply to the entire table. Examples include:**

- **Composite PRIMARY KEY:** A primary key that consists of more than one column.
- **FOREIGN KEY constraint:** A constraint that links a column (or combination of columns) in this table to a primary key in another table.
- **UNIQUE constraint:** Ensures that the combination of values in a set of columns is unique across all rows.

**Example: CREATE TABLE** Orders (

Order_id Int(6) Primary Key,

Customer_id Int (7),

Prod_id Int(8),

Order_Quantity Int(8) Not Null,

Pricr_per_unit Int(5) not Null

Order_date Date Not Null,

```
FOREIGN KEY (customer_id) REFERENCES Customers(Cid),
FOREIGN KEY (Prod_id) REFERENCES Product (Pid)
);
```

- **Creating stored procedure**

A **stored procedure** in a database is a precompiled collection of one or more SQL statements that can be executed as a single unit. Stored procedures are created and stored within the database and can be reused multiple times.

**General syntax:** CREATE PROCEDURE procedure_name ([IN|OUT|INOUT] parameter_name data_type, ...) BEGIN SQL statements DML/DDL statements (INSERT, UPDATE, DELETE, etc.) END;

**Example:** The following example creates a stored procedure that finds all students specified by the input parameter sfname:

```
DELIMITER //
CREATE PROCEDURE student (
   IN name VARCHAR (255)
)
BEGIN
   SELECT *
   FROM student
   WHERE sfname = name;
END //
DELIMITER ;
```

- ✓ **Describing DROP Commands**

- **Drop Database**

**General syntax**: **Drop database** name;

The DROP DATABASE command in SQL is used to delete an existing database along with all the objects it contains, such as tables, views, stored procedures, and other database objects. This action is irreversible, meaning once the database is dropped, all the data and the structure of the database are permanently lost.

**Example: Drop database** Ikaze_Online_Shop;

- **Drop Table**

To remove a table from database

**General syntax: Drop table** table name;

- **DROP TABLE:** This is the SQL command used to delete a table.
- **table_name:** This is the name of the table you want to delete. The table name should correspond to an existing table in the database.

**Example: Drop table** Customer;

✓ **Describing ALTER Commands**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

**ALTER TABLE - ADD Column**

To add a column in a table, use the following syntax:

**General syntax**: **ALTER TABLE** *table_name* **ADD** *column_name datatype constraint*;

**Example: ALTER TABLE** Customers **ADD** Email varchar (255) unique not null;

**ALTER TABLE - DROP COLUMN**

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

**General syntax: ALTER TABLE** *table_name* **DROP COLUMN** *column_name*;

**Example: ALTER TABLE** Customers DROP COLUMN Email;

**ALTER TABLE - MODIFY COLUMN datatype and contraint**

To change the data type of a column in a table, use the following syntax:

**General syntax: ALTER TABLE** *table_name* MODIFY COLUMN *column_name datatype*;

**Example: ALTER TABLE** Persons **MODIFY COLUMN** DateOfBirth year;

**ALTER TABLE – CHANGE COLUMN NAME**

**General syntax: ALTER TABLE** *table_name* CHANGE current_ *column_name New_column_name datatype constraint*;

**Example:** Change the column name "FirstName"of customer table to Customer_FirstName varchar(255)Null.

- **ALTER TABLE** Customer **CHANGE** FirstName Customer_FirstName varchar(255)Null;

**ALTER TABLE – RENAME a Table**

**General Syntax: General syntax: ALTER TABLE** *table_name RENAME TO New_Table_Name;*

**Example:** Let's rename **Customer** table to **Customers.**

- **ALTER TABLE** *Customer RENAME TO Customers;*

✓ **Describing TRUNCATE commands**

The SQL **TRUNCATE TABLE** command is used to empty a table.

**General syntax: TRUNCATE TABLE** table_name;

- The TRUNCATE TABLE command in SQL is used to remove all rows from a table, effectively clearing the table of its data.
- TRUNCATE TABLE removes all rows at once and is generally faster because it does not log individual row deletions.

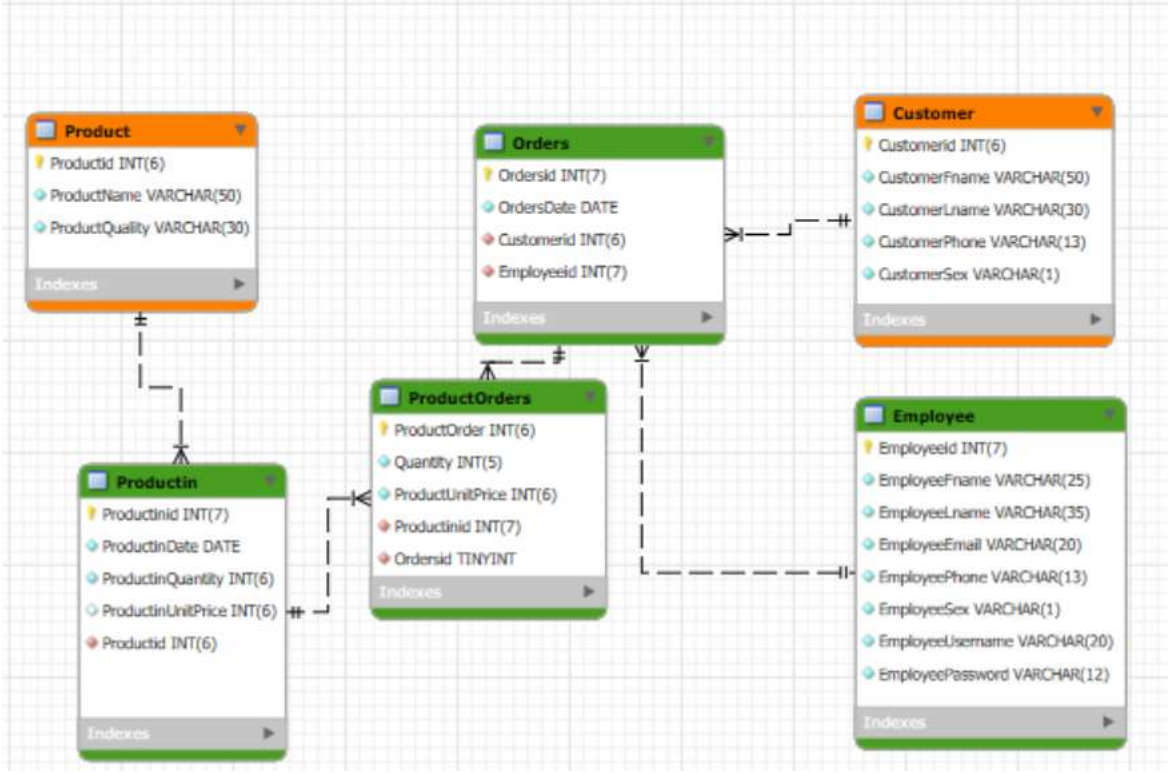
**Example: TRUNCATE TABLE** EMPLOYEE;

**Practical Activity 3.2.2: Applying DDL Commands**

**Task:**

1: Read the task below:

As database developer you are asked to go in computer lab to create database and tables by considering the provided physical database schema.




2: Referring to the key reading, perform the task described on task 2

3: Present your work to your trainer and a whole class

**Key readings 3.2.2: Applying DDL Commands**

✓ **Applying DDL Commands**

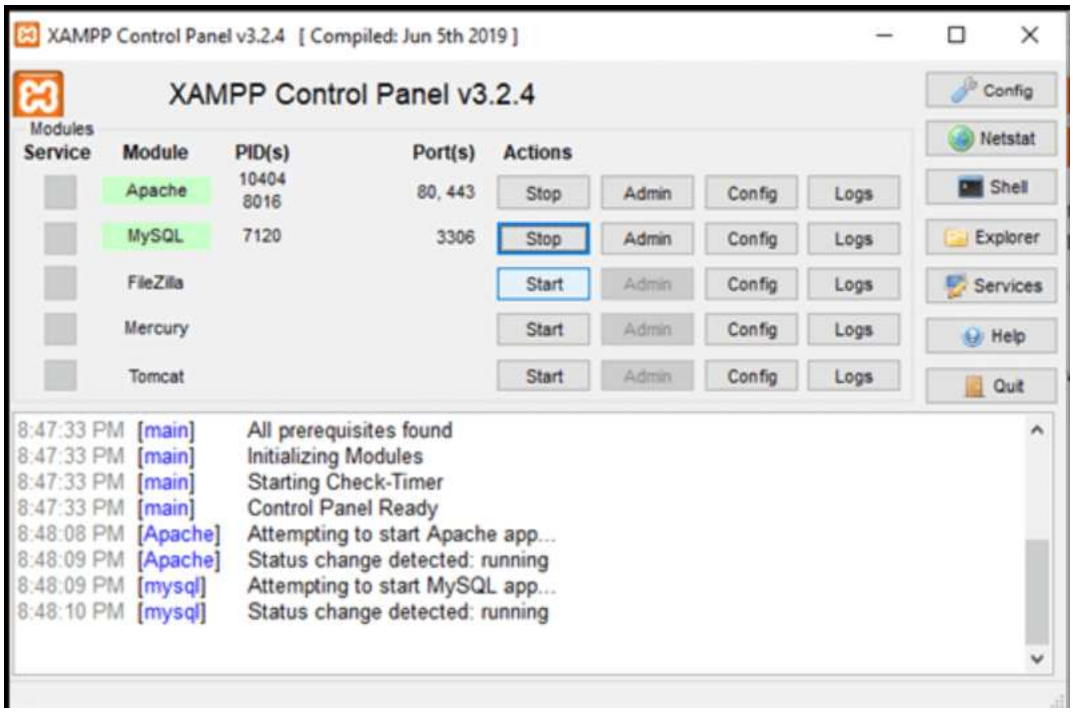
Creating database in MYSQL by using XAMPP Follow the below steps:

- **Step 1: Opening XAMPP**

In this section, we will show you how to open MySQL in XAMPP. Go to your system's XAMPP folder or simply click the XAMPP Icon to open it. The Control Panel is now visible, and you may use it to start or stop any module.

- **Step 2: Starting XAMPP**

Select the "**Start**" option for the Apache and MySQL modules, respectively. The user will see the following screen once it has started working:




- **Step 3: Accessing Admin**

Next, select the MySQL Module and click the "Admin" button. The user is immediately redirected to the following address in a web browser:

http://localhost/phpmyadmin

- **Step 4: Creating a Database**

Next, we will learn how to create database in XAMPP. Setting up an **XAMPP** database is straightforward, making it convenient for beginners to practice database management. A number of tabs appear, including Database, SQL, **User Accounts**, Export, **Import**,

Settings, and so on.

- Select the "**SQL**" tab from the drop-down menu.
- Write a statement to create database
- Click on to run statement.

**To add tables on created database, follow the below steps:**

**Step 5: Creating a Table**

- Click on Database name
- Click on SQL Menu
- Write an SQL Statement
- Run

**Points to Remember**

- Only command to Create database and database object in SQL is **CREATE** command.
- To drop database and database object in SQL, always use **DROP** command.
- Only command to make any modification (add column, modify column and drop column) on table is **ALTER** command.
- To delete the record in table by remaining the structure of table in SQL, always use **TRUNCATE** command.
- General syntax to Create database: **Create database** database_name;
- General syntax to create table: **CREATE TABLE** *table_name* (*column1 datatype constraint*, *column2 datatype constraint*,……., *columnN datatype constraint*, *constraint1, constraint2,* ……., *constraintN*);
- General syntax to Drop Database: **Drop database** database_name;
- **General syntax: Drop table** table name;
- General syntax to modify table structure:
  - ✓ Add Column: **ALTER TABLE** table_name **ADD** column_name datatype;
  - ✓ Modify column datatype and constraint: **ALTER TABLE** table_name **MODIFY COLUMN** datatype and constraint**;**
  - ✓ Drop column: **ALTER TABLE** table_name **DROP COLUMN** column_name**;**
  - ✓ Change Column name**: ALTER TABLE** table_name **CHANGE** Current_column_name New_column_Name Datatype and constraint**;**
  - ✓ Rename table: **ALTER TABLE** Table_Name **RENAME TO** New_Table_Name;
- General syntax Truncate Table: **TRUNCATE TABLE** table_name;

**Application of learning 3.2.**

Physical database schema for a university management system is provided to you. The schema outlines the structure for several tables, including Students, Courses, Enrollments, and Professors, along with their respective columns, data types, keys, and constraints. Your task is to use Data Definition Language (DDL) commands to build and maintain this database.

Initially, you'll need to **CREATE** the tables according to the schema, establishing all necessary relationships through primary and foreign keys. As the system evolves, you might need to **ALTER** existing tables to accommodate new requirements, such as adding a column to track students' graduation status or modifying the data type of the Course Description field to allow more text.

If a specific table, like Enrollments, is no longer needed due to a change in the system's design, you may be required to **DROP** it entirely. Additionally, before migrating data or resetting a table for new data, you might need to **TRUNCATE** tables such as Courses, clearing all records while retaining the table structure for future use.

## Indicative content 3.3: Apply DML Commands

**Duration: 4hrs**

### Theoretical Activity 3.3.1: Description of DML Commands

**Tasks:**

1: Answer the following questions related to the DML commands

1. Identify DML INSERT Commands
2. Describe DML UPDATE Commands
3. Elaborate DML DELETE Commands
4. Describe DML CALL Commands
5. Describe DML EXPLAIN CALL Commands
6. Describe DML LOCK Commands

2: Present your findings to your classmates and trainer

3: Ask questions where necessary

4: For more clarification, read the key readings 3.3.1.

**Key readings 3.3.1: Description of DML Commands**

- **Describing DML INSERT Command**

Insert is used to add records in a table

To add records in a table the "Insert into" is used

✓ **Command to insert one record**

**General syntax:** INSERT INTO tablename(column1,
column2,….columnn)VALUES(value1, value2,….valuen);

- **Tablename:** The name of the table where you want to insert data.
- **column1, column2, ….columnn:** These are the names of the columns where the values will be inserted. You can specify one or more columns, depending on how many columns in the table you want to insert data into.

The number of columns listed must match the number of values provided in the VALUES clause.

- **VALUES (value1, value2, ….valuen):** These are the values to be inserted into the respective columns. Each value corresponds to the column specified at the same position.

The values must match the data type of the columns they are being inserted into.

**Example:** Insert into customer(id,name,age)values(1,”Jeanette”,26);
**Is the same as using:**
Insert into customer values (1,”Jeanette”,26);
**Notes:** the above syntax is used to:
Insert data in all columns
Insert Data only in specified columns

✓ **Command to insert multiple records into a table at once**

**General syntax:** Insert into Table_name(column1,column2,….columnn) values(value1.1,value1.2,……value1.n),(value2.1,value2.2,……value2.n),(valuen.1,valuen.2,……valuen.n);

- **INSERT INTO** Table_name (column1, column2, …. columnn): Table_name: The name of the table where you want to insert the data.
- **(column1, column2, …. columnn):** The list of columns where the data will be inserted. These columns must exist in the table and should be in the same order as the values you are inserting.
- **VALUES:** This keyword indicates that you are specifying the data values that will be inserted into the columns listed earlier.
  **(value1.1, value1.2, …… value1.n):** This represents the first set of values corresponding to each column specified earlier. Each value will be inserted into the respective column for the first row.
  **(value2.1, value2.2, …… value2.n):** This is the second set of values for the second row. It follows the same structure as the first set but will be inserted as a new row.
  **(valuen.1, valuen.2, …… valuen.n):** This represents the nth set of values. It follows the same structure as the previous sets but corresponds to the nth row.

**Example:** Insert into customer(id,name,age)values(2,”john”,45),
(3,“Caline”,34),
(4,”JMV”,35),

(5,”Thomas”,35);

- **Describing DML UPDATE Commands**

**UPDATE:** It is used to update existing data within a table.

**General Syntax: UPDATE Table_name** SET column1 = value1, column2 = value2, .... columnn = valuen WHERE condition;

- **UPDATE Table_name**: **Table_name**: The name of the table where you want to update records.
- **SET column1 = value1, column2 = value2, .... columnn = valuen**: This specifies the columns to be updated and the new values to assign to them. Multiple columns can be updated in a single SET clause, separated by commas.
- **WHERE condition**: This clause specifies which rows should be updated. The condition determines which records are affected by the UPDATE. If the WHERE clause is omitted, all rows in the table will be updated.

**Example:** UPDATE Customers SET ContactName='Alfred Schmidt', City='Frankfurt' WHERE CustomerID = 1;

- **Describing DML DELETE Commands**

**DELETE:** It is used to delete records from a database table.

**General Syntax:** DELETE FROM Table_name WHERE condition;

Example: DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

- **Describing DML CALL Commands**

The CALL statement in SQL is used to execute a stored procedure. Stored procedures are precompiled collections of SQL statements that can perform operations like data manipulation, business logic, and more. The CALL statement is typically supported in SQL databases like MySQL, Oracle, and others.

**General Syntax:** CALL procedure_name(argument1, argument2, .... argumentn);

**Components Explained:**

- **CALL**: The keyword used to invoke a stored procedure.
- **procedure_name**: The name of the stored procedure you want to execute.
- **(argument1, argument2, .... argumentn)**:
  - These are the arguments or parameters that the stored procedure takes.
  - If the procedure does not require any parameters, the parentheses are still

included but left empty: CALL procedure_name();.

- The parameters can be of different types:
  - **IN** parameters: Input values passed to the procedure.
  - **OUT** parameters: Variables that will store output values from the procedure.
  - **INOUT** parameters: Variables that pass input to and receive output from the procedure.

**Example:** Assume you have a stored procedure named AddEmployee that takes three parameters: firstName, lastName, and salary.

**Answer:** CALL AddEmployee('John', 'Doe', 50000);

- **Describing DML EXPLAIN CALL Commands**

The EXPLAIN CALL statement is used in MySQL to provide information about the execution plan for a stored procedure or function. It helps in understanding how MySQL executes a stored procedure, which can be useful for performance tuning and debugging.

General syntax: EXPLAIN CALL stored_procedure_name(parameters);

Suppose you have a stored procedure named get_user_data that takes one parameter, user_id. You can analyze the execution plan as follows:

EXPLAIN CALL get_user_data(1);

**Notes:**

- The EXPLAIN command will show you the execution plan of the queries inside the stored procedure, providing insight into how the procedure executes.
- Some database systems might not support EXPLAIN directly on CALL statements. In such cases, you'd need to look into the specific database documentation or use alternative methods like logging or profiling.

- **Describing DML LOCK Commands**

In the context of databases, a **lock** is a mechanism used to control access to data in a database to ensure consistency and prevent conflicts when multiple transactions are trying to access or modify the same data simultaneously.

✓ **There are several types of locks, including:**

- **Shared Lock (S-lock)**: This allows multiple transactions to read a resource (such as a row or table) but prevents any transaction from modifying it. Shared locks

are non-exclusive, meaning multiple transactions can hold a shared lock on the same resource.

- **Exclusive Lock (X-lock)**: This allows a transaction to both read and modify a resource but prevents other transactions from accessing that resource. Exclusive locks are exclusive, meaning only one transaction can hold an exclusive lock on a resource at a time.
- **Update Lock (U-lock)**: This is a special type of lock that prevents deadlocks when a transaction intends to update a resource. An update lock is compatible with shared locks but not with other update or exclusive locks.

**General syntax**: LOCK TABLES table_name [AS alias] lock_type [, table_name [AS alias] lock_type] ...;

UNLOCK TABLES;

- **table_name**: The name of the table to be locked.
- **alias**: An optional alias for the table.
- **lock_type**:
  - READ: Allows reading but prevents writing.
  - WRITE: Prevents both reading and writing by other connections.

Example: LOCK TABLES orders READ, customers WRITE;

UNLOCK TABLES;

**Practical Activity 3.3.2: Apply DML Commands**


**Task:**

You are requested to go in computer lab to perform the task described below:

Bright Future University is in the process of implementing a new database system to manage its academic operations, including student enrollment, class assignments, and professor-course allocations. The database structure includes six key tables:

I. **Student**(StudentID INT(10)PK, StudentName varchar(100), StudentEmail varchar(100), StudentPhone varchar(20), StudentBirthDay date)
II. **Subject**(SubjectID int(7)PK, SubjectName varchar(100))

III. **Professor**(ProfessorIDint(7)PK,ProfessorNamevarchar(100),ProfessorEmail varchar(100), ProfessorPhone varchar(20) )
IV. **ProfessorSubject**(ProfessorSubjectID Int(10), ProfessorID INT(10)FK, SubjectID INT(10)FK)
V. **Class** (ClassID INT(10)PK, ClassCode varchar(20),BeginDate date, EndDate Date, SubjectID int(7)FK, ProfessorID int(7)Fk)
VI. **ClassStudent**(ClassStudentID int(10)PK, ClassID int(10)FK, StudentID int(10)FK)

which are already created with appropriate relationships and constraints.

As the database designer for this project, your task is to use these existing tables and perform the following tasks:

1. **Insert Four Sample Records:**

   Add four sample records for each of the following tables: Student, Subject, Professor, ProfessorSubject, Class, and ClassStudent. Each record should represent actual student enrollment in courses, professors assigned to subjects, and class schedules.

2. **Delete Two Records from the ProfessorSubject Table:**

   After inserting the sample records, identify and delete two records from the ProfessorSubject table, ensuring that the relationships between professors and subjects are maintained in the system.

3. **Update Two Records in the Student Table:**

   Choose two student records from the Student table and update specific fields such as StudentEmail or StudentPhone to reflect new information.

2: Insert, update, and delete data in database table

4: Present your work to your trainer and a whole class

4: Ask clarification where necessary

5: Perform the task provided in application of learning 3.3.

**Key readings 3.3.2: Apply DML Commands**

- **Apply DML Commands**

✓ **Steps to insert data in a database table**

**Select the Database**

If you have multiple databases, you must first select the one where the table resides.

**Check the Table Structure (Optional)**

It's a good idea to inspect the table structure to understand what columns you need to insert data into.

- **Select SQL Menu**
- **Insert Data into the Table**

Write INSERT statement to add a new row (or rows) of data into the table.

- **Run the statement**

To run the statement, click on GO or RUN

- **Verify the Data Insertion**

Click on table name where data was expected to be inserted> in the menu list click browse

- ✓ **Steps to update database data**
- **Select the Database by clicking its name**

Choose the database where the table is located.

  - Select table by clicking table name to Check the Table Structure (Optional)
  - Select SQL menu
  - Write a command to update database data
  - Click on go to run the command
  - Verify if the updated data is applied to database by clicking on database table name>browse

- ✓ **Steps to delete database data**
  - Select the Database by clicking its name: choose the database where the table is located.
  - Select table by clicking table name to Check the Table Structure (Optional)
  - Select SQL menu
  - Write a command to delete database record(s)
  - Click on go to run the command
  - Verify if the deleted data is applied to database by clicking on database table name>browse

**Points to Remember**

- Only command to insert data in database table in SQL is **INSERT** command.
- To modify data from database table in SQL always use **UPDATE** command.

- Use **DELETE** command to delete data from database table in SQL
- Only command to call a stored procedure from database in SQL is **CALL** command.
- To explain call from database you are recommended to use **EXPLAIN CALL** command.
- Use **LOCK** command to lock database table.
- **To insert data in database table use a General Syntax:** INSERT INTO tablename(column1, column2,….columnn)VALUES(value1, value2,….valuen);
- **To update data from database, use a General Syntax: UPDATE Table_name** SET column1 = value1, column2 = value2, .... columnn = valuen WHERE condition;
- **To delete data from atabase, use a General Syntax:** DELETE FROM Table_name WHERE condition;
- **To call a stored procedure from database use a General Syntax:** CALL procedure_name(argument1, argument2, .... argumentn);
- **To explain call from database, use a General Syntax:** EXPLAIN CALL stored_procedure_name(parameters);
- **To lock database table, use a General Syntax:** LOCK TABLES table_name [AS alias] lock_type [, table_name [AS alias] lock_type] ...;
- DML commands, doesn't return output when there are errors in the statement
- When there is an error in the statement follow instruction Provided by MySQL to correct that error
- Respect integrity rules when Inserting, Deleting, and updating data from database

**Application of learning 3.3.**

SKY secondary school located in Rusizi district, Murenge sector, Bungo village uses a School Management System (SMS) to handle its day-to-day operations. The SMS database is already set up, containing multiple tables to manage students, teachers, courses, and enrollments.

As technician, you are asked to manipulate the data in the provided database by performing various operations such as inserting new records, updating existing records, and deleting records as needed.

**Database Structure:** Below is the simplified structure of the SMS database:

1. **Students Table** (students):
   i.student_id (Primary Key, INT)
   ii.first_name (VARCHAR)
   iii.last_name (VARCHAR)
   iv.date_of_birth (DATE)
   v.enrollment_date (DATE)

vi.grade_level (INT)

2. **Teachers Table** (teachers):
    i. teacher_id (Primary Key, INT)
    ii. first_name (VARCHAR)
    iii. last_name (VARCHAR)
    iv. subject (VARCHAR)
    v. hire_date (DATE)
3. **Courses Table** (courses):
    i. course_id (Primary Key, INT)
    ii. course_name (VARCHAR)
    iii. teacher_id (Foreign Key, INT)
4. **Enrollments Table** (enrollments):
    i. enrollment_id (Primary Key, INT)
    ii. student_id (Foreign Key, INT)
    iii. course_id (Foreign Key, INT)
    iv. enrollment_date (DATE)

**Tasks:**

**1. Insert New Data:**

- A new student, *Emma Watson*, has enrolled in the school. Insert her information into the students table. She was born on *2008-09-15*, enrolled on *2024-09-01*, and is in grade 10.
- A new teacher, *Dr. Sarah Smith*, has been hired to teach *Physics*. Insert her information into the teachers table. Her hire date is *2024-08-20*.

**2. Update Existing Data:**

- *John Doe* (with student_id = 3) has advanced from grade 9 to grade 10. Update his grade_level in the students table.
- *Mrs. Jane Roberts* (with teacher_id = 5) has been promoted and will now teach *Mathematics* instead of *History*. Update her subject in the teachers table.

**3. Delete Data:**

- A course with the course_id of 7 (*Art History*) is no longer offered at the school. Remove this course from the courses table.
- A student, *Mark Wilson* (with student_id = 12), has transferred to another school. Delete his record from the students table.

## Indicative content 3.4: Apply DQL Commands

**Duration: 3 hrs**

### Theoretical Activity 3.4.1: Description of DQL Commands

**Tasks:**

1: Read the following questions and answer to them accordingly

i. Describe DQL SELECT Command
ii. What are SQL aggregation functions and their use?
iii. What are SQL clauses used in SQL statement?

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 3.4.1.

**Key readings 3.4.1.: Description of DQL Commands**

- **Description of DQL Commands**

✓ **SELECT**

The SELECT statement is a core component of **Data Query Language (DQL)** in SQL. DQL is used primarily for querying and retrieving data from a database. The SELECT statement enables users to specify which data they want to extract and how it should be presented.

**General syntax:** SELECT column1, column2, ... FROM table_name;

- **SELECT**: The keyword that indicates you want to retrieve data.
- **column1, column2, ...**: The columns you want to retrieve data from. You can list specific column names or use * to select all columns.
- **FROM**: The keyword that specifies the table from which to retrieve the data.
- **table_name**: The name of the table containing the data.

**Example 1: Selecting Specific Columns**

- SELECT first_name, last_name FROM employees;

**Example 2: Selecting All Columns**

- SELECT * FROM employees;

This query retrieves all columns from the employee's table.

✓ **SQL aggregate function**

An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

Aggregate functions are often used with the GROUP BY clause of the SELECT statement. The GROUP BY clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

Let consider employee table:

**Employee table**

| Id | Name | Salary |
|---|---|---|
| 1 | A | 802 |
| 2 | B | 403 |
| 3 | C | 604 |
| 4 | D | 705 |
| 5 | E | 606 |
| 6 | F | NULL |

- MIN() - returns the smallest value within the selected column

Example: Determine the lowest salary

**SELECT MIN**(Salary) **AS** LowestSalary **FROM** Employee;

Output:

| LowestSalary |
|---|
| 403 |

- MAX() - returns the largest value within the selected column

Example: Get the highest salary

**SELECT MAX(**Salary) **AS** HighestSalary **FROM** Employee;

Output:

| HighestSalary |
|---|
| 802 |

- COUNT() - returns the number of rows in a set

Example: count the number of employees

**SELECT COUNT**(*) **AS** TotalEmployees **FROM** Employee;

Output:

| **TotalEmployees** |
|---|
| 6 |

- SUM() - returns the total sum of a numerical column

  Example: Calculate the total salary

  **SELECT SUM**(Salary) **AS** TotalSalary **FROM** Employee;

  Output:

| **TotalSalary** |
|---|
| 3120 |

- AVG() - returns the average value of a numerical column

Example: Find the average salary

**SELECT AVG**(Salary) **AS** AverageSalary **FROM** Employee;

Output:

| **AverageSalary** |
|---|
| 624 |

**Notes:** Aggregate functions ignore null values (except for COUNT()).

- ✓ **SQL clause**

An SQL clause is a component of an SQL statement that specifies a particular action or condition.

- Clauses are used to define various aspects of the SQL query
- Each clause has a specific function within the query and can be combined with other clauses to create complex queries.

- ✓ **WHERE clause**

The WHERE clause is used with the SELECT, UPDATE, and DELETE.

The WHERE clause used with the SELECT statement to extract only records that fulfill specified conditions:

**General Syntax**: SELECTcolumn_list FROM table_name WHERECondition;

**Example:** SELECT * FROM Customers WHERE Country='Mexico';

SQL requires single quotes around text values (most database systems will also allow double quotes). However, numeric fields should not be enclosed in quotes:

SELECT * FROM Customers WHERE CustomerID=1;

- **Comparison operators in where clause**

**Example:** SELECT * FROM Customers WHERE CustomerID=1;

This query returns all information (all columns) from the Customers table for the customer who has a CustomerID of 1.

- **LIKE operator in Where clause**

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator:

- The percent sign (%)
- The underscore (_)

The percent sign represents zero, one, or multiple characters. The underscore represents a single number or character. The symbols can be used in combinations.
Here are number of examples showing WHERE part having different LIKE clause with '%' and '_' operators:

| **Statement** | **Description** |
|---|---|
| WHERE SALARY LIKE '200%' | Finds any values that start with 200 |
| WHERE SALARY LIKE '%200%' | Finds any values that have 200 in any position |
| WHERE SALARY LIKE '_00%' | Finds any values that have 00 in the second and third positions |
| WHERE SALARY LIKE '2_%_%' | Finds any values that start with 2 and are at least 3 characters in length |
| WHERE SALARY LIKE '%2' | Finds any values that end with 2 |
| WHERE SALARY LIKE '_2%3' | Finds any values that have a 2 in the second position and end with a 3 |
| WHERE SALARY LIKE '2___3' | Finds any values in a five-digit number that start with 2 and end with 3 |

Let us take a real example, consider the CUSTOMERS table having the following records:

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | RAAMESH | 32 | Ahmedabad | 2000 |
| 2 | KHILAN | 25 | Delhi | 1500 |
| 3 | KAUSHIK | 23 | Kota | 2000 |
| 4 | Chaitali | 25 | Mumbai | 6500 |
| 5 | Hardik | 27 | Bhopal | 8500 |
| 6 | Komal | 22 | MP | 4500 |
| 7 | Muffy | 24 | iNDORE | 10000 |

Following is an example, which would display all the records from CUSTOMERS table where SALARY starts with 200:

**SELECT * FROM CUSTOMERS WHERE SALARY LIKE '200%';**

This would produce the following result:

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | Ramesh | 32 | Ahmedabad | 2000 |
| 3 | kaushik | 23 | Kota | 2000 |

- **Boolean operators**

The SQL AND&OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators.

- **The AND Operator**

The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause. All conditions separated by the AND must be TRUE

- **The OR Operator**

The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

The only any ONE of the conditions separated by the OR must be TRUE.

- **Range in where clause**

The WHERE clause in SQL can be used with a range to filter data that falls within a specific set of values. You can use the BETWEEN operator or comparison operators like >= and <= to specify the range.

The **BETWEEN operator** is used to filter the results within a specified range, inclusive of the boundary values.

**Example:** SELECT * FROM Orders WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31';

This query retrieves all orders from the Orders table where the OrderDate is within the year 2023.

- **Order by**

The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort query results in ascending order by default.

To sort the records in descending order, use the DESC keyword.

**General Syntax:** SELECT*column1*, *column2,.....* FROM *table_name* ORDERBY*column1, column2, ...* ASC|DESC;

**Examples**

- SELECT * FROM Customers ORDERBY Country;

- SELECT * FROM Customers ORDERBY Country DESC;

**ORDER BY Several Columns**

- SELECT * FROM CustomersORDERBY Country, CustomerName;
- SELECT * FROM CustomersORDERBY Country ASC, CustomerName DESC;

**GROUP BY clause**

The GROUP BY statement group rows that have the same values into summary rows like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

**General Syntax:**

SELECT *column_name(s)*
FROM *table_name*
GROUPBY*column_name(s)*
ORDERBY*column_name(s);*

**Example**:

- SELECT NAME, SUM (SALARY) FROM CUSTOMERS GROUP BY NAME;

Now, let us look at a table where the CUSTOMERS table has the following records with duplicate names

- SELECT NAME, SUM (SALARY) FROM CUSTOMERS GROUP BY NAME;

**HAVING clause**

- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.
- The HAVING Clause enables you to specify conditions that filter which group results appear in the results.
- The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.
- The HAVING clause must follow the GROUP BY clause in a query and must also precedes the ORDER BY clause if used.

**Example:** SELECT ID, NAME, AGE, ADDRESS, SALARY FROM CUSTOMERS GROUP BY age HAVING COUNT (age)>=2;

**Out put:**

| ID | NAME | AGE | ADDRESS | Salary |
|---|---|---|---|---|
| 2 | khilan | 25 | Delhi | 1500 |

### SQL JOIN

SQL JOIN clause is used to query and access data from multiple tables by establishing logical relationships between them. It can access data from multiple tables simultaneously using common key values shared across different tables.

We can use SQL JOIN with multiple tables. It can also be paired with other clauses, the most popular use will be using JOIN with WHERE clause to filter data retrieval.

**Example:** Consider the two tables below as follows:

**Student:**

| ROLL_NO | NAME | ADDRESS | PHONE | Age |
|---|---|---|---|---|
| 1 | HARSH | DELHI | XXXXXXXXXX | 18 |
| 2 | PRATIK | BIHAR | XXXXXXXXXX | 19 |
| 3 | RIYANKA | SILIGURI | XXXXXXXXXX | 20 |
| 4 | DEEP | RAMNAGAR | XXXXXXXXXX | 18 |
| 5 | SAPTARHI | KOLKATA | XXXXXXXXXX | 19 |
| 6 | DHANRAJ | BARABAJAR | XXXXXXXXXX | 20 |
| 7 | ROHIT | BALURGHAT | XXXXXXXXXX | 18 |
| 8 | NIRAJ | ALIPUR | XXXXXXXXXX | 19 |

**StudentCourse** :

| COURSE_ID | ROLL_NO |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 2 | 3 |
| 3 | 4 |
| 1 | 5 |
| 4 | 9 |
| 5 | 10 |
| 4 | 11 |

Both these tables are connected by one common key (column) i.e ROLL_NO.

We can perform a JOIN operation using the given SQL query:

SELECT s.roll_no, s.name, s.address, s.phone, s.age, sc.course_id

FROM Student s JOIN StudentCourse sc ON s.roll_no = sc.roll_no;

**Output:**

| ROLL_NO | NAME | ADDRESS | PHONE | AGE | COURSE_ID |
|---|---|---|---|---|---|
| 1 | HARSH | DELHI | XXXXXXXXX | 18 | 1 |
| 2 | PRATIK | BIHAR | XXXXXXXXX | 19 | 2 |
| 3 | RIYANKA | SILGURI | XXXXXXXXX | 20 | 2 |
| 4 | DEEP | RAMNAGAR | XXXXXXXXX | 18 | 3 |
| 5 | SAPTARHI | KOLKATA | XXXXXXXXX | 19 | 1 |

Types of JOIN in SQL

There are many types of Joins in SQL. Depending on the use case, you can use different type of SQL JOIN clause. Here are the frequently used SQL JOIN types:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- NATURAL JOIN

**SQL INNER JOIN**

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

**General Syntax: for SQL INNER JOIN is:**

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching_column = table2.matching_column;

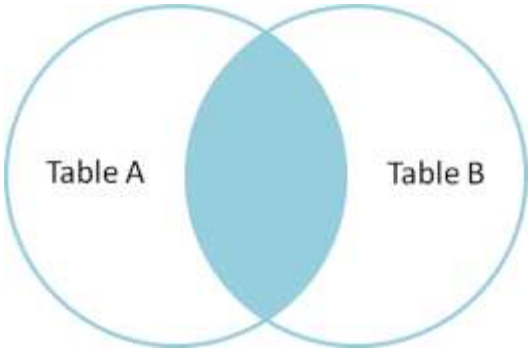
**Here,**

**table1:** First table.

**table2:** Second table

matching_column: Column common to both the tables.

Note: We can also write JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.




**Example:** Let's look at the example of INNER JOIN clause, and understand it's working.

This query will show the ID, names and age of students enrolled in different courses.

**SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student INNER JOIN StudentCourse**

**ON Student.ROLL_NO = StudentCourse.ROLL_NO;**

**Output:**

| COURSE_ID | NAME | Age |
|---|---|---|
| 1 | HARSH | 18 |
| 2 | PRATIK | 19 |
| 2 | RIYANKA | 20 |
| 3 | DEEP | 18 |
| 1 | SAPTARHI | 19 |

- **SQL LEFT JOIN**

LEFT JOIN returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

**General Syntax of LEFT JOIN in SQL is**:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
```

LEFT JOIN table2
ON table1.matching_column = table2.matching_column;

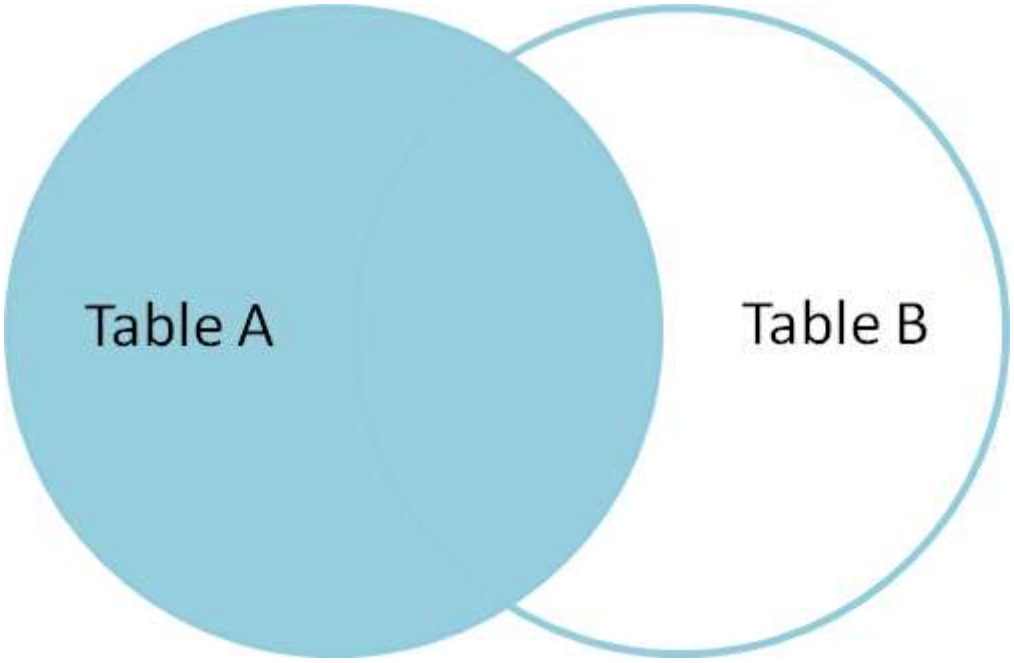
Here,

**table1:** First table.

**table2:** Second table

**matching_column:** Column common to both the tables.

Note: We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are the same.




**Example:** Let's look at the example of LEFT JOIN clause, and understand it's working

⇒ SELECT Student.NAME,StudentCourse.COURSE_ID

FROM Student

LEFT JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

**Output:**

| NAME | COURSE_ID |
|---|---|
| HARSH | 1 |
| PRATIK | 2 |
| RIYANKA | 2 |
| DEEP | 3 |
| SAPTARHI | 1 |
| DHANRAJ | *NULL* |
| ROHIT | *NULL* |
| NIRAJ | *NULL* |

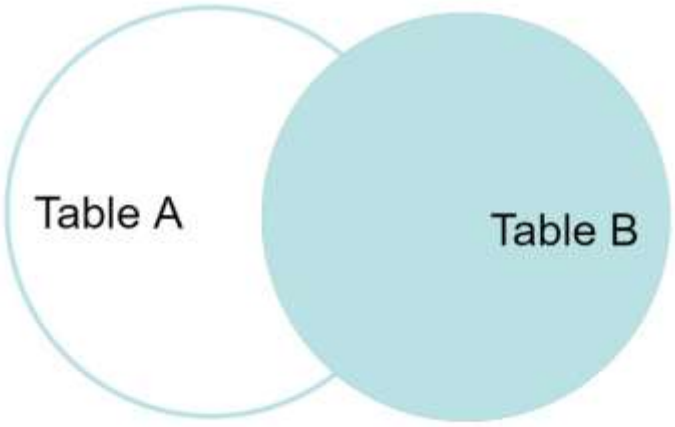
- **SQL RIGHT JOIN**

RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. It is very similar to LEFT JOIN For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

**The syntax of RIGHT JOIN in SQL is:**

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Here,
**table1**: First table.
**table2**: Second table
**matching_column**: Column common to both the tables.
**Note**: We can also use **RIGHT OUTER JOIN** instead of RIGHT JOIN, both are the same.




**Example:** Let's look at the example of RIGHT JOIN clause, and understand it's working

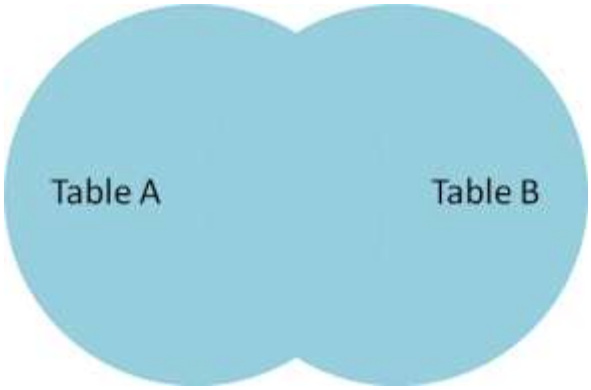
```
SELECT    Student.NAME,StudentCourse.COURSE_ID
FROM      Student
RIGHT     JOIN    StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

**Output:**

| NAME | COURSE_ID |
|---|---|
| HARSH | 1 |
| PRATIK | 2 |
| RIYANKA | 2 |
| DEEP | 3 |
| SAPTARHI | 1 |
| *NULL* | 4 |
| *NULL* | 5 |
| *NULL* | 4 |

- **SQL FULL JOIN**

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.




**General Syntax:**

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

**Here,**

- table1: First table.
- table2: Second table
- matching_column: Column common to both the tables.

**Example:** Let's look at the example of FULL JOIN clause, and understand it's working

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
FULL JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

**Out put:**

| NAME | COURSE_ID |
|---|---|
| HARSH | 1 |
| PRATIK | 2 |
| RIYANKA | 2 |
| DEEP | 3 |
| SAPTARHI | 1 |
| DHANRAJ | NULL |

| | |
|---|---|
| ROHIT | NULL |
| NIRAJ | NULL |
| NULL | 4 |
| NULL | 5 |
| NULL | 4 |

- **SQL Natural join (?)**

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Both tables must have at least one common column with same column name and same data type.

The two table are joined using Cross join.

DBMS will look for a common column with same name and data type Tuples having exactly same values in common columns are kept in result.

Natural join Example:

Look at the two tables below- Employee and Department

| **Employee** | | |
|---|---|---|
| **Emp_id** | **Emp_name** | **Dept_id** |
| **1** | **Ram** | **10** |
| **2** | **Jon** | **30** |
| **3** | **Bob** | **50** |

| **Dept_name** |
|---|
| **IT** |
| **HR** |
| **TIS** |

**Problem: Find all Employees and their respective departments.**

**Solution: Select** *from employee nature join department;

**Out put:**

| Emp_id | Emp_name | Dept_id | Dept_name |
|---|---|---|---|
| 1 | Ram | 10 | IT |
| 2 | Jon | 30 | HR |

The SQL NATURAL JOIN is a type of EQUI JOIN and is structured in such a way that, columns with the same name of associated tables will appear once only

**Practical Activity 3.4.2: Apply DQL Commands**


**Task:**

1: Read key reading 3.4.2

2: Read key read the below task:

**Customer table:**

| Customerid | CustomerFname | CustomerLname | CustomerPhone | CustomerSex |
|---|---|---|---|---|
| 1 | Anne | HAWA | 567 | F |
| 2 | John | RWEMA | 789 | M |
| 3 | Regis | MANZI | 678 | M |
| 4 | Charles | RWEMA | 780 | M |
| 5 | Kellia | UWASE | +250783345459 | F |

Employee table

| EmployeeId | EmployeeFname | EmployeeLname | EmployeeEmail | EmployeeSex | EmployeeUserName | EmployeePassword | EmployeePhone |
|---|---|---|---|---|---|---|---|
| 1 | Bonaventure | HABIMANA | habventure@gmail.c | M | k89 | kln89 | 409 |
| 2 | Aime | KEZA | keme@gmil.com | F | F789 | Yun | +250788888888 |

**Orders table**

| OrdersId | OrdersDate | Customerid | Employeeid |
|---|---|---|---|
| 1 | 2014-09-17 | 4 | 1 |
| 2 | 2014-09-18 | 4 | 2 |
| 3 | 2014-09-19 | 2 | 2 |
| 4 | 2014-10-22 | 3 | 2 |
| 5 | 2024-10-17 | 2 | 1 |
| 6 | 2024-11-12 | 5 | 2 |

**Based on the provided data stored in database;**

b. Make a report of all customers we have in database arrange their first names ascending
c. Make a report that displays sex and number of customers for each sex
d. Make a report of customers who ordered more than one time
e. Generate a report that displays a customer whose last name is RWEMA and Last name is John

3: Referring to the key reading, perform the task 2

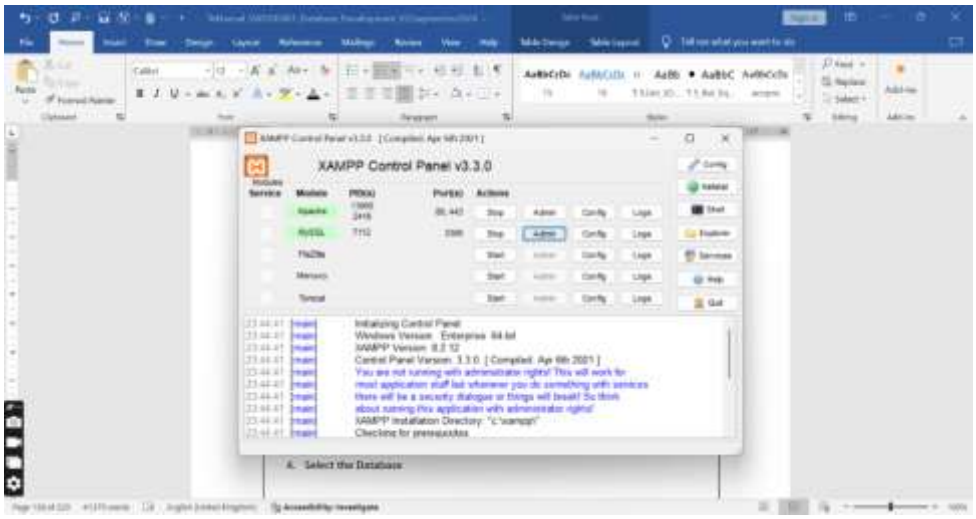
4: Present your work to your trainer and a whole class

**Key readings 3.4.2: Apply DQL Commands**

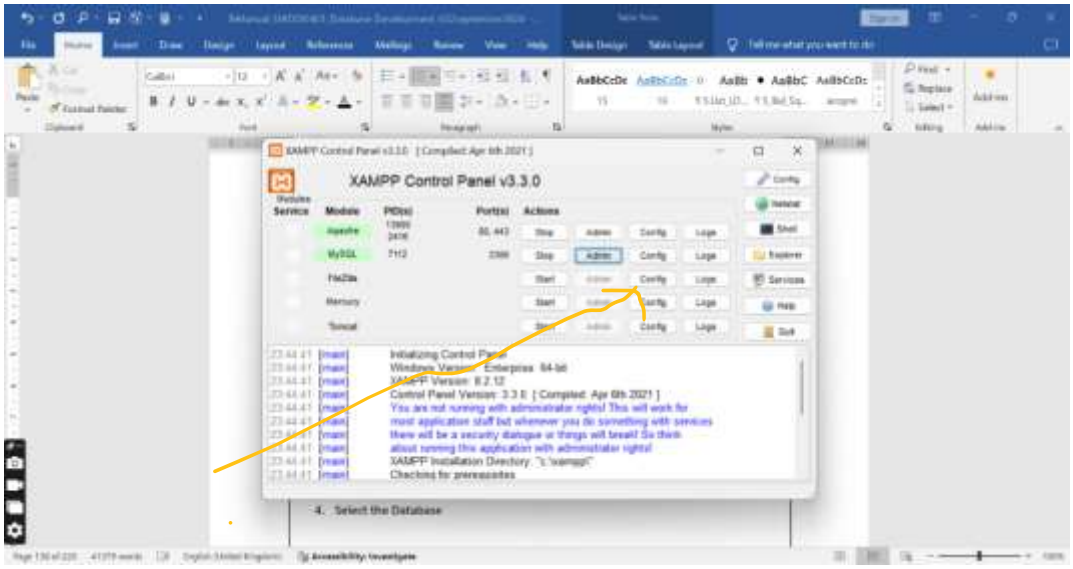
- **Apply DQL Commands**

✓ To select data from database table, follow the steps below:

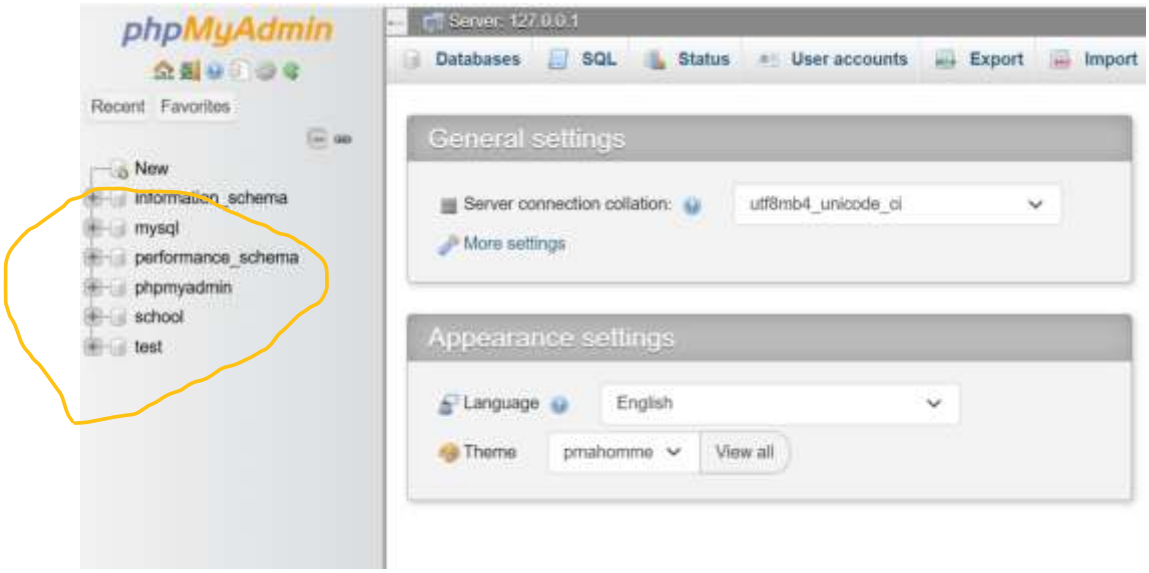
- Open xampp by double clicking on its icon
- Start MySQL and Apache



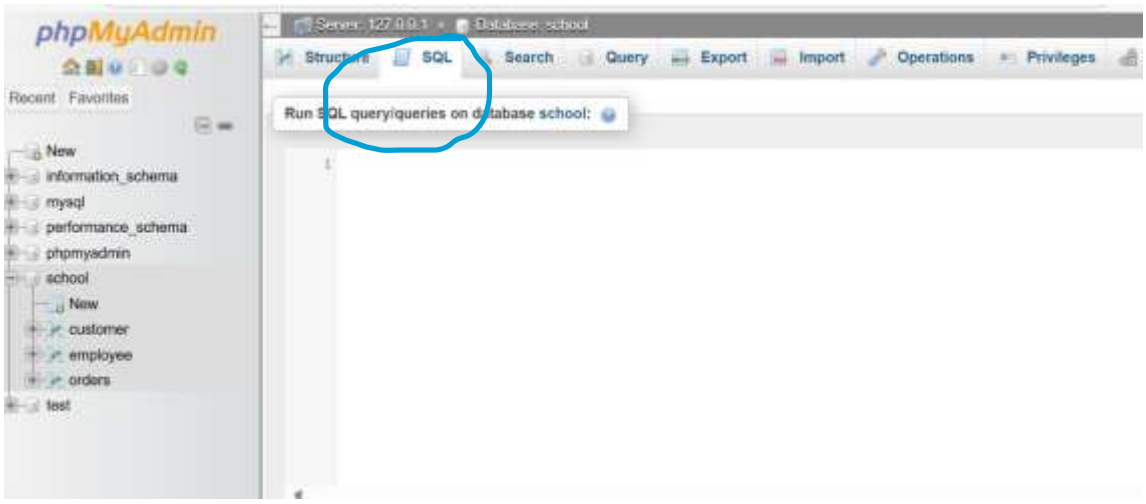

- Run MySQL as Admin



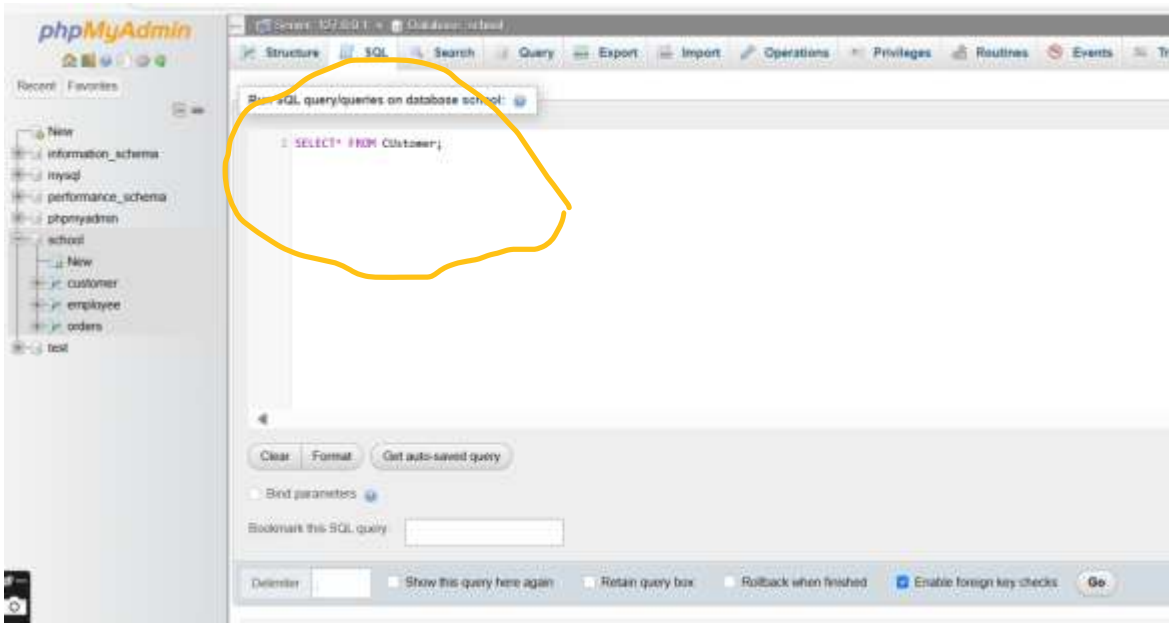

Select the Database



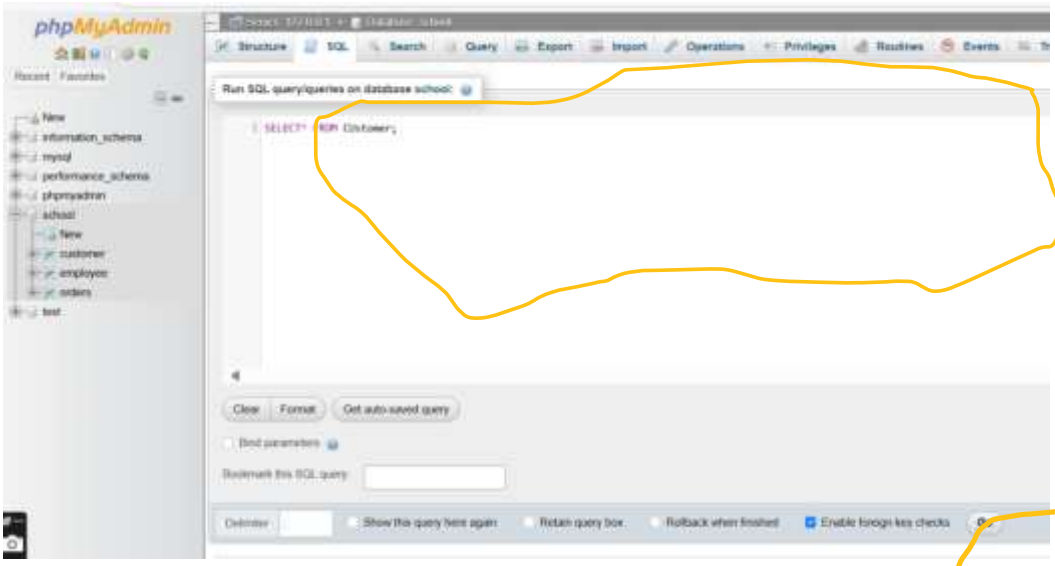

Select SQL Menu



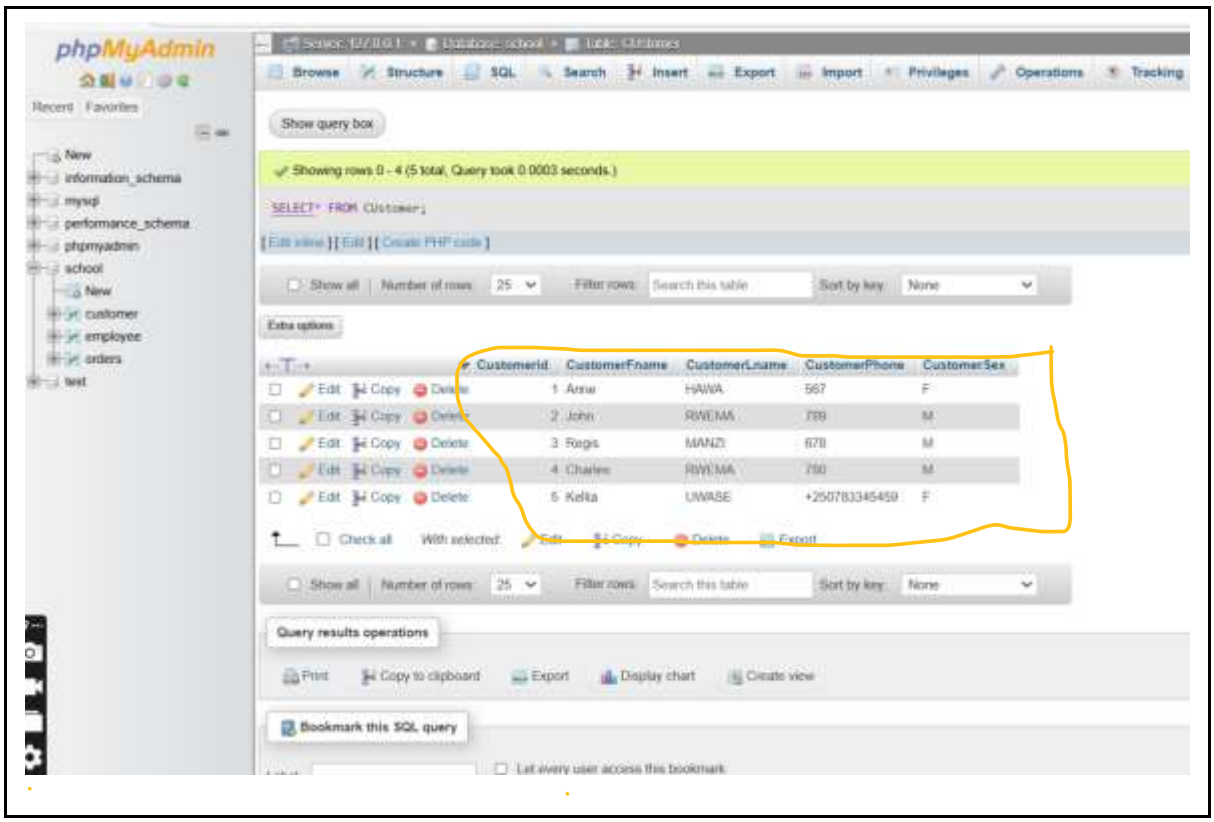
- Type a statement to select data




- Run it by clicking on GO




- Show the result set




**Points to Remember**

- It is mandatory to use **SELECT** command to retrieve data from SQL database.
- To performs a calculation on a set of values and returns a single value you are recommended to use an aggregate function.
- To Ensure that complex operations are performed correctly, you are recommended to use SQL clauses.
- To retrieve data from database by select command follow the general syntax: **SELECT** column1, column2…… **FROM** table_name;
- To select all columns of the table always use: **SELECT** * **FROM** table_name;
- While setting the condition to follow for the data retrieved use **WHERE** clause in syntax as followed: **SELECT** column1, column2…… **FROM** table_name **WHERE** condition;

**Application of learning 3.4.**

You are working for a Neighboring secondary school that uses a School Management System (SMS) to handle its day-to-day operations. The SMS database is already set up, containing multiple tables to manage students, teachers, courses, and enrollments with information.

**Database Structure:** Below is the simplified structure of the SMS database:

**Notes: The given data are sample; the tables contain more information**

**Students table**

| student_id | first_name | last_name | date_of_birth | enrollment_date | grade_level |
|---|---|---|---|---|---|
| 1 | *Emma* | *Watson* | *2008-01-15* | *2024-09-01* | 10 |
| 3 | *John* | *Doe* | *2008-09-15* | *2024-09-01* | 9 |
| 5 | *Smith* | Doe | *2007-09-15* | *2024-09-01* | 10 |
| 12 | *Mark* | *Wilson* | 2009-03-03 | *2024-09-01* | 9 |

**Teachers table**

| teacher_id | first_name | last_name | subject | hire_date |
|---|---|---|---|---|
| 1 | *Dr. Sarah* | *Smith* | Physics | *2024-08-20* |
| 5 | *Mrs. Jane* | *Roberts* | History | *2024-08-20* |
| 3 | Sarah | bovia | Mathematics | *2024-08-20* |

**Courses table**

| course_id | course_name | teacher_id |
|---|---|---|
| 1 | Biology | 3 |
| 7 | *Art History* | 1 |
| 4 | *Mathematics* | 3 |
| 2 | *Physics* | 1 |
| 3 | *history* | 5 |

**Enrolments**

| enrollment_id | student_id | course_id | enrollment_date |
|---|---|---|---|
| 1 | 1 | 1 | 2024-09-07 |
| 2 | 3 | 1 | 2024-09-07 |
| 3 | 12 | 1 | 2024-09-07 |
| 4 | 3 | 4 | 2024-09-08 |
| 5 | 1 | 4 | 2024-09-08 |

As a database developer your task is to generate differents reports depending on the manager needs:

1. Generate a report that shows the information of all students
2. Display students who born in 2008
3. Generate a report that shows first name and last name of all students who enrolled in Biology
4. Display the maximum grade_level from students table
5. Display informations of student sort them ascending based on students date_of _birth

6. Display a number of students who enrolled in Mathemetics whose grade_leve is greater than or equal to 10

## Indicative content 3.5: Applying DCL Commands

**Duration: 2 hrs**

### Theoretical Activity 3.5.1: Description of DCL Commands

**Tasks:**

1: You are requested to answer the following question:

I. Describe SQL Grant Command

II. Describe SQL Revoke Command

2: Present the findings to the whole class

3: Participate in grouping and agreeing on correct answers

4: Ask questions where necessary.

5: For more clarification, read the key readings 2.5.1.

### Key readings 3.5.1.: Description of DCL Commands

- **DCL commands**

These are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owners of the database object can provide/remove privileges on a database object.

✓ **SQL GRANT Command**

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

**General Syntax for the GRANT command is:**

```
GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];
```

- **privilege_name** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- **object_name** is the name of a database object like TABLE, VIEW, STORED PROC and SEQUENCE.
- **user_name** is the name of the user to whom an access right is being granted.
- **user_name** is the name of the user to whom an access right is being granted.
- **PUBLIC** is used to grant access rights to all users.

- **ROLES** are a set of privileges grouped together.
- **WITH GRANT OPTION** - allows a user to grant access rights to other users.

**Example:** GRANT SELECT ON employee TO user1; This command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

✓ **SQL REVOKE Command**

The REVOKE command removes user access rights or privileges to the database objects.

**General syntax:**

REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}

**Example:** REVOKE SELECT ON employee FROM user1;

This command will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

✓ **Privileges and Roles:**

- ***Privileges*:** Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

○ System privileges - This allows the user to CREATE, ALTER, or DROP database objects.
○ Object privileges - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

| System Privileges | Description |
|---|---|
| **CREATE object** | allows users to create the specified object in their own schema. |
| **CREATE ANY object** | allows users to create the specified object in any schema. |
| **ALTER object** | allows users to alter the specified object in their own schema. |
| **ALTER ANY object** | allows users to alter the specified object in any schema. |
| **DROP object** | allows users to drop the specified object in their own schema. |

| | |
|---|---|
| **DROP ANY object** | allows users to drop the specified object in any schema. |

Few of the object privileges are listed below:

| Object Privileges | Description |
|---|---|
| **INSERT** | allows users to insert rows into a table. |
| **SELECT** | allows users to select data from a database object. |
| **UPDATE** | allows user to update data in a table. |
| **EXECUTE** | allows user to execute a stored procedure or a function. |

- **Roles:** Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle.

Some of the privileges granted to the system roles are as given below:

| System Role | Privileges Granted to the Role |
|---|---|
| **CONNECT** | CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc. |
| **RESOURCE** | CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects. |
| **DBA** | ALL SYSTEM PRIVILEGES |

- Creating Roles:

**General syntax:**

CREATE ROLE role_name [IDENTIFIED BY password];

**Example:** To create a role called "developer" with password as "pwd", the code will be as follows

CREATE ROLE testing [IDENTIFIED BY pwd];

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password.

- GRANT or REVOKE privilege to a role as below.
-

**Example:** To grant CREATE TABLE privilege to a user by creating a testing role:
First, create a testing Role

- CREATE ROLE testing;

Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.

- GRANT CREATE TABLE TO testing;

Third, grant the role to a user.

- GRANT testing TO user1;

- Revoke a CREATE TABLE privilege from testing ROLE, you can write:
  - REVOKE CREATE TABLE FROM testing;

**General Syntax**: DROP ROLE role_name;
**Example:** To drop a role called developer, you can write:

- DROP ROLE testing;

**Practical Activity 3.5.2: Applying DCL Commands**


**Task:**

1: You are requested to read the task described below:

With the provided users and their privileges in the database you are requested to apply DCL commands by:

1. Creating the provided users in the database
2. Granting privileges to the users as provided
3. Revoking access right to the users that are not currently active

2: Referring to the key reading, perform task 2

3: Present your work to your trainer and a whole class

**Key readings 3.5.2: Applying DCL Commands**

- **DCL commands**

✓ To grant privileges to a user in **MySQL**, follow these steps:

- **Login to MySQL as an Administrator:**

First, log in to MySQL as the root user or any user with sufficient privileges to grant permissions.

Use: mysql -u root -p

- **Create a User (Optional):**

If the user doesn't exist yet, you need to create the user first.

**Use: CREATE USER 'username'@'host' IDENTIFIED BY 'password';**

- **username**: The name of the user you want to create.
- **host**: The host from which the user will connect (use '%' for any host or 'localhost' for local).
- **password**: The user's password.

- **Grant Privileges:**

Use the GRANT statement to give the user privileges on specific databases or tables.

**General Syntax: GRANT privileges ON database.table TO 'username'@'host';**

- **privileges: List of privileges you want to grant (e.g., SELECT, INSERT, UPDATE, ALL).**
- **database.table: The specific database and table (e.g., mydb.* grants privileges on all tables in mydb, or mydb.mytable for one table).**
- **username@host: The user to whom you are granting privileges.**

**Example: To grant all privileges on the entire mydb database to username:**

GRANT ALL PRIVILEGES ON mydb.* TO 'username'@'localhost';
To grant SELECT and INSERT privileges on the employees table of the company database:

**GRANT SELECT, INSERT ON company.employees TO 'username'@'localhost';**

**Or**

I. **Create a Role**

**Use the CREATE ROLE command to create a new role.**

**CREATE ROLE role_name;**

### II. Grant Privileges to the Role

After creating the role, you can assign specific privileges using the GRANT command. These privileges can be related to DML, DDL, or other operations.

GRANT privilege_type ON object TO role_name;

- privilege_type: Specifies the type of privilege (e.g., SELECT, INSERT, UPDATE, DELETE, ALL PRIVILEGES).
- object: The database object (e.g., table_name, schema).
- role_name: The role that is being granted the privilege.

GRANT SELECT, INSERT ON employees TO hr_manager;

### III. Assign Role to a User

To enable a user to use the privileges assigned to the role, assign the role to the user with the GRANT command.

**GRANT role_name TO user_name;**

**Example: GRANT hr_manager TO john;**

- **Apply Changes:**

Run the following command to reload the privilege tables and apply changes.
FLUSH PRIVILEGES;

- **Verify User Privileges (Optional):**

To check the privileges granted to the user: SHOW GRANTS FOR 'username'@'host';

- **Grant with Grant Option (Optional):**

**To allow the user to grant the same privileges to others, use the WITH GRANT OPTION clause.**

**GRANT SELECT, INSERT ON company.employees TO 'username'@'localhost' WITH GRANT OPTION;**

- **Revoke Privileges (If Necessary):**

If you need to revoke privileges later, use the REVOKE command:

REVOKE privilege ON database.table FROM 'username'@'host';

**For example**, to revoke INSERT privilege from the user on the employees table:

REVOKE INSERT ON company.employees FROM 'username'@'localhost';

**Points to Remember**

- To grant access to the user of the system use GRANT command.
- To remove access to the user of the system use REVOKE command.
- General Syntax for the GRANT command is: GRANT privilege_name
  ON object_name
  TO {user_name |PUBLIC |role_name}
  [WITH GRANT OPTION];
- **General syntax: RE**VOKE privilege_name
  ON object_name
  FROM {user_name |PUBLIC |role_name}

**Application of learning 3.5.**

STARS secondary school located at Muhanga District, Bweramana Sector, Buramba Village uses a School Management System (SMS) to handle its day-to-day operations. The SMS database is already set up, containing multiple tables to manage students, teachers, courses, and enrollments with information. The database is not secured everyone can access information in database, this may cause the data loss and falsified data. As technician you are asked to perform the bellow activities in order to avoid that issues.

- ✓ Students record his/her information in students table and enroll in the courses
- ✓ Teacher views the courses he/she has to teach and a list of students enrolled in that courses. She/he can also add a new teacher in the database.
- ✓ Admin adding a new teacher from a teacher user

## Indicative content 3.6: Applying TCL Commands

**Duration:3 hrs**

### Theoretical Activity 3.6.1: Description of TCL Commands

**Tasks:**

1: You are requested to answer the following question:

I. Describe TCL COMMIT Command
II. Describe Save Point Command
III. Describe TCL Roll Back Command
IV. Describe TCL Set Transaction Command
V. Describe TCL set constraint Command

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 3.6.1.

**Key readings 3.6.1.: Description of TCL Commands**

- **TCL commands**

Transaction Control Language (TCL) commands are used to manage transactions in a database. Transactions are a sequence of one or more SQL operations that are executed as a single unit of work, ensuring data integrity.

✓ **COMMIT**

- The COMMIT command is used to **save all the transactions to the database** that have been performed during the current transaction.
- Once a transaction is committed, it becomes permanent and cannot be undone.
- This command is typically used at the end of a series of SQL statements to ensure that all changes made during the transaction are saved.

**Syntax:**

COMMIT;

Example:

INSERT INTO students (name, grade) VALUES ('Mukantwari', 'A');
COMMIT;

✓ **SAVEPOINT**

- The SAVEPOINT command is used to **set a point within a transaction to which we can later roll back.**
- This command allows for partial rollbacks within a transaction, providing more control over which parts of a transaction to undo.

**Syntax**:

```
SAVEPOINT savepoint_name;
```

Example:

```
BEGIN;

INSERT INTO orders (order_id, amount) VALUES (101, 500);
SAVEPOINT sp1;

UPDATE orders SET amount = 600 WHERE order_id = 101;
SAVEPOINT sp2;

DELETE FROM orders WHERE order_id = 102;
SAVEPOINT sp3;

COMMIT;
```

✓ **ROLLBACK**

- The ROLLBACK command is used to **undo all the transactions that have been performed during the current transaction but have not yet been committed.**
- This command is useful for reverting the database to its previous state in case an error occurs or if the changes made are not desired.

**Syntax:**

```
ROLLBACK;
```

Example:

```
BEGIN;
INSERT INTO employees (id, name) VALUES (1, 'Alice');
SAVEPOINT sp1;
INSERT INTO employees (id, name) VALUES (2, 'Bob');
ROLLBACK TO SAVEPOINT sp1;
COMMIT;
```

✓ **Uses of TCL Commands**

- **COMMIT**: Used after data modifications (INSERT, UPDATE, and DELETE) to save changes to the database.
- **ROLLBACK**: Used to revert changes if something goes wrong, ensuring data integrity.

- **SAVEPOINT**: Used to create intermediate points within a transaction to which you can roll back, providing finer control over transaction management.
- **SET TRANSACTION**: Used to configure transaction behavior, ensuring proper isolation and consistency as per requirements.

✓ **SET TRANSACTION**

The SET TRANSACTION command is used to establish characteristics for the current transaction in SQL. It allows you to control the transaction's behavior, setting its isolation level, read/write access, and more.

- **Purpose**: To define the transaction's isolation level and other characteristics before executing a transaction.
- **Syntax**:

SET TRANSACTION [READ WRITE | READ ONLY] [ISOLATION LEVEL {SERIALIZABLE | READ COMMITTED | READ UNCOMMITTED | REPEATABLE READ}];

Example:

```
SET TRANSACTION READ WRITE ISOLATION LEVEL SERIALIZABLE;

BEGIN;

INSERT INTO accounts (account_id, balance) VALUES (101, 1000);

COMMIT;
```

**Key Parameters**:

- **READ WRITE / READ ONLY**: Specifies whether the transaction can modify the database. READ WRITE allows changes, while READ ONLY restricts changes to prevent unintended updates.
- **ISOLATION LEVEL**: Sets how the transaction is isolated from other concurrent transactions. Isolation levels include:
  - **READ UNCOMMITTED**: Lowest isolation level; allows dirty reads (reading uncommitted changes from other transactions).
  - **READ COMMITTED**: Default level; prevents dirty reads but allows non-repeatable reads (data can change if another transaction modifies it).
  - **REPEATABLE READ**: Prevents dirty and non-repeatable reads but allows phantom reads (new rows added by other transactions).
  - **SERIALIZABLE**: Highest isolation level; ensures complete isolation, preventing dirty, non-repeatable, and phantom reads.

✓ **SET CONSTRAINTS**

The SET CONSTRAINTS command is used to manage the enforcement of constraints (like foreign keys) during a transaction. This command allows temporarily deferring constraint

checks until the end of the transaction, which is particularly useful when performing bulk updates or inserts that may initially violate constraints.

- **Purpose**: To set the constraint checking mode during the execution of a transaction.

**Syntax**: SET CONSTRAINTS {ALL | constraint_name} {DEFERRED | IMMEDIATE};
**Example:** SET CONSTRAINTS ALL DEFERRED;

```
BEGIN;
UPDATE inventory SET stock = stock - 10 WHERE product_id = 101;
SET CONSTRAINTS ALL IMMEDIATE;
COMMIT;
```

- **Key Parameters**:
  - **ALL | constraint_name**: Specifies which constraints to affect. ALL targets all constraints, or you can specify individual constraint names.
  - **DEFERRED**: Delays the constraint checks until the transaction is committed. This setting is beneficial when multiple operations could violate constraints but would be valid at the end of the transaction.
  - **IMMEDIATE**: Enforces the constraints immediately after each operation within the transaction. This is the default behavior.

**Practical Activity 3.6.2: Applying TCL Commands**


**Task:**

1: Read the given task

2: Follow the demonstration of trainer to perform the following task:

You manage a retail database with Employees, Orders, and Customers tables. Perform the following tasks using TCL commands to control the transaction flow:

a) **Insert Records and Commit:**

i. Insert records into Employees with:
(EmpID: 1, Name: 'John Doe', Department: 'HR')
(EmpID: 2, Name: 'Jane Smith', Department: 'Finance')

ii. Use COMMIT to save the changes.

b) **Create Savepoint and Insert:**

i. Insert (OrderID: 101, ProductName: 'Laptop', Quantity: 2) into Orders.

ii. Set a SAVEPOINT named savepoint1.

iii. Insert another record: (OrderID: 102, ProductName: 'Phone', Quantity: 5).

c) **Rollback to Savepoint:**

i. Realize the second order insert is incorrect; ROLLBACK TO savepoint1 to undo it.

d) **Update and Full Rollback:**

i. Update Employees to set Department = 'Marketing' for EmpID = 1.

ii. Delete the employee with EmpID = 2.

iii. Use ROLLBACK to undo both changes.

e) **Set Transaction and Query:**

i. Set the transaction to READ ONLY.

ii. Retrieve all employees from the HR department.

f) **Defer Constraints:**

i. Set constraints to DEFERRED.

ii. Insert (OrderID: 103, CustomerID: 201) into Orders.

iii. Update Customers to change Customer Name for CustomerID = 201 to 'Michael Johnson'.

iv. Enforce constraints immediately after the changes.

3: Present your work to your trainer and a whole class

**Key readings 3.6.2: Applying TCL Commands**

- **TCL Commands**
  - ✓ **COMMIT**
    - **Steps Followed:**
  - After performing a series of DML (Data Manipulation Language) operations such as INSERT, UPDATE, or DELETE, use COMMIT to save all changes to the database.
  - Ensure that the operations are correct and the data is in a valid state before executing COMMIT.
  - Execute the COMMIT command to make the changes permanent.

**Key Notes:**

- COMMIT cannot be undone. Once executed, the changes are saved to the database permanently.
- It releases the locks held during the transaction, making the changes visible to other users.
- It is good practice to COMMIT frequently to avoid data loss in case of a system failure.

**Considerations:**

- Always validate data before committing, as any mistake will require additional operations to correct.
- Use COMMIT at logical points in your application or script where the data is consistent and complete.

✓ **SAVEPOINT**

**Steps Followed:**

- Use SAVEPOINT to create a point within a transaction to which you can later roll back.
- After executing several DML operations, use SAVEPOINT to mark a specific point.
- Continue with additional operations. If an error occurs, you can use ROLLBACK TO SAVEPOINT to undo only the operations after the savepoint.

**Key Notes:**

- SAVEPOINT helps in partial rollbacks, allowing a subset of a transaction to be undone.
- Multiple savepoints can be created within a single transaction, each with a unique name.

**Considerations:**

- SAVEPOINT is useful in complex transactions where some operations might be valid while others might need to be reversed.
- Efficient use of savepoints can help in optimizing error handling in transaction management.

✓ **ROLLBACK**

**Steps Followed:**

- After a series of operations, use ROLLBACK to undo changes made within the current transaction.
- If you want to undo changes up to a specific point, use ROLLBACK TO SAVEPOINT.
- When ROLLBACK is executed without a save point, all changes since the last COMMIT are undone.

**Key Notes:**

- ROLLBACK restores the database to its last committed state.
- ROLLBACK TO SAVEPOINT undoes changes made after a specific savepoint, while retaining the changes made before it.

**Considerations:**

- ROLLBACK should be used in cases of errors or invalid data to maintain data integrity.
- Use ROLLBACK cautiously as it can result in the loss of all changes made in the transaction.

✓ **SET TRANSACTION**

**Steps Followed:**

- Before starting a transaction, use SET TRANSACTION to define properties like the isolation level and read/write access.
- Set the transaction properties by executing a SET TRANSACTION command, for example, SET TRANSACTION READ ONLY;
- Perform the necessary DML operations as required within the defined transaction.

**Key Notes:**

- The isolation level determines how changes made by one transaction are visible to others, ensuring data consistency and preventing issues like dirty reads or phantom reads.
- SET TRANSACTION can be used to specify whether a transaction should be read-only or read-write.

**Considerations:**

- Carefully choose the isolation level based on the requirements of data consistency and performance.
- For read-only operations, using READ ONLY can optimize performance and reduce locking overhead.

✓ **SET CONSTRAINTS**

**Steps Followed:**

- Use SET CONSTRAINTS to temporarily defer or immediately enforce constraints like FOREIGN KEY or UNIQUE.
- Execute SET CONSTRAINTS ALL DEFERRED to delay the constraint check until the end of the transaction.
- Execute SET CONSTRAINTS ALL IMMEDIATE to enforce constraints immediately.

**Key Notes:**

- Deferred constraints are useful when the order of data modifications may temporarily violate a constraint but will be resolved by the end of the transaction.
- It provides flexibility in managing the order of operations in complex transactions.

**Considerations:**

- Use deferred constraints judiciously to avoid accidental data integrity issues.

Always ensure that all constraints are satisfied before committing the transaction.

**Points to Remember**

- The general syntax to Permanently save all the changes made during the current transaction in the database. Use **COMMIT;**
- to set a point within a transaction to which you can later roll back. Use **SAVEPOINT** savepoint_name;
- to undo changes made during the current transaction, either fully or up to a SAVEPOINT. Use **ROLLBACK;**
- Rollback to a Specific Savepoint: **ROLLBACK TO SAVEPOINT** savepoint_name;
- to specify the characteristics of the transaction, such as isolation level or read/write access mode. Use **SET** TRANSACTION [READ WRITE | READ **ONLY**] [ISOLATION LEVEL {READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}];
  Examples: **SET** TRANSACTION READ **ONLY** ISOLATION LEVEL READ COMMITTED;
- to set the behavior of constraint checking during a transaction. It can be used to defer or enforce constraints. Use **SET** CONSTRAINTS constraint_name [, ...] {IMMEDIATE | DEFERRED};

**Application of learning 3.6.**

You are tasked with managing a database for a school. The database contains information about students, courses, and grades. You will need to use SQL commands to create the database structure, insert data, update records, query information, and control transactions. **Tasks:**

1. **Create the Database Structure:**
   i. Create a database named "SchoolDB".
   ii. Create a table named "Students" with columns: StudentID, FirstName, LastName, and GradeLevel.
   iii. Create a table named "Courses" with columns: CourseID, CourseName, and Department.
   iv. Create a table named "Grades" with columns: GradeID, StudentID, CourseID, Grade.
2. **Insert Data:**
   i. Insert 5 student records into the "Students" table.
   ii. Insert 3 course records into the "Courses" table.

iii. Insert 10 grade records into the "Grades" table, linking students to courses and assigning grades.

3. **Query Data:**
    i. Find the average grade for a specific course.
    ii. List all students in a particular grade level.
    iii. Retrieve the names of students who have a grade higher than 90 in a specific course.
4. **Transaction Management:**
    i. Begin a transaction to update multiple records related to a student's information.
    ii. Create a savepoint before updating the student's grade.
    iii. If an error occurs during the update, roll back the transaction to the savepoint.
    iv. If the update is successful, commit the transaction.
5. **Query Data:**
    i. Find the average grade for a specific course.
    ii. List all students in a particular grade level.
    iii. Retrieve the names of students who have a grade higher than 90 in a specific course.
6. **Transaction Management:**
    i. Begin a transaction to update multiple records related to a student's information.
    ii. Create a savepoint before updating the student's grade.
    iii. If an error occurs during the update, roll back the transaction to the savepoint.
    iv. If the update is successful, commit the transaction.
7. **Data Security:**
    i. Grant privileges to a specific user to view and modify student data.
    ii. Revoke privileges from a user who no longer needs access.

## Learning outcome 3 end assessment

**Theoretical assessment**

1) Match the terms in COLUMN A related to their description in COLUMN B

| Answer | COLUMN A | COLUMN B |
|---|---|---|
| ….. | 1) SQL Arithmetic Operators | Used to update an existing table structure, such as adding or modifying columns. |
| ….. | 2) ALTER Table | B. Command to permanently save all changes made during the current transaction. |
| ….. | 3) INSERT | C. Allows you to add new rows to a table. |
| ….. | 4) COMMIT | D. Operator used for bit-level operations, e.g.: &. |
| ….. | 5) SQL Aggregate Function | E. Command used to control access privileges to database objects. |
| ….. | 6) GRANT | F. Keyword used to retrieve data from a database, often with conditions and sorting. |
| ….. | 7) EXPLAIN CALL | G. Command to set a point in a transaction that you can roll back to. |
| ….. | 8) SELECT | H. Function used to perform calculations on a set of values, such as SUM or AVG. |
| ….. | 9) SAVEPOINT | I. Used to describe the details of how a stored procedure is called. |
| ….. | 10) BITWISE AND Operator | J. Performs mathematical operations, like addition or subtraction, on numeric data. |

**2) Read the following statements and fill in the blank using the words given in parentheses (put all choices here )**

I. The _____ operator is used in SQL for adding two numeric values together.

II. The _____ command in SQL is used to create a new table in the database.

III. To revoke previously granted privileges in SQL, the _____ command is used.

IV. SQL's _____ command is used to insert new records into a table.

V. The _____ SQL command is used to temporarily set a point to which a transaction can be rolled back.

3) Circle the letter with the correct answers:

I. **What is the correct syntax for creating a new database named "mydb"?**
   - CREATE DATABASE mydb;
   - CREATE DATABASE mydb();
   - DATABASE CREATE mydb; d)
   - NEW DATABASE mydb;

II. **Which clause is used to specify conditions for selecting rows in a SELECT statement?**
   a) WHERE
   b) FROM
   c) GROUP BY
   d) HAVING

**III. What is the correct syntax for inserting a new row into a table named "Customers"?**

A. INSERT INTO Customers (CustomerID, CustomerName) VALUES (1, 'John Doe')

B. INSERT Customers (CustomerID, CustomerName) = (1, 'John Doe')

C. ADD ROW Customers (CustomerID, CustomerName) = (1, 'John Doe')

D. CREATE ROW Customers (CustomerID, CustomerName) VALUES (1, 'John Doe')

IV. **Which SQL aggregate function calculates the average value of a column?**

A. SUM()

B. AVG()

C. COUNT()

D. MAX ()

4) What command is used to delete a database?
5) What SQL clause is used to group rows based on a specified column?

**Practical assessment**

ABJ is a secondary school that is located in your sector, the school has a large number of students within different options; for that reason, the school needs a comprehensive database to manage students, teachers, courses, and grades because the manual system it uses returns errors in the termly reports and annual reports.

You are asked to:

1. Create the necessary tables: Students, Teachers, Courses, and Grades, ensuring the correct relationships, are in place.

2. Create the tables, populate them with sample data using DML commands, including inserting records for students, teachers, and courses, as well as updating and deleting specific entries as needed.
3. Apply DQL commands to retrieve information, such as the list of students enrolled in a specific course or the grades of a particular student in term and at the end of the year.
4. Manage permissions to database by ensuring that only authorized users can view or modify certain data.
5. Employ TCL commands to handle transactions, ensuring that operations like grade updates are completed successfully and, if any issues arise, the system can roll back to maintain data integrity.

**END**

## References


C.J.Date, (2009). SQL and Relational Theory: How to Write Accurate SQL. Boston, Massachusetts, USA: O'Reilly Media.

Hector Garcia-Molina, J. U. (2008). Database Systems: The Complete Book. Upper Saddle River, New Jersey, USA: Pearson Prentice Hall.

https://beginner-sql-tutorial.com/sql-grant-revoke-privileges-roles.htm

https://docs.getdbt.com/terms/dml

https://www.dbvis.com/thetable/sql-ddl-the-definitive-guide-on-data-definition-language/

https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/

Raghu Ramakrishnan, J. G. (2003). Database Management Systems. Boston, Massachusetts, USA: McGraw-Hill.

| Indicative contents |
|---|
| **4.1 Enforcement of data access control**<br>**4.2 Management of Auditing and logging**<br>**4.3 Implementation of Data encryption**<br>**4.4. Configuration of database backup and restore** |

## Key Competencies for Learning Outcome 4 : Implement Database security

| Knowledge | Skills | Attitudes |
|---|---|---|
| • Description of database security<br>• Description of data control access<br>• Description of auditing and logging<br>• Description of data encryption<br>• Introduction of data backup and restore | • Implementing data access control<br>• Managing Auditing and logging<br>• Implementing Data encryption<br>• Configuring database backup and restore | • Having accountability skills while enforcing data access control and managing auditing and logging<br>• Having adaptability skills while implementing data encryption<br>• Being Patient while Configuring database backup and restore |

**Duration: 19hrs**

**Learning outcome 4 objectives**:

By the end of the learning outcome, the trainees will be able to:

1. Describe properly database security based on database security measures
2. Describe properly data access control based on database security measures GGGV
3. Enforce properly access control based on database security measures
4. Manage clearly auditing and logging based on the security policies
5. Implement correctly data encryption based on data security measures
6. Configure regularly Backup and Recovery of data based on DBMS

**Resources**

| Equipment | Tools | Materials |
|---|---|---|
| • Computer | • Ms SQL Server<br>• Browsers | • Internet |

## Indicative content 4.1: Enforcement of Data Access Control

**Duration: 4 hrs**

### Theoretical Activity 4.1.1: Description of database security

**Tasks:**

1: You are asked to answer the following questions:

I. What do you understand by database security

II. Explain types of database security

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 4.1.1.

**Key readings 4.1.1: Description of database security**

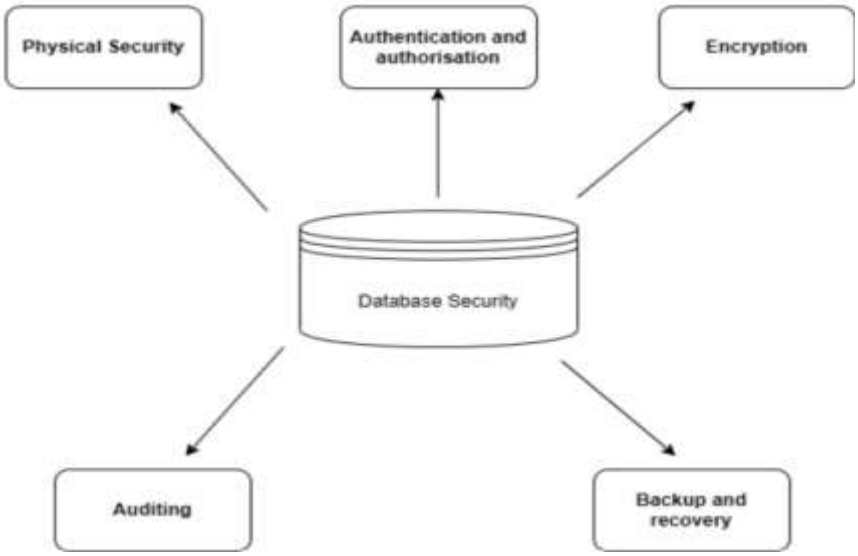
- **Database security**
  - ✓ **Introduction to database security**

Database security is the processes, tools, and controls that secure and protect databases against accidental and intentional threats. The objective of database security is to secure sensitive data and maintain the confidentiality, availability, and integrity of the database.

In addition to protecting the data within the database, database security protects the database management system and associated applications, systems, physical and virtual servers, and network infrastructure.

To answer the question "what is database security," it's important to acknowledge that there are several types of security risks. Database security must guard against human error, excessive employee database privileges, hacker and insider attacks, malware, backup storage media exposure, physical damage to database servers, and vulnerable databases such as unpatched databases or those with too much data in buffers.

  - ✓ **Types of database security**




- **Physical Security**

The safeguarding of the actual infrastructure that contains the database is referred to as physical security. This covers the server room's physical security as well as the security of the network and storage systems. Only authorized workers should be able to enter the server room, and CCTV cameras should be placed to keep an eye on the space. Utilizing firewalls, intrusion detection systems, and other security measures, the network infrastructure should be protected. Encryption should be used on storage devices to prevent theft and unauthorized access.

- **Authentication and authorization**

Two crucial elements of database security are **authentication and authorization**. **Authorization** is the process of giving access to certain resources depending on the user's role and rights, whereas **authentication** is the process of confirming a user's identity. Strong passwords and regular password changes should be mandated by password regulations. To add another level of protection, two-factor authentication ought to be implemented. To guarantee that users can only access data that they are authorized to access, authorization regulations should be strictly adhered to.

- **Encryption**

Data encryption is the process of encoding data so that it can only be read with a special key. Both data in transit (being transferred over a network) and data at rest (being kept on a disc) can be protected via encryption. Data at rest can be encrypted using disc encryption, while data in transit can be encrypted using SSL/TLS. Sensitive data in a database can also be encrypted using column-level encryption.

- **Auditing**

Monitoring and logging database activities is referred to as auditing. Unauthorized access, data breaches, and other security problems may all be found and avoided with the use of auditing. Auditing should be set up to record pertinent data, including user

activity, failed login attempts, and alterations to the database structure. The logs should be kept in a secure location and routinely checked for any unusual activities.

- **Backup and recovery**

Essential elements of database security, backup and recovery. To guarantee that data can be restored in the case of a disaster or cyberattack, regular backups should be made. Backups have to be encrypted and kept in a safe place. Regular testing of recovery processes is necessary to guarantee efficient and speedy data recovery.

**Theoretical Activity 4.1.2: Description of data access control**


**Tasks:**

1: You are asked to answer the following questions related to data access control:

I. What is data access control?
II. Identify data classifications
III. Differentiate role from permission
IV. Differentiate authentication from authorisation

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 4.1.2.

**Key readings 4.1.2: Description of data access control**

- **Data access control**
  - ✓ **Definition**

Database access control, or DB access control, is a method of allowing access to a company's sensitive information only to user groups who are allowed to access such data and restricting access to unauthorized persons to prevent data breaches in database systems.

Database Access Control in DBMS includes two main components: authentication and authorization.

- ✓ **Access control policies**

What is an access control policy? As the name suggests, these are sets of policies, instructions, and restrictions that are in place which specify who can access your data,

when they can do so, and up to which level. These policies need to be implemented accordingly at all levels of the organization.

an access control policy secures sensitive data and minimizes the risk of an attack. access control policies function by authenticating user credentials, proving their identity, and allowing the pre-approved permissions associated with their username and ip address.

### Types of Access Control Policies

- **Mandatory Access Control**

Access controls which are based on the rules and regulations set up by the authority. In other words, the access remains only with the owner and overseers.

- **Discretionary Access Control**

Access control policies which are decided by the data owner. The business owner themselves decides the number of people and level of access to data.

- **Role-Based Access Control**

Access controls which are based on the roles of individuals. For instance, people in higher authority roles may have more privileges compared to the lower-level management. This distinction makes business processes consistent and straightforward for individuals.

- **Rule-Based Access Control**

Not to be confused with Role-Based Access Control, this access control is an addition to other policies where certain rules are defined based on business processes and infrastructure for the access control standards.

### Identify the data classifications

Data classification is the process of organizing data into categories based on its sensitivity, importance, or confidentiality. Here are some common data classifications:

- **Public Data**:
  - Information that is freely available to the public.
  - No risk if disclosed.
  - Example: Marketing materials, public reports, press releases.
- **Internal Data:**
  - Information meant for internal use within an organization.
  - Limited risk if disclosed to outsiders.
  - Example: Internal memos, internal policy documents, employee directories.
- **Confidential Data**:
  - Sensitive information that should only be accessed by authorized individuals.
  - High risk if disclosed or modified.

- Example: Customer data, financial records, business strategies.
- **Restricted Data (Highly Confidential):**
- Extremely sensitive data with the highest security requirements.
- Critical risk if disclosed, leading to severe legal, financial, or reputational damage.
- Example: Trade secrets, government classified documents, intellectual property.

The exact classifications may vary between organizations, but these levels help determine the appropriate security measures and access controls.

### Define roles and permission

In SQL Server, roles and permissions are used to control what actions users or groups of users can perform on database objects such as tables, views, stored procedures, etc. SQL Server provides both **fixed roles** (predefined by SQL Server) and **user-defined roles** (custom roles created by the administrator), each with specific permissions.

- **Roles in SQL Server Database:**

A role is a collection of permissions that define what actions a user can perform. Instead of assigning permissions directly to individual users, roles are used to group permissions and assign them collectively.

**Server-Level Roles:**

Server-level roles manage access to the entire SQL Server instance, not just a specific database.

- **sysadmin**: Grants full control over the SQL Server instance, including all databases, configurations, and users.
- **dbcreator**: Allows creating, altering, and dropping databases.
- **securityadmin**: Manages logins and server-level permissions.
- **serveradmin**: Can manage server-wide configurations and settings.
- **diskadmin**: Manages disk files.

**Database-Level Roles:**

Database-level roles control access to a specific database and its objects.

- **db_owner**: Full access and control over the database.
- **db_datareader**: Can read all data from all user tables.
- **db_datawriter**: Can insert, update, and delete data in all user tables.
- **db_ddladmin**: Can run DDL (Data Definition Language) commands, such as CREATE, ALTER, and DROP.
- **db_securityadmin**: Manages database-level security by granting and revoking permissions.
- **db_backupoperator**: Can back up the database.
- **db_accessadmin**: Can manage database access for logins.

- **db_bizuser**: Typically used to define a custom business role with specific privileges.

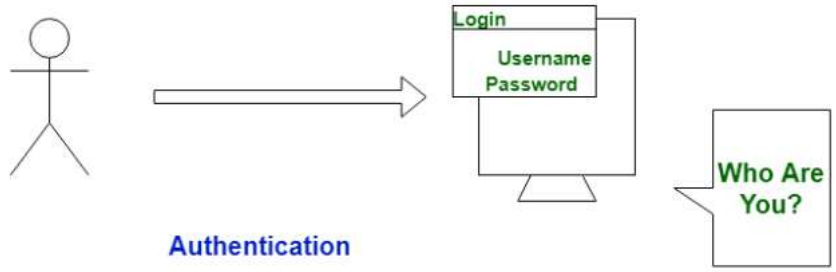
**Permissions in SQL Server Database:**

Permissions define what actions a user or role can perform on a specific object. SQL Server has **server-level** and **database-level** permissions.

- **Server-Level Permissions:**
- **ALTER ANY LOGIN**: Create, alter, or drop logins.
- **CONTROL SERVER**: Full control over the SQL Server instance.
- **VIEW SERVER STATE**: View the state of the server.
- **Database-Level Permissions:**
- **SELECT**: Read data from a table or view.
- **INSERT**: Insert new records into a table.
- **UPDATE**: Modify existing data in a table.
- **DELETE**: Remove records from a table.
- **EXECUTE**: Run a stored procedure or function.
- **ALTER**: Modify an object (e.g., change a table's structure).
- **REFERENCES**: Create foreign key relationships between tables.
- **CONTROL**: Full control over a database object (similar to ownership).

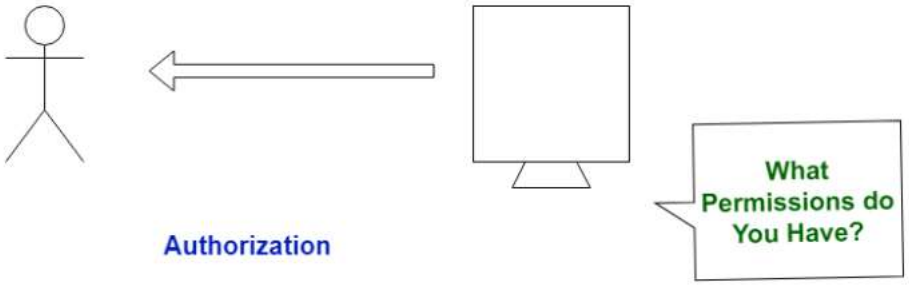
✓ **Authentication**


Authentication is the method of verifying the identity of a consumer or system to ensure they're who they claim to be. It involves checking credentials which include usernames, passwords, or biometric information like fingerprints or facial recognition. This step is vital for securing access to systems, programs, and sensitive records. By confirming identities, authentication saves you from unauthorized entry and protects you against safety breaches.




- ✓ **Authorization**


Authorization is the method of figuring out and granting permissions to a demonstrated user or system, specifying what assets they can access and what actions they're allowed to carry out. It comes after authentication and guarantees that the authenticated entity has the proper rights to use certain data, applications, or services. This step is important for implementing protection guidelines and controlling access within the system, thereby stopping unauthorized activities.




**Practical Activity 4.1.3: Authenticating database by creating database users sSQL Server**


**Task:**

1: Read key **reading 4.1.3**.in trainee manual

2: Read carefully the below task:

D&G University is expanding its student information system (SIS) to support more users as part of a digital transformation project. As the university's database developer, you have been tasked with ensuring robust user management and security for the SQL Server database supporting this system. Your tasks are:

- To **identify the existing user accounts** in the system and audit their current permissions to ensure they align with best practices.
- To **create new logins and users** for additional staff, including the admissions and academic departments, ensuring that each user has appropriate access
- To **configure SQL Server authentication settings and test the authentication system** by logging in with newly created accounts to verify that access control is functioning as expected.

3: Create users accounts, authorise and authenticate them as required in the above task.

4: Present your work to your trainer and a whole class.

**Key readings 4.1.3: Authenticating database in SQL Server**

To authenticate and create a new user in SQL Server using SQL Server Management Studio (SSMS), you need to follow several steps. This involves creating a login at the server level and then creating a user associated with that login at the database level.

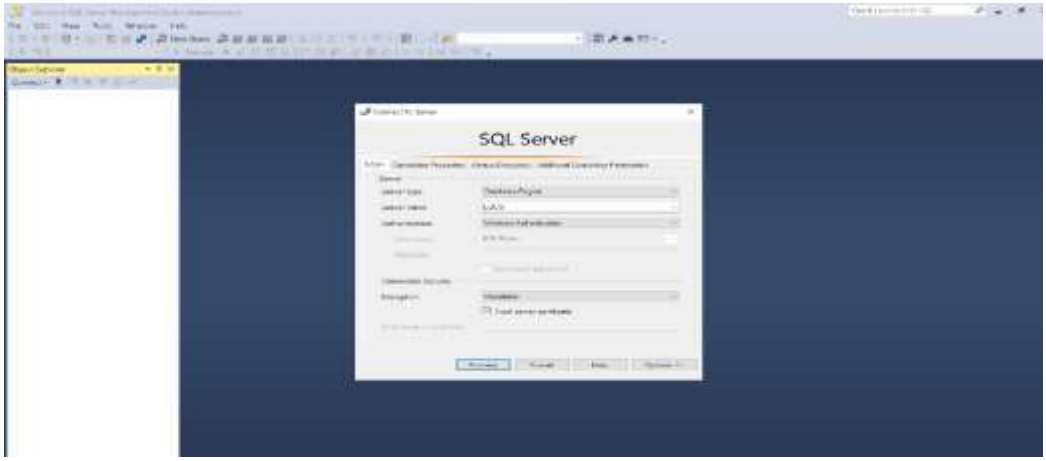
- **Steps to authenticate database by creating database users**
  - ✓ **Authentication in SQL Server**
    - **Types of authentications in SQL Server**
  - **Windows Authentication:** Uses Active Directory (AD) credentials. The user is authenticated by Windows, and SQL Server trusts that authentication.
  - **SQL Server Authentication:** Uses a username and password created within SQL Server. This is independent of Windows credentials.

Choose the authentication mode to connect to SQL Server using SSMS:

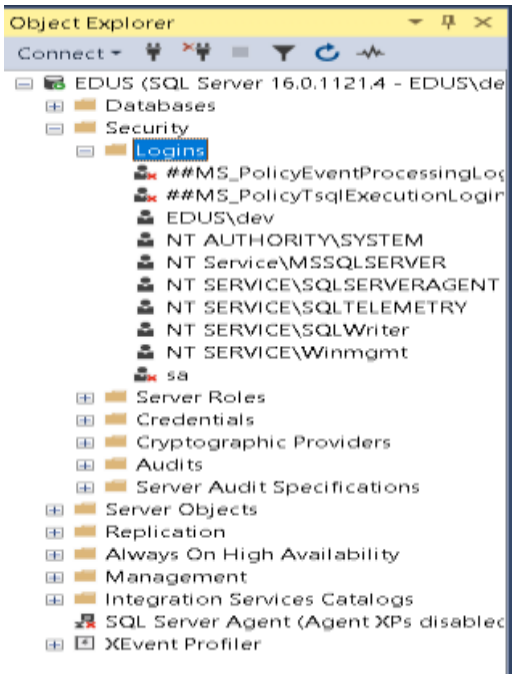
- **Authentication modes**
  - **Windows Authentication**: Choose this if you're using your Windows credentials.
  - **SQL Server Authentication**: Enter the SQL Server username and password.
    - ✓ **Creating a New Login in SQL Server**

A **login** is a security principal at the SQL Server instance level. To create a new login in SSMS:

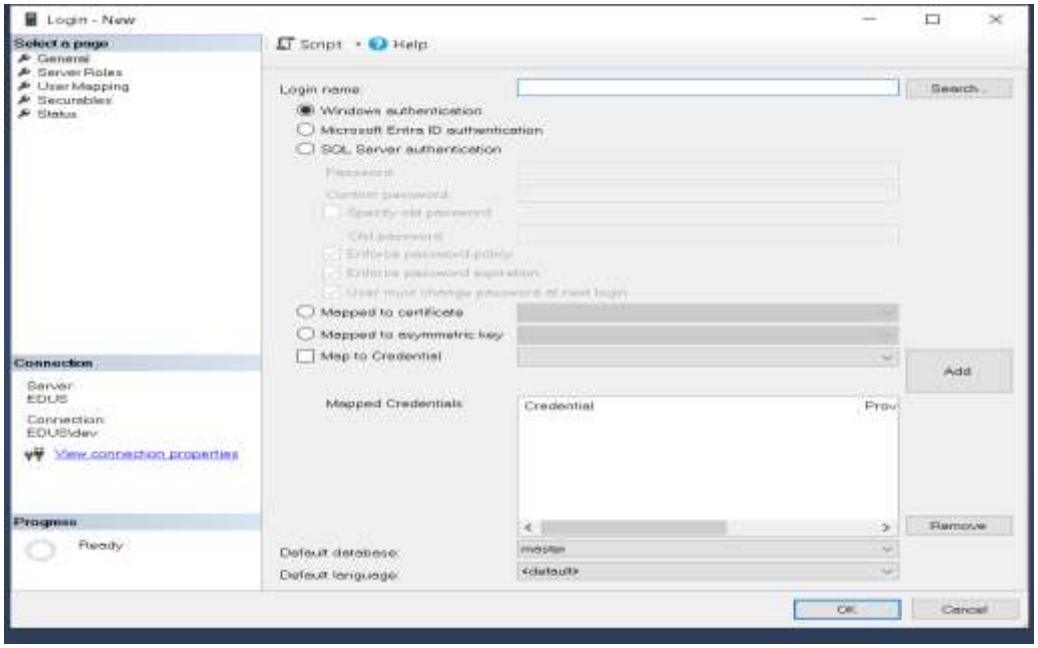
- **Open SQL Server Management Studio (SSMS)** and connect to your SQL Server instance.




- In **Object Explorer**, expand the **Security** folder.




- Right-click on **Logins** and select **New Login...**.




✓ **Creating a Login Using Windows Authentication**

In the **Login - New** window, under the **General** tab:

- Enter the **Login Name** in the format Domain\UserName.

- **Domain**: This is the name of the Windows domain to which the user belongs. A domain is a network that centralizes user management, security, and resources.
- **UserName**: This is the specific user's account name in that domain.

So, the format Domain\UserName is used to uniquely identify a user within a network managed by Active Directory.

Example of "Domain\UserName"

Let's assume your organization has a Windows domain called CORP and a user named Dev. When creating a login in SQL Server using Windows Authentication, the **Login Name** will be CORP\Dev.

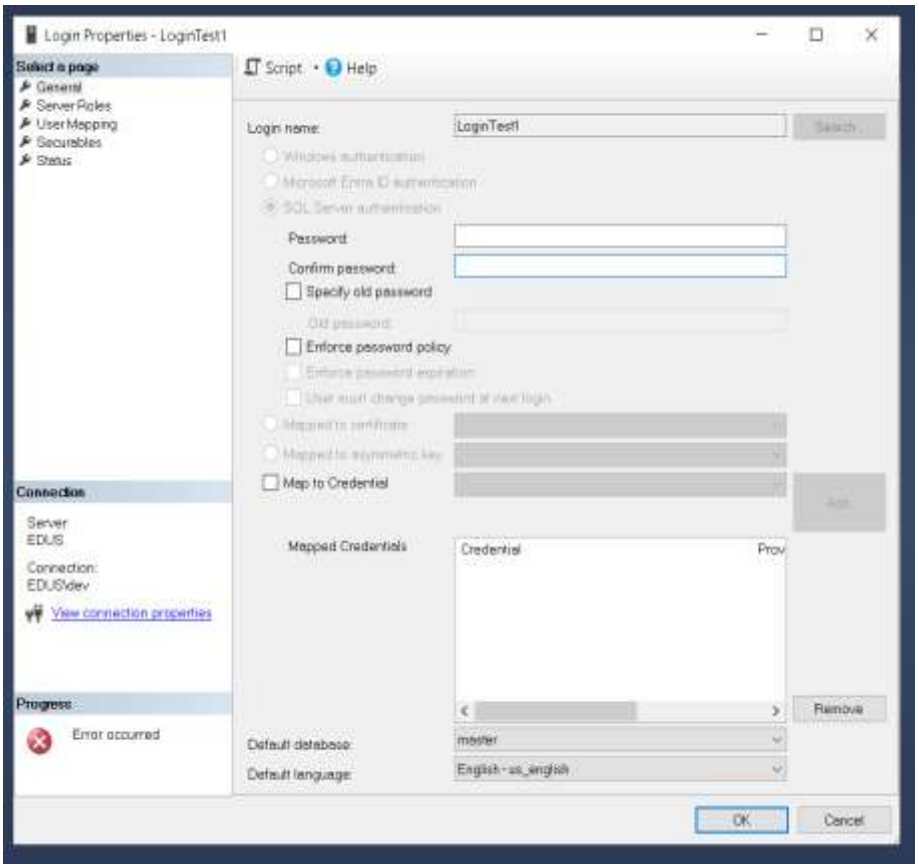
- Select Windows Authentication.
- Configure user mapping(check db owner), if needed, and click **OK**.

**Creating a Login Using SQL Server Authentication**

- In the Login - New window, under the General tab:

  - Enter the **Login Name**.
  - Select **SQL Server Authentication**.
  - Enter a **Password** and **Confirm Password**.

- Uncheck the box that says **Enforce password policy** if you want to set a custom policy.



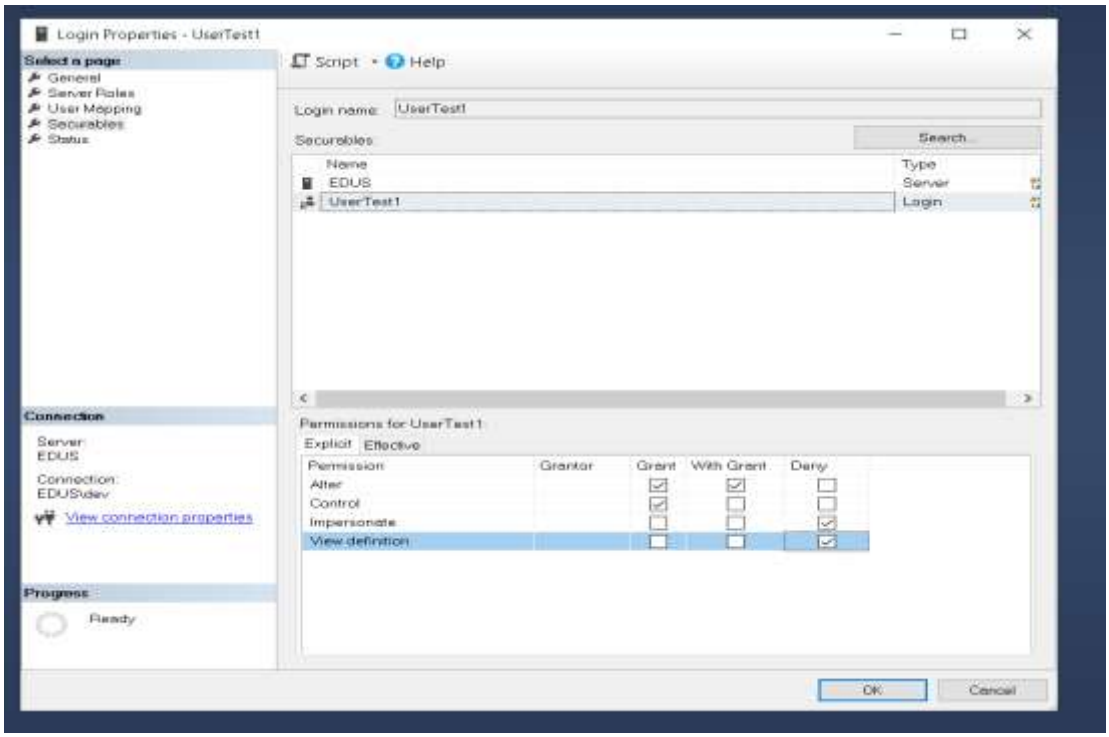

- Configure other settings like **Default Database** (optional), **Server Roles**, and **User Mapping**.
- Click **OK** to create the login.

### Grant Permissions to the New User

To grant specific permissions to the new user:

- **Right-click on the new user** under the **Users** folder in the target database.


- Select **Properties**.
- Go to the **Securables** page and add objects (like tables or stored procedures) to which you want to grant permissions.



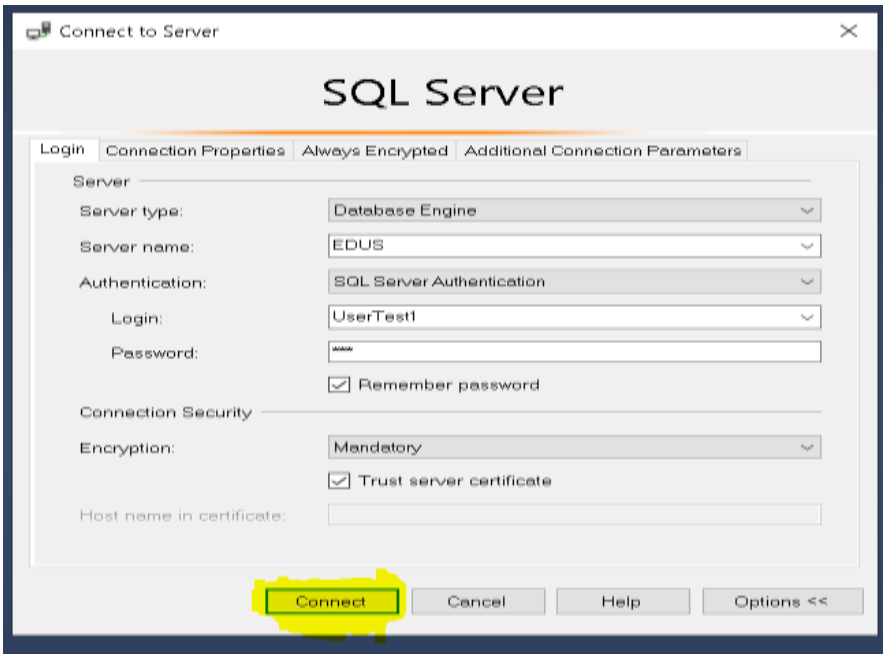

- Select the specific permissions (e.g., SELECT, INSERT, UPDATE, DELETE) and click **OK**.

### Test the Authentication System

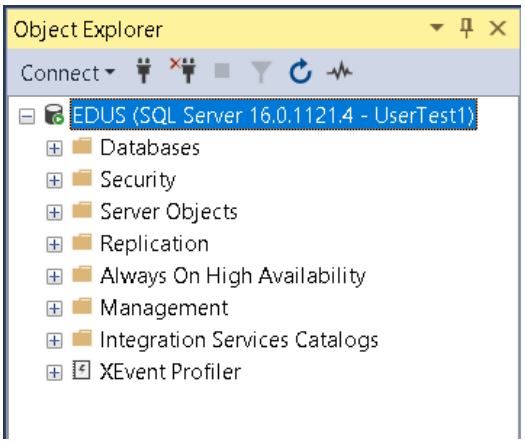
To test the authentication system, log in to SSMS using the new login credentials and verify that the permissions are correctly set.

- **Test a User Login**:

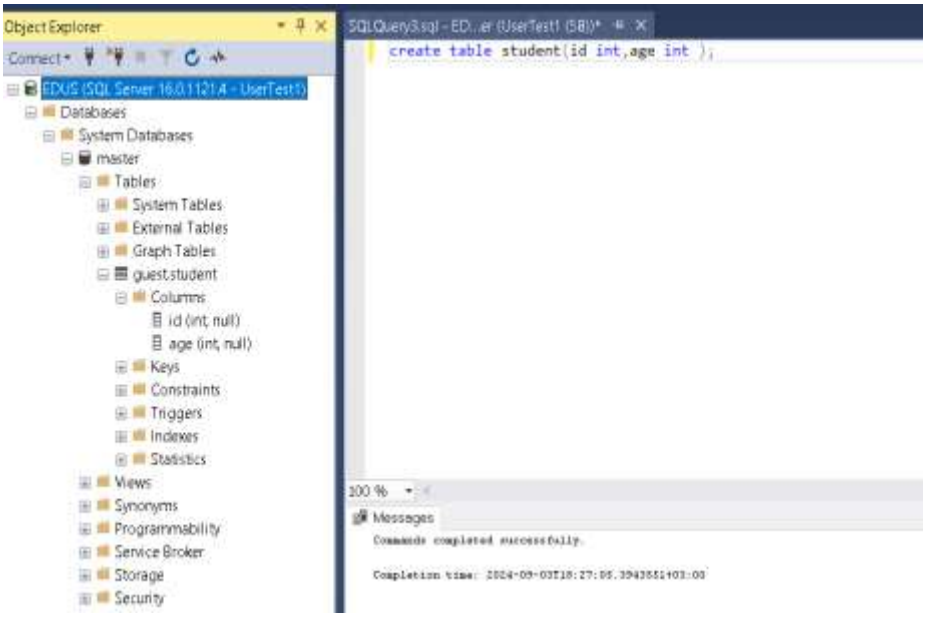
- Open **SSMS** and click **Connect**.
- Choose **SQL Server Authentication** or **Windows Authentication**, depending on how the login was created.
- Enter the **Login Name** and **Password**.




- Click **Connect** to verify if the login is successful.




- **Test Database Access and Permissions**:
  - After logging in, attempt to perform actions based on assigned permissions (e.g., run a SELECT statement on a table).
  - Verify if the user can or cannot perform the actions as expected.




### Monitor and Maintain

Monitoring and maintaining involve periodically reviewing logins, roles, and permissions to ensure security and compliance.

- **Review Server Logins**:
  - Expand the **Security** folder in **Object Explorer**.
  - Review the list of **Logins** and their properties to ensure only authorized users have access.
- **Review Database Users and Roles**:
  - Expand **Databases**, then expand a database.
  - Expand **Security > Users** to review users in each database and their roles.

**Practical Activity 4.1.4: Authorizing database users in SQL server**


**Task:**

1: Read key reading 4.1.4.in trainee manual

2: Read the task described below:

As the database developer for ABC Corporation, you have been tasked with enhancing the security and authorization controls within the company's SQL Server environment. Your tasks are:

- To create roles to manage user permissions efficiently. Begin by creating a server-level role named CustomServerRole and assign an appropriate owner.
- Within the Test1 database, create a database role named Test1DBRole to handle specific database-level permissions.
- Once the roles are set up, proceed to assign permissions to ensure proper access control. Grant server-level permissions.

**For Test1DBRole**,

- Assign database-level permissions including SELECT, INSERT, UPDATE, and DELETE within the Test1 database.
- Assign these roles to users. Link CustomServerRole to a specific login that requires server-level access, and assign Test1DBRole to a user within the Test1 database to ensure they have the necessary permissions to manage data.
- After configuring these roles and permissions, test the authorization by logging in with the respective user credentials. Perform various actions and attempting to create a database to ensure the user permissions are functioning correctly as defined.
- Set up an ongoing process to monitor and maintain these roles and permissions.

3: Perform the task 2 described above

4: Present your work to your trainer and a whole class

**Key readings 4.1.4: Authorizing database users in SQL server**

- **Authorization**

Authorization in SQL Server involves managing access control by defining roles and assigning permissions to these roles. This determines what actions users can perform on

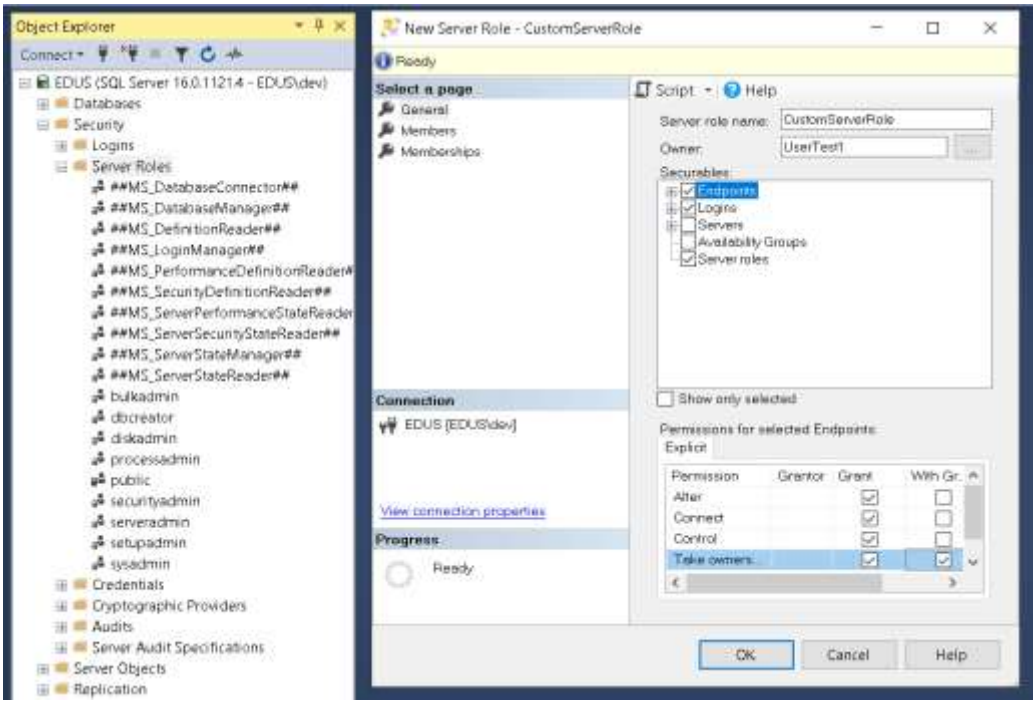
the server or database objects.

✓ **Create Roles**

You can create roles at both the **server level** (server roles) and **database level** (database roles).

**Create a Server Role**

- **Open SSMS** and connect to your SQL Server instance.
- In **Object Explorer**, expand the **Security** folder.
- Right-click on **Server Roles** and select **New Server Role...**.
- In the **New Server Role** window:
  - Enter the **Role Name** (e.g., CustomServerRole).
  - Assign an **Owner** for the role.
  - Use the **Securables** tab to define which server-level objects the role has access to (e.g., granting control over specific databases).

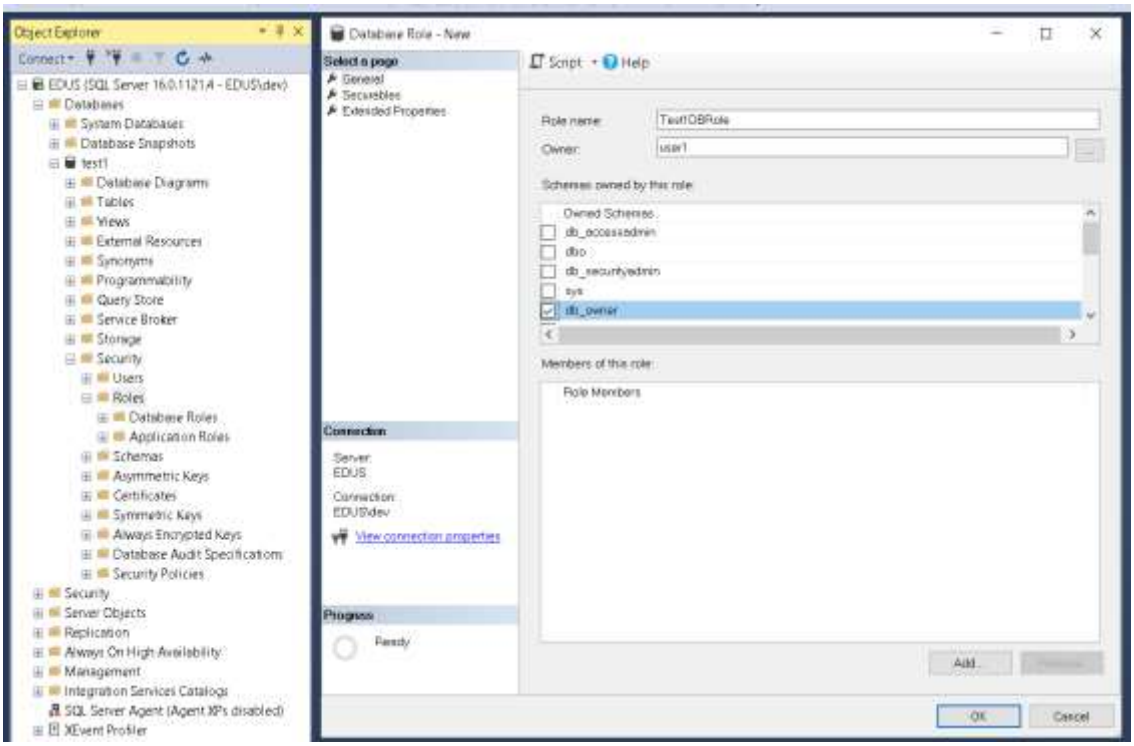



- Click **OK** to create the server role.

**Create a Database Role**

- Expand the **Databases** folder, then expand the target database (e.g., Test1.)
- Expand the **Security** folder under the database.

- Right-click on **Roles > Database Roles** and select **New Database Role...**.
- In the **Database Role - New** window:
  - Enter the **Role Name** (e.g., Test1DBRole).
  - Specify the **Owner** of the role.




- Click **OK** to create the database role.

### Assign Permissions/Privileges to Roles

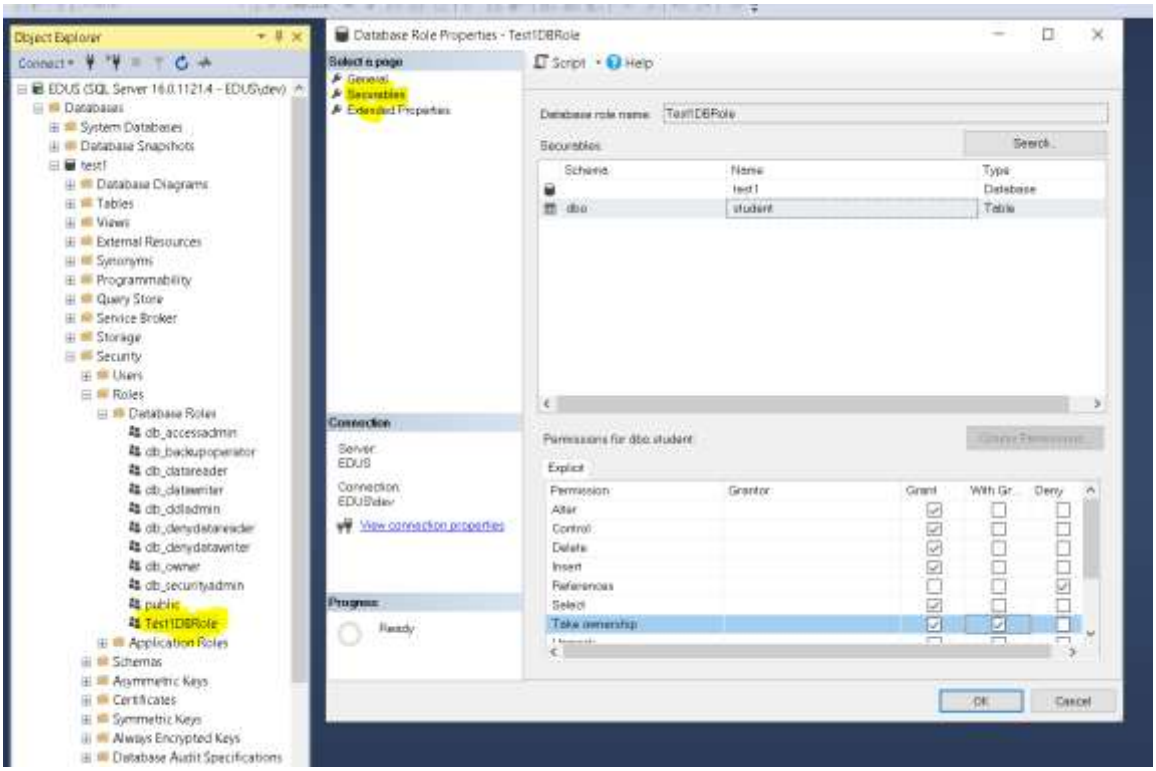
Once roles are created, you can assign permissions to them. Permissions can be granted at the object level (tables, views, stored procedures, etc.).

### Assign Server-Level Permissions to a Server Role:

- Right-click on the server role you created under **Security > Server Roles**.
- Select **Properties**.
- Go to the **Securables** tab.
- Add specific server-level objects or databases to the list.
- Check the desired permissions (e.g., ALTER, CONTROL, CONNECT).
- Click **OK** to apply changes.

### Database-Level Permissions to a Database Role:

- Go to **Databases > YourDatabase > Security > Roles > Database Roles**.
- Right-click on the database role you created and select **Properties**.
- Go to the **Securables** tab.
- Add specific objects (e.g., tables, views, stored procedures) to the list.
- Grant specific permissions like SELECT, INSERT, UPDATE, or DELETE.

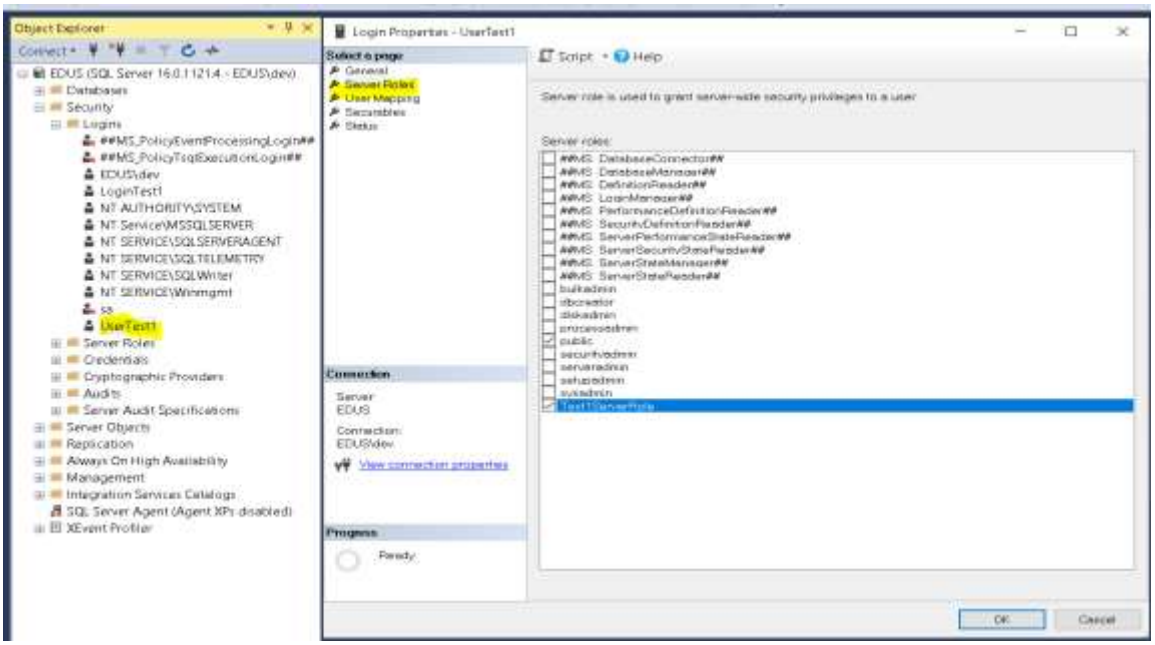



- Click **OK** to apply the permissions.

### Assign Roles to Users

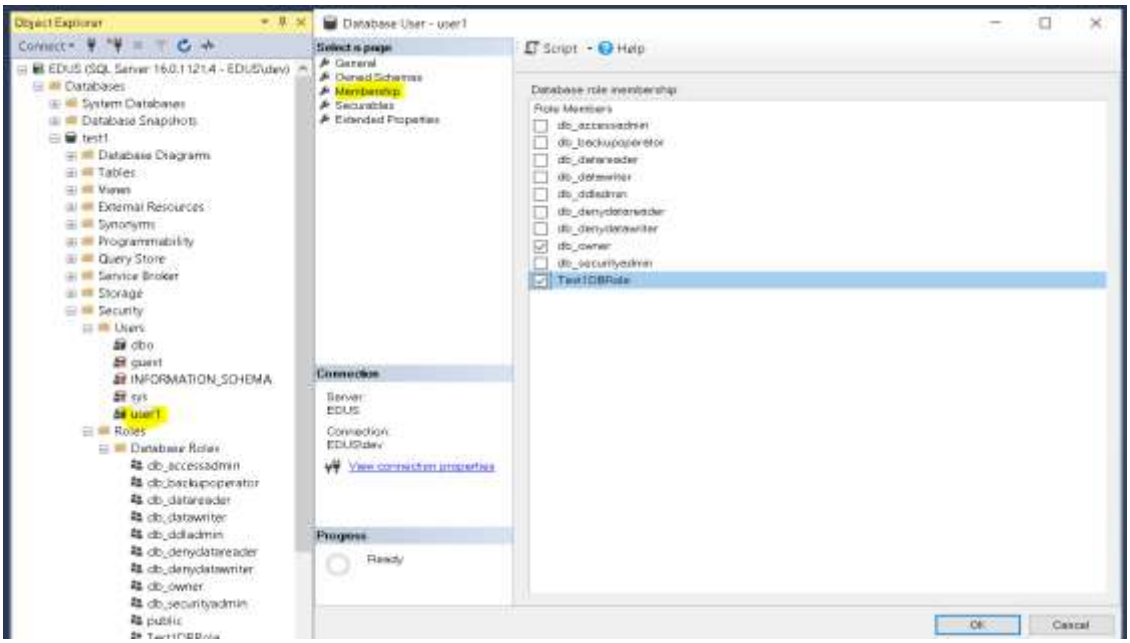
After defining roles and their permissions, assign these roles to users or logins.

- **Assign a Server Role to a Login**:
  - Expand the **Security > Logins** folder.
  - Right-click on the login you want to assign a role to and select **Properties**.
  - Go to the **Server Roles** tab.
  - Check the server roles (e.g., Test1ServerRole) that you want to assign.



- Click **OK**.

- **Assign a Database Role to a User**:
  - Expand **Databases > YourDatabase > Security > Users**.
  - Right-click on the user you want to assign a role to and select **Properties**.
  - Go to the **Membership** tab.
  - Check the database roles (e.g., Test1DBRole) that you want to assign.



- Click **OK**.

### Test the Authorization System

To test the authorization system, log in using the user credentials and verify the permissions and roles assigned.

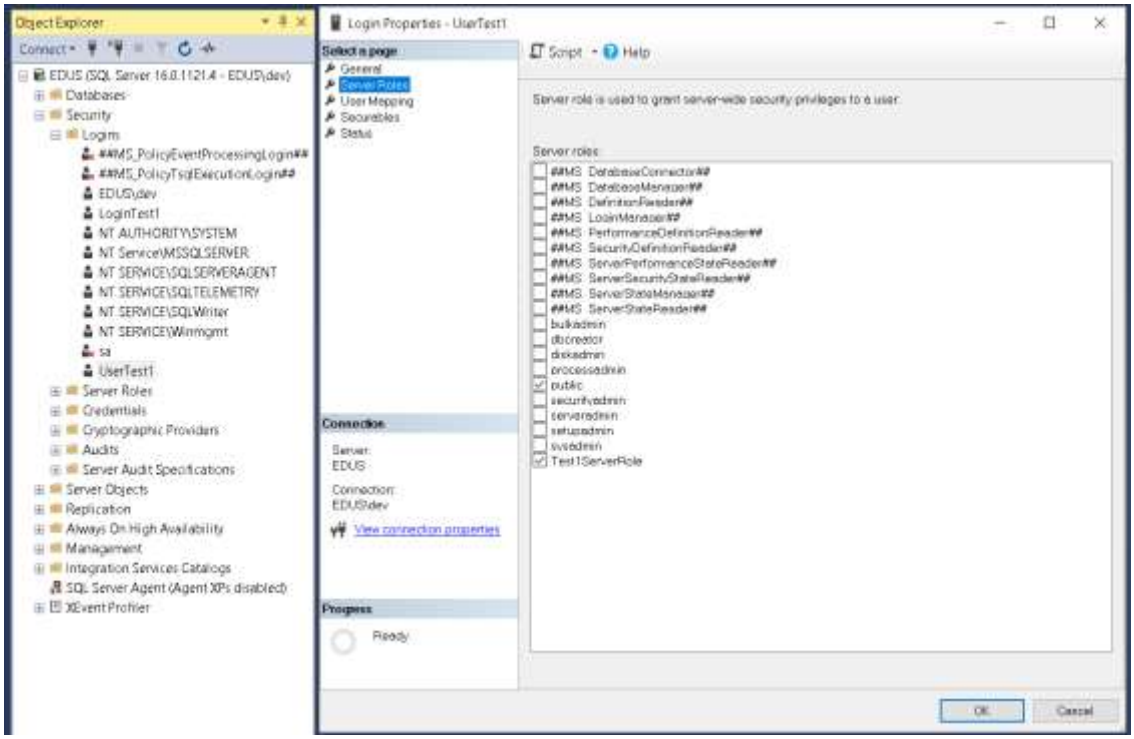
- **Test with a User Login**:
  - Disconnect from the current session in SSMS.
  - Click **Connect** and use the login credentials of the user for whom you assigned roles.
  - Try performing various actions like selecting data, updating a table, or creating a database, based on the permissions assigned to their roles.
  - Verify that the user can only perform actions that their roles allow.

### Monitor and Maintain

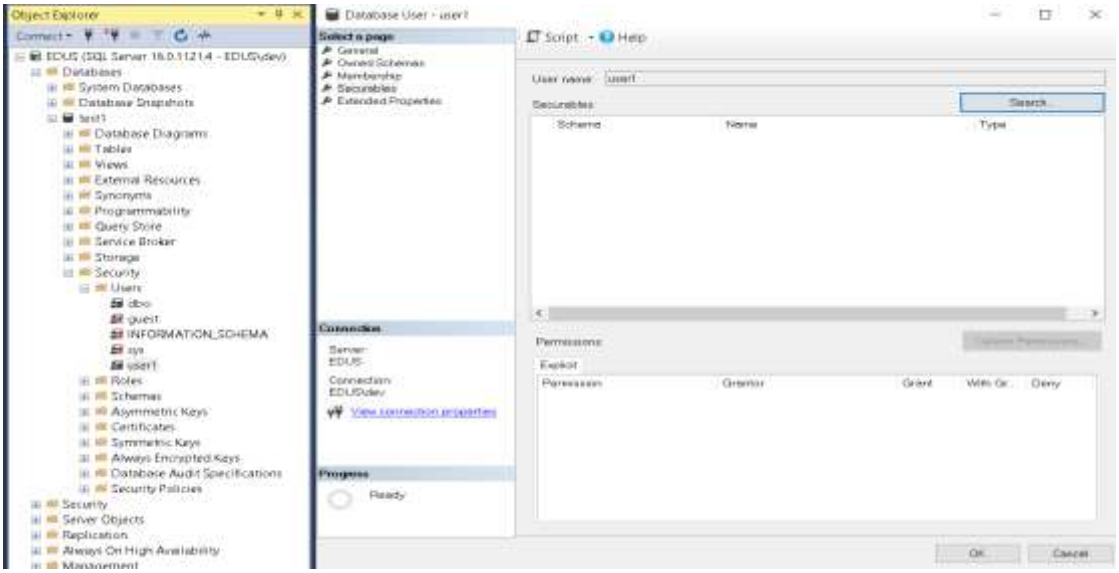
Monitoring and maintaining the authorization system involves regularly reviewing roles, permissions, and user assignments to ensure they meet security policies.

- **Review Roles and Permissions**:
  - Periodically review **Security > Server Roles** and **Security > Database Roles** to ensure that only necessary roles exist and that they have appropriate permissions.
  - Right-click on roles and select **Properties** to view and adjust permissions if needed.
- **Review User Role Assignments**:
  - Review user assignments under **Security > Logins** (for server roles) and **Security > Users** (for database roles).

    i. for server roles




ii. for database roles



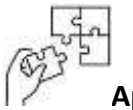

- Make changes if users no longer need certain roles or need additional permissions.

- **Remove Unused or Outdated Roles**:
  - Right-click on any role under **Server Roles** or **Database Roles** that is no longer needed and select **Delete** to remove it.
  - Confirm the deletion.

- **Audit User Activity**:
    - Set up **SQL Server Audit** to monitor and log activities for specific users or roles.
    - Use **SQL Server Profiler** to track real-time activities and detect any unauthorized actions.

**Points to Remember**

- Database security is the process, tool and controls that secure and protect databases against accidental and intentional threats.
- To Identify database security types, focus on Physical Security, authentication and Authorization, Data Encryption, auditing, and Backup and Recovery.
- Indicate the categories of data in the database
- Identify the terms used in database control access by focusing on Role, Permission, Authentication, authorization
- Before adding users to the database, check for existing users to avoid errors caused by duplicate usernames in the DBMS.
- A newly created user will use the "new login" feature in SQL Server to access the database.
- Authorize database users by assigning them appropriate "database roles" and "server roles."
- Adjust the configuration manager according to the authentication methods that will be implemented.
- Log in with different credentials to verify the effectiveness of the authentication process.
- Formulate a role name that reflects the access rights associated with that role.
- Assign permissions to the created role to define what actions it can perform.
- Assign the roles to the created user(s) to grant them the specified access rights.
- Test authorization by logging in with user credentials and verifying the assigned permissions.



**Application of learning 4.1.**

Bright Future Academy school located in Musanze District, Kanama Sector, Munini Village manages a large amount of data for students, teachers, courses, grades, and administrative operations in a SQL Server database named **SchoolDB**. To ensure secure access and maintain proper controls over the data, you need to set up and manage

roles, permissions, authentication, and authorization within SQL Server Management Studio (SSMS). As a Database Administrator you are asked to perform the followings tasks in order to secure its database.

**Tasks to Perform:**

1. **Define Roles and Permissions**
2. **Authenticate database**
3. **Authorize: database users**

## Indicative content 4.2: Management of Auditing and Logging

**Duration: 5 hrs**

### Theoretical Activity 4.2.1: Description of database Auditing and logging

**Tasks:**

1: You are asked to answer the following questions related to database auditing and logging:

i. What is data access control?
ii. Identify data classifications
iii. Differentiate role from permission
iv. Differentiate authentication from authorisation

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 4.2.1.

**Key readings 4.2.1.: Description of database Auditing and logging**

- **Logging**

**Audit logging in a database** refers to the process of recording and tracking activities that occur within the database. This includes actions taken by users, changes made to data, and modifications to database structures. Audit logs are crucial for monitoring user activity, ensuring data integrity, and maintaining security and compliance.

**Logging in a database** refers to the process of recording various operations and transactions that occur within the database. The primary purpose of logging is to ensure data integrity, recoverability, and consistency in case of system failures or errors. Database logs keep a sequential record of changes to the data, queries executed, and other events, providing a way to undo (roll back) or redo (reapply) transactions as needed.

✓ **Types of Logs**:

- **Write-Ahead Log (WAL)**: Ensures that the log is written before any changes are made to the database itself. This guarantees that in case of a crash, the changes can be replayed from the log to restore the database.
- **Redo Log**: Stores changes that need to be reapplied to recover the database in case of a failure. Used in crash recovery.

- **Undo Log**: Records changes that need to be undone if a transaction fails or is rolled back.

- **Auditing**

**Auditing in a database** refers to the process of monitoring and recording selected user actions and database activities to ensure compliance, enhance security, and provide accountability. Auditing helps organizations track who accessed the database, what operations were performed, and whether any unauthorized actions were taken. This is crucial for security, legal compliance, troubleshooting, and performance analysis.

✓ **Types of Database Auditing:**

- **Access Auditing**:
  - Records user login attempts, including both successful and failed attempts.
  - Logs the duration of user sessions, IP addresses, and other metadata related to access.
- **Data Modification Auditing**:
  - Tracks changes to database records, including INSERT, UPDATE, and DELETE operations.
  - Captures the "before" and "after" state of the data to show exactly how it was changed.
- **Schema Auditing**:
  - Monitors structural changes to the database schema, such as creating or altering tables, views, procedures, or indexes.
- **Permission Auditing**:
  - Logs changes to database roles and permissions, such as granting or revoking access to specific users.
  - Tracks when a user or group is given additional privileges or when their access is restricted.
- **Backup and Restore Auditing**:
  - Records all database backup and restore operations.
  - Provides details on who performed the action, the start and end times, and the success or failure of the operation.

**Practical Activity 4.2.2: Managing database logging**


**Task:**

1: Read key reading 4.2.2

2: Read the below task:

XYZ Corporation, a growing financial services company, uses a SQL Server-based Financial Management System (FMS) to handle sensitive data related to clients, transactions, and employee activities. Given the importance of security, data integrity, and performance, the company needs to implement comprehensive logging and monitoring on its SQL Server to track various activities. These logs will be used for audit purposes, performance tuning, and ensuring compliance with industry standards.

As an employee of XYZ corporation you are assigned to manage logging for XYZ Corporation's FMS, you will use SQL Server Management Studio (SSMS) to set up, monitor, and maintain SQL Server logs for security, performance, and error tracking.

3: Perform task2 described above.

4: Present your work to your trainer and a whole class

**Key readings 4.2.2: Managing database logging**

- **Logging in SQL Server Using SSMS (SQL Server Management Studio)**

Logging in SQL Server involves capturing and storing detailed information about database operations and user activities. Effective logging helps in monitoring, troubleshooting, and maintaining database security by keeping track of various events and actions.

✓ **Steps to Manage Logging in SSMS**

**Identify the Logging Requirements**

To set up logging in SQL Server, first identify the events that need to be logged. Consider the following requirements:

- **User Activities**: Log actions such as SELECT, INSERT, UPDATE, and DELETE.
- **Security Events**: Log failed login attempts and permission changes.
- **System Events**: Log server errors, database startup, and shutdown events.
- **Performance Monitoring**: Log query performance statistics and resource

utilization.

- **Configure Logging Settings**

To configure logging settings, you need to set up SQL Server to log the identified events and store them in an appropriate format (e.g., file, table, or event viewer).

- ✓ **Set Up SQL Server Error Logs:**
- In SSMS, connect to your SQL Server instance.
- Expand the **Management** folder and right-click on **SQL Server Logs**.
- Select **Configure SQL Server Logs**.
- Choose the **Log Destination** (e.g., File, Windows Application Log).
- Set the **Maximum Number of Error Logs** and **Size** to manage disk space usage.
- Click **OK** to save the configuration.
- ✓ **Configure Extended Events for Logging:**
- In SSMS, expand the **Management** folder.
- Right-click on **Extended Events** and select **New Session...**.
- In the **New Session Wizard**, provide a **Session Name**.
- Choose the events to capture (e.g., sql_batch_completed, rpc_completed).
- Specify **Event Fields** to include additional data such as client_app_name or database_id.
- Set the **Storage Target** (e.g., file or ring buffer) and specify the file path if using a file.
- Click **OK** to create and start the session.
- ✓ **Monitor Log Data**

- **Monitor SQL Server Error Logs:**
  - In SSMS, expand the **Management** folder.
  - Right-click on **SQL Server Logs** and select **View SQL Server Log**.
  - Use filters to view specific logs (e.g., errors, warnings, or informational messages).

- **Monitor Extended Events Logs:**
  - Expand **Management > Extended Events > Sessions**.
  - Right-click on your logging session and select **Watch Live Data**.
  - View real-time data and apply filters for specific events or user activities.
- **Analyze Log Data**
  - **Analyze SQL Server Error Logs:**
    - Use the **View SQL Server Log** option to analyze entries based on severity, error codes, and user actions.
    - Identify patterns of repeated errors or security breaches.
  - **Analyze Extended Events Logs:**
    - Export the extended events data to a file or table for detailed analysis.
    - Use SSMS or third-party tools to query and visualize log data for insights into database performance and security.
- **Archive Log Data**
  - **Archive SQL Server Error Logs:**
    - In SSMS, expand **Management > SQL Server Logs**.
    - Right-click on **SQL Server Logs** and select **Recycle SQL Server Error Log** to close the current log file and start a new one.
    - Regularly move older log files to a secure archive location to manage disk space and maintain records.
  - **Archive Extended Events Logs:**
    - Set up a job in SQL Server Agent to periodically copy log files to an archive directory.
    - Compress archived logs to save space and protect sensitive information.
- **Corrective Action**
  - **Take Corrective Actions Based on Log Data:**
    - If frequent errors are detected, investigate and fix the underlying issues (e.g., optimize queries, fix faulty applications).
    - If suspicious activity or unauthorized access is identified, review and update security settings, user permissions, and firewall rules.

- Ensure log settings are correctly configured to capture all necessary events without affecting server performance.

**Practical Activity 4.2.3: Managing database auditing**


**Task:**

1: Read key reading 4.2.3.in trainee manual

2: Read the below task

As the Database Administrator at DataSecure Solutions, your role involves configuring and managing auditing for the CompanyDB database using SQL Server Management Studio (SSMS) to ensure security and compliance.

i. To begin, you will identify the key events that need to be audited
ii. Next,configure the audit settings by creating a SQL Server Audit with a designated target.
iii. Setting up a Server Audit Specification to track server-level events and a Database Audit Specification to monitor specific database-level activities
iv. Enable the audit and review the audit logs in SSMS to monitor activities and identify any unauthorized access or changes.
v. Using the Audit Logs Viewer, analyze the audit data by filtering entries based on user activity, timestamps, and actions performed, paying close attention to any suspicious activities.
vi. If any unauthorized access or suspicious activity is detected, you will take corrective actions by adjusting user permissions, tightening security settings, or modifying the audit specifications to improve monitoring accuracy.

3: Ask clarification where necessary

4: Manage auditing, as required based on the above task

5: Present your work to your trainer and a whole class

**Key readings 4.2.3: Managing database auditing**

- **Auditing in SQL Server Using SSMS (Sql Server Management Studio)**

Auditing in SQL Server involves tracking and recording events related to database access and changes. SQL Server provides built-in auditing capabilities to help meet regulatory compliance and security requirements.

✓ **Steps to Manage Auditing in SSMS**

**Identify the Data That Needs to Be Audited**

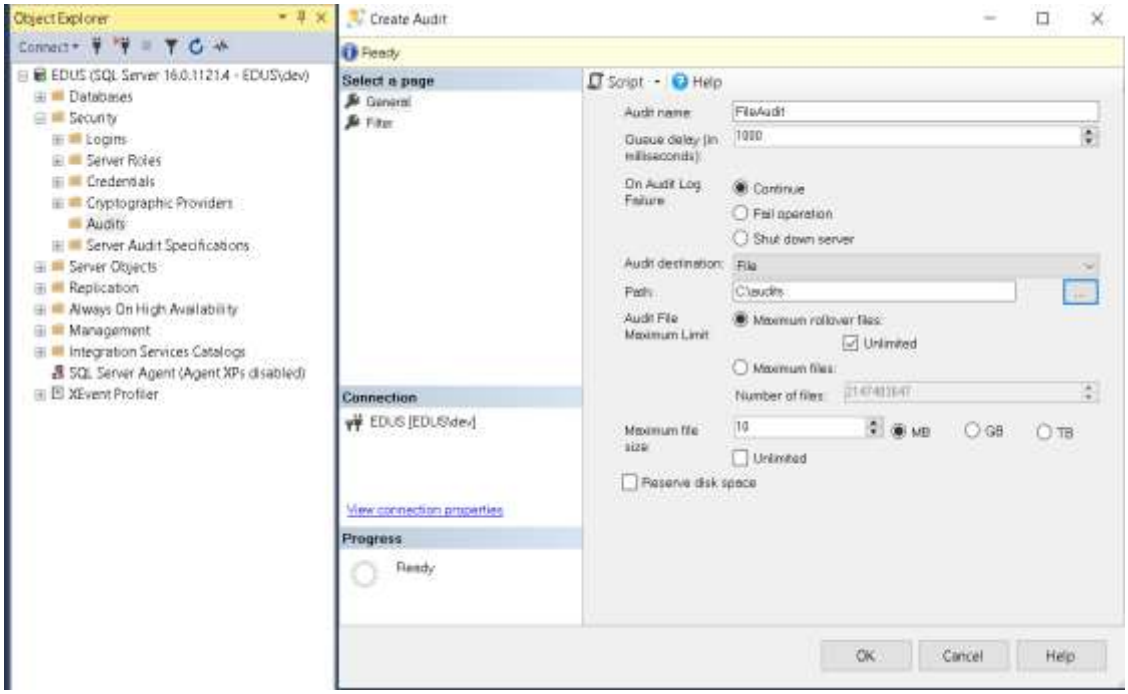
Determine which events and data need to be audited. This can include:

- Login attempts (successful or failed).
- Data modifications (e.g., INSERT, UPDATE, DELETE).
- Permission changes.
- Schema changes.

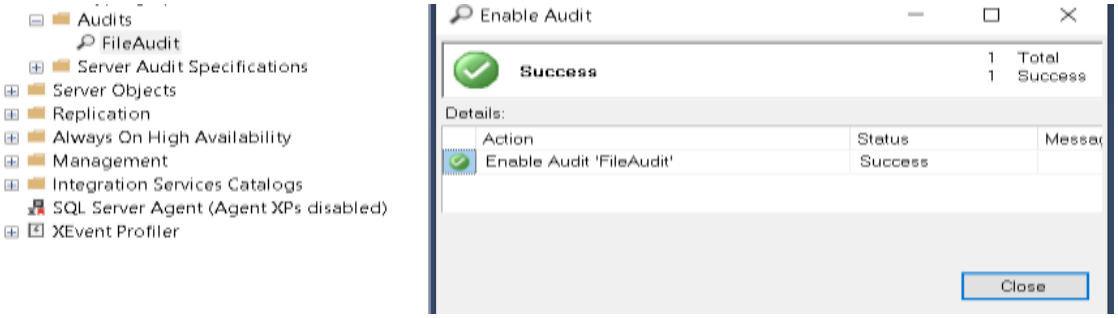
**Configure Audit Settings**

To configure SQL Server Audit settings, create an audit object that specifies the target (e.g., file, security log, application log) and then define audit specifications for server or database-level actions.

- **Create a SQL Server Audit**:
  - In **SSMS**, expand the **Security** folder.
  - Right-click on **Audits** and select **New Audit...**.
  - In the **Create Audit** window:
    - Enter an **Audit Name**.
    - Set the **Audit Destination** (e.g., a file, Application Log, or Security Log).
    - Specify the **File Path** if you are using a file as the destination.

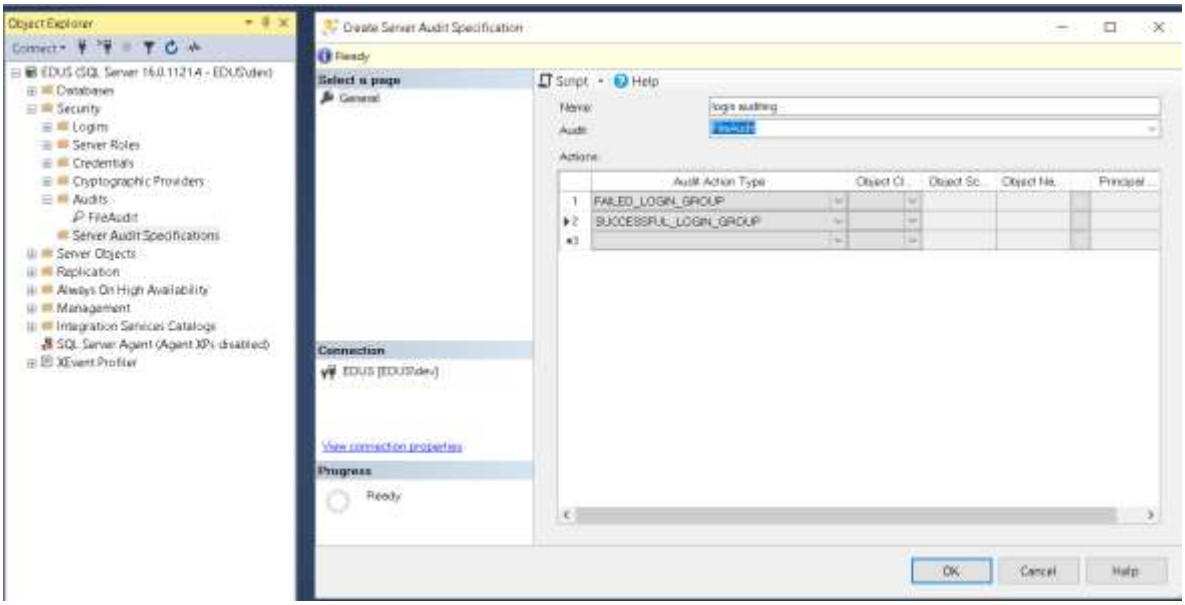



- Click **OK** to create the audit.
- Click on created audit (e.g., FileAudit) and select **Enable audit.**

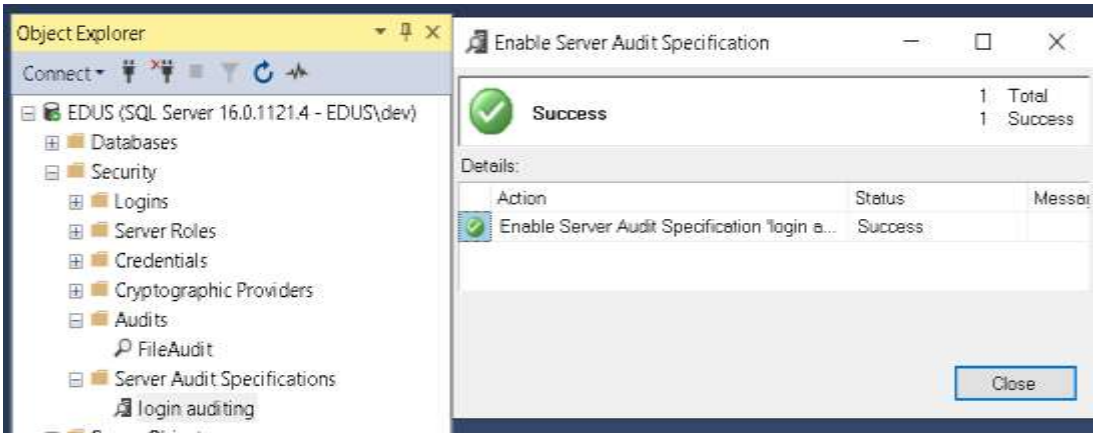



- Click **close** to enable the audit.

- **Create a Server Audit Specification**:
  - Right-click on **Server Audit Specifications** under **Security** and select **New Server Audit Specification...**.
  - In the **Create Server Audit Specification** window:
    - Enter a **Name** for the specification.
    - Select the **Audit** you created earlier.
    - Choose the **Action Types** to audit (e.g., FAILED_LOGI_GROUP).



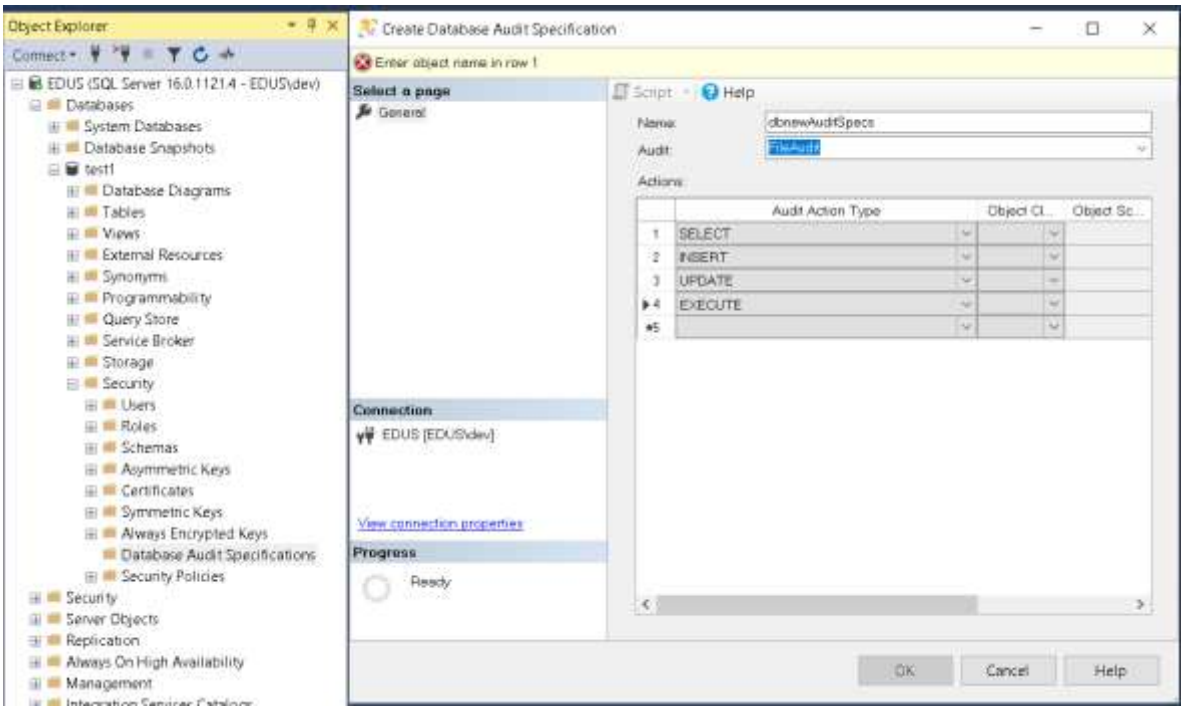
- Click **OK**.
- Click on created Server Auditing Specification (e.g., login auditing) and select **Enable Server Auditing Specification.**



-

- **Create a Database Audit Specification**:
  - Expand **Databases > YourDatabase > Security**.
  - Right-click on **Database Audit Specifications** and select **New Database Audit Specification...**.
  - In the **Create Database Audit Specification** window:
    - Enter a **Name**.
    - Select the **Audit** you created earlier.

- Choose **Audit Action Types** (e.g., SELECT, INSERT, UPDATE).




- Click **OK**.

**Review the Audit**

- **Enable and Review Audits**:
  - Right-click on the created **Audit** under **Security > Audits** and select **Enable**.
  - Right-click on the **Server Audit Specification** or **Database Audit Specification** and select **Enable**.
  - View the audit logs by right-clicking on the **Audit** and selecting **View Audit Logs**.

**Analyze Audit Data**

- **Analyze Audit Logs**:
  - Use the **Audit Logs Viewer** in SSMS to analyze entries based on user activity, timestamps, and actions performed.
  - Look for unauthorized access, changes, or suspicious activities.

**Corrective Action**

- **Take Corrective Actions Based on Audit Data**:
  - If unauthorized access or suspicious activity is detected, review user

permissions, tighten security settings, or take disciplinary actions as required.

- Adjust audit specifications to better capture relevant data or exclude unnecessary events.

**Points to Remember**

- Logging in databases records events, providing a way to track operations, detect security incidents, and facilitate recovery from failures.
- Determine which events need to be logged.
- Set up SQL Server error logs and Extended Events sessions to capture specific activities.
- Use SSMS to view and monitor error logs and Extended Events for tracking logged activities.
- Review logs for patterns of unauthorized activities or failed login attempts using SQL queries.
- Determine events to be audited, such as login attempts, data modifications, and permission changes.
- Set up SQL Server Audit, Server Audit Specification, and Database Audit Specification for targeted events.
- Enable audit settings and review audit logs in SSMS for monitoring activities.
- Use SSMS to analyse audit logs for user activities and detect any unusual actions.
- Adjust permissions, tighten security, or update audit settings if suspicious activity is found.

**Application of learning 4.2.**

As the Database Administrator for Bright Future Academy, you are responsible for ensuring the security and compliance of the SchoolDB database, which stores sensitive information such as student records, grades, and personal data. To maintain security and adhere to compliance standards, you will implement logging and auditing mechanisms using SQL Server Management Studio (SSMS).

- First, identify logging requirements, focusing on tracking events and failed login attempts.
- Then, configure logging settings, creating an Extended Events session named SchoolDBActivityLog to capture these activities, ensuring logs are stored securely.

- Regular monitoring of SQL Server error logs and Extended Events logs.
- Set up a process to periodically archive log data for future analysis or audits, and if suspicious activities are detected, you will take corrective action such as adjusting user permissions or reviewing security settings.

On the auditing side,

- Identify key data to be audited and permission or schema changes.
- Using SSMS, configure audit settings by creating a SQL Server Audit with a target destination, link it to a Server Audit Specification to track server-level and set up a Database Audit Specification for auditing SchoolDB-level activities

## Indicative content 4.3: Implementation of Data Encryption

**Duration: 4 hrs**

### Theoretical Activity 4.3.1: Description of Data encryption

**Tasks:**

1: You are asked to answer the following questions related to data encryption

I. What is data encryption

II. Give and explain data encryption techniques

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 4.3.1.

**Key readings 4.3.1: Description of Data encryption**

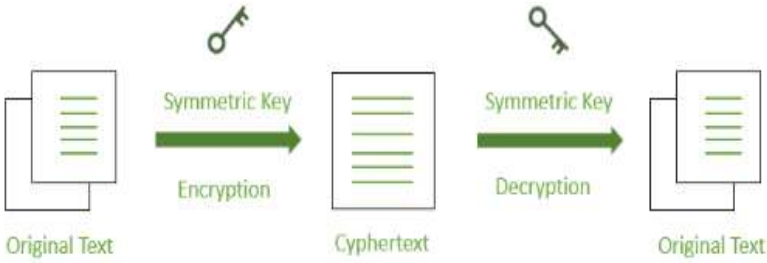
- **Data encryption**
  - ✓ **Definition**

Data Encryption is a method of preserving data confidentiality by transforming it into ciphertext, which can only be decoded using a unique decryption key produced at the time of the encryption or before it. The conversion of plaintext into ciphertext is known as encryption.

- ✓ **Encryption technics**

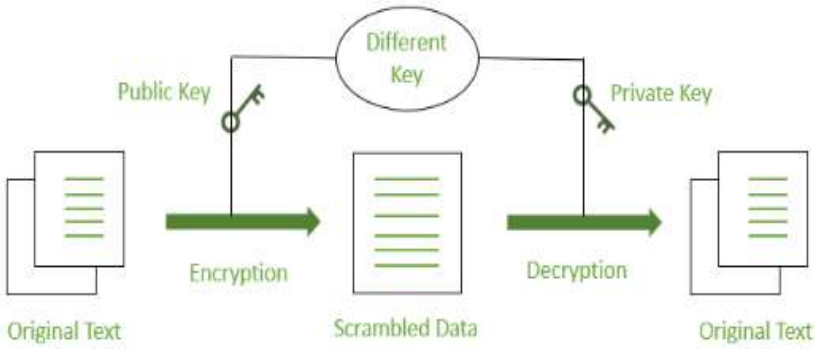
There are multiple encryption techniques, each of which have been developed with various security requirements in mind. Symmetric and Asymmetric encryption are the two types of data encryption.

- **Symmetric Encryption**
  - **Symmetric Encryption**: In symmetric encryption, the same key is used for both encryption and decryption. Popular algorithms include AES (Advanced Encryption Standard), DES (Data Encryption Standard), and Blowfish.




*Symmetric Encryption*

- **Asymmetric Encryption**
  - **Asymmetric Encryption (Public Key Encryption)**: Asymmetric encryption uses two keys: a public key for encryption and a private key for decryption. RSA and ECC (Elliptic Curve Cryptography) are common asymmetric encryption algorithms.




*Asymmetric Encryption*

- **Hashing**
  - **Hashing**: Hashing converts data into a fixed-size string of characters, typically a digest or hash value. It's a one-way process, meaning you can't reverse the hash to retrieve the original data. Common hash functions include SHA-256 (Secure Hash Algorithm) and MD5 (Message Digest Algorithm 5).

**Practical Activity 4.3.2: Implementing Data Encryption in SQL Server**


**Task:**

1: Read key reading 4.3.2.in trainee manual

2: Read the below task

As the lead database developer at TechSolutions Inc., you are tasked with improving the security of the company's customer and financial data by applying various encryption techniques. The company handles sensitive data, including customer information and financial transactions, and requires strong data protection measures.

- First, you will implement **symmetric encryption** to secure large volumes of data at rest
- Using **AES (Advanced Encryption Standard)**, encrypt the data
- Next, incorporate **asymmetric encryption** for secure communications and key exchange between the company's internal systems and external partners.
- Utilize **RSA encryption** to secure data transmission, allowing external partners to encrypt information using TechSolutions' public key, which can only be decrypted with the private key held securely by the company.
- Finally, implement **hashing** for password storage and data integrity checks.
- Use a secure hashing algorithm like **SHA-256** to hash user passwords before storing them in the database, ensuring that even if the database is compromised, the original passwords remain protected.
- Additionally, apply hashing to verify the integrity of data files, ensuring that any unauthorized changes can be detected.

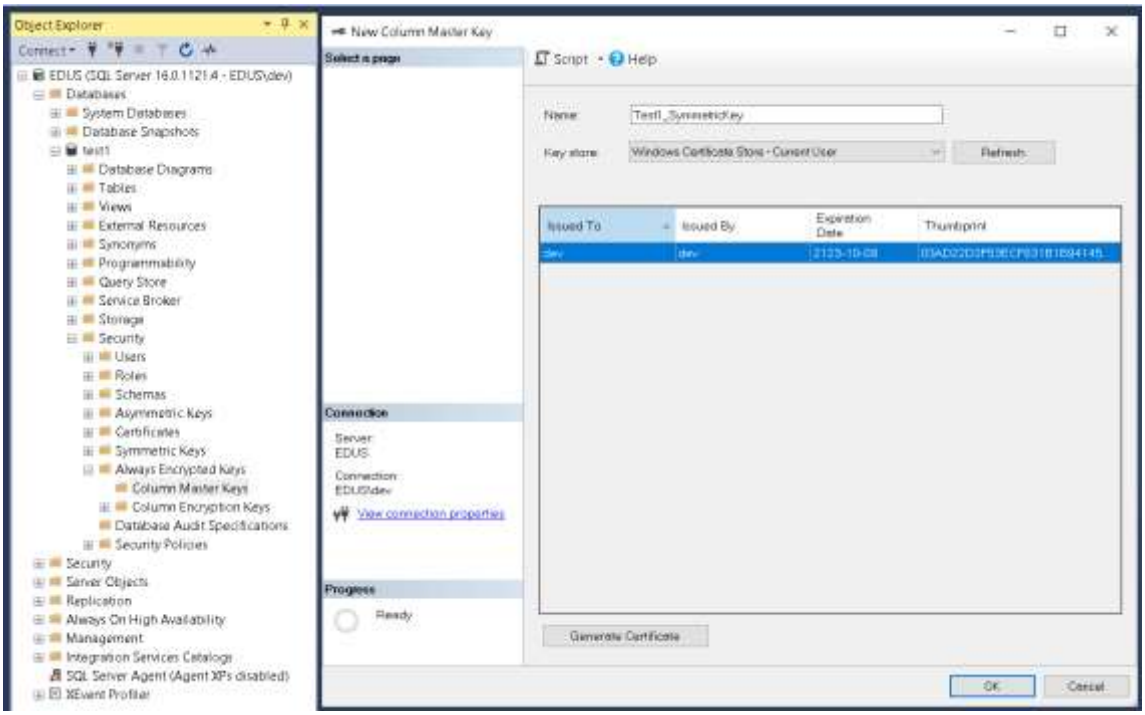
3: Encrypt data as requested in the above task2

4: Present your work to your trainer and a whole class

**Key readings 4.3.2: Implementing Data Encryption in SQL Server**

- **Steps to Implement Symmetric Encryption:**
  - ✓ **Create a Master Key**:
    - Open SSMS and connect to your SQL Server instance.
    - In **Object Explorer**, expand the **Databases** folder, and then expand your target database.
    - Expand **Security > Always Encrypted Keys > Column Master Keys**.
    - Right-click on **Column Master Keys** and select **New Column Master Key**.

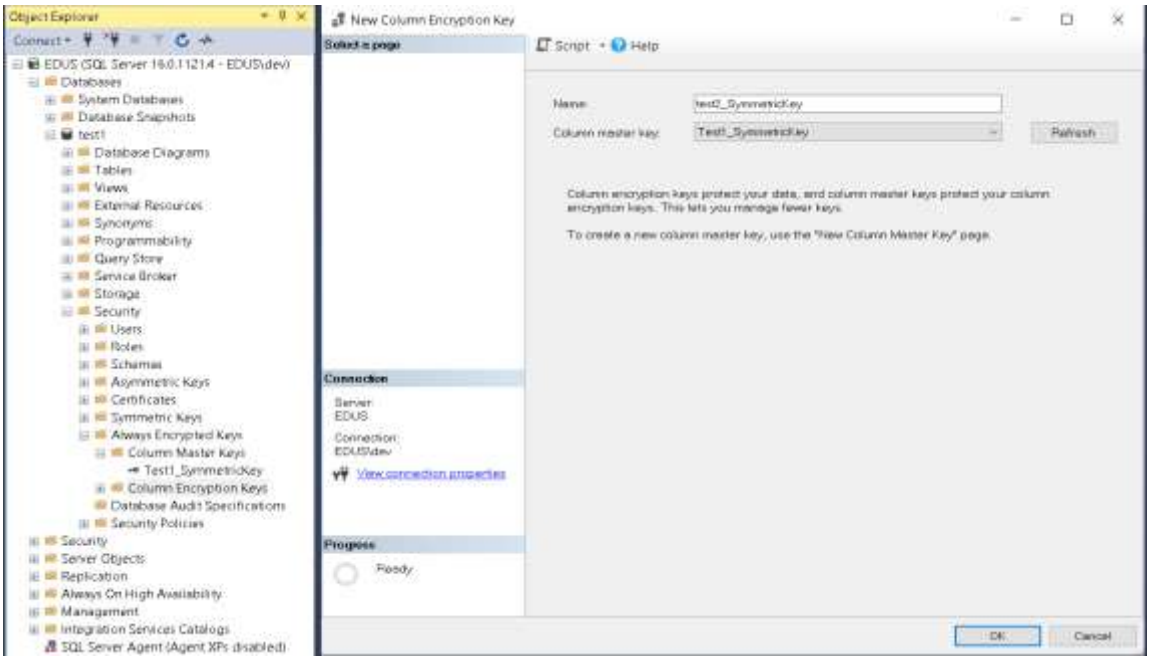
- In the **New Column Master Key** dialog:
  - Enter a **Name** for the master key (e.g., Test1_SymmetricKey).
  - Choose the **Key Store Provider** (e.g., Windows Certificate Store).




  - Click **OK** to create the master key.

✓ **Create a Symmetric Key**:

- Right-click on **Always Encrypted Keys > Column Encryption Keys** and select **New Column Encryption Key**.
- In the **New Column Encryption Key** dialog:
  - Enter a **Name** for the encryption key (e.g., test2_SymmetricKey).
  - Select the **Column Master Key** you created earlier.



- Click **OK** to create the symmetric key.

✓ **Encrypt a Column Using Symmetric Encryption**:

- Right-click on the table with the column you want to encrypt.
- Select **Design** to open the table in design mode.
- Select the column to encrypt and go to the **Column Properties**.
- Under **Encryption**, select the **Column Encryption Key** you created.
- Save the table to apply encryption.

- **Asymmetric Encryption**

**Asymmetric Encryption** involves two keys: a public key for encryption and a private key for decryption. It is more secure than symmetric encryption but slower.

Steps to Implement Asymmetric Encryption:

✓ **Create an Asymmetric Key**:

- Expand **Databases > YourDatabase > Security > Asymmetric Keys**.
- Right-click on **Asymmetric Keys** and select **New Asymmetric Key...**.
- In the **New Asymmetric Key** dialog:
  - Enter a **Name** (e.g., AsymmetricKey_Encryption).
  - Select the **Key Source** and **Algorithm** (e.g., RSA_2048).
  - Click **OK** to create the asymmetric key.

✓ **Encrypt Data Using Asymmetric Key**:

- Right-click on a column or table and choose **Design**.
- For columns that require encryption, set the encryption properties using the **Asymmetric Key**.
- Save the changes to apply encryption to the selected column.

**Hashing**

**Hashing** is used for data integrity checks rather than encryption. It generates a fixed-size string that cannot be reversed to retrieve the original data.

Steps to Apply Hashing:

✓ **Use Computed Columns for Hashing**:

- Right-click on the table where you want to apply hashing and select **Design**.
- Add a new computed column (e.g., HashedColumn).
- Set the **Formula** to use a hashing function like HASHBYTES ('SHA2_256', [ColumnName]).
- Save the table design.

✓ **Monitor and Maintain Encryption**

After implementing encryption, it's essential to monitor and maintain the encryption keys, certificates, and encrypted data.

- **Backup Encryption Keys and Certificates**:

- Regularly back up your encryption keys and certificates to avoid data loss in case of server failure.
- Right-click on **Always Encrypted Keys** or **Certificates** and choose **Export...**.

**Review Encryption Configurations**:

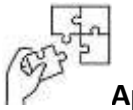
- Periodically review the encryption settings to ensure compliance with security policies.

**Rotate Keys and Certificates**:

- As a best practice, regularly rotate encryption keys and certificates to enhance security.

**Points to Remember**

- Understand that the conversion of plaintext into ciphertext is known as encryption, which is an essential aspect of database security.
- Recognize that data encryption falls into different categories, including Symmetric Encryption, Asymmetric Encryption, and Hashing.
- Use SQL Server Management Studio (SSMS) to create a master key, set up a symmetric key, and encrypt a column using the symmetric key.
- Create an asymmetric key in SSMS and use it to encrypt data.
- Create a computed column that utilizes a hashing function to ensure data integrity.
- Backup encryption keys and certificates, review encryption configurations, and regularly rotate keys and certificates to maintain security.



**Application of learning 4.3.**

As the Database Administrator for "Bright Future Academy," you need to ensure the confidentiality and integrity of sensitive data stored in the **SchoolDB** database, To protect this data from unauthorized access and breaches, you must implement data encryption techniques using SQL Server Management Studio (SSMS).

Your task is to apply encryption methods to safeguard sensitive information within **SchoolDB** and ensure that encryption keys and certificates are managed securely.

**Tasks to Perform:**

1. **Symmetric Encryption**
2. **Asymmetric Encryption**
3. **Hashing**

## Indicative content 4.4: Configuration of Database Backup and Restore

**Duration: 6 hrs**

### Theoretical Activity 4.4.1: Description of database backup and restore

**Tasks:**

1: You are asked to answer the following question related to database auditing and logging:

i. What is a database backup?
ii. What is a database restoring?

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 4.4.1.

**Key readings 4.4.1.: Description of database backup and restore**

- **Database backup**

A backup is a copy of the database or specific parts of it, created to protect data from accidental loss, corruption, hardware failure, or other issues. Backups ensure that a reliable

✓ **Types of Backups:**

**Full Back up:**

- A complete copy of the entire database.
- Advantage: Easiest to restore, as it contains everything.
- Disadvantage: Time-consuming and requires more storage.

**Incremental Backup:**

- Backs up only the data that has changed since the last backup (whether full or incremental).
- Advantage: Requires less storage and time compared to full backups.
- Disadvantage: Restoring can be more complex, as it requires applying multiple incremental backups in sequence.

**Differential Backup:**

- Backs up all changes since the last full backup (not incremental backups).
- Advantage: Faster to restore than incremental backups, since only the last full and differential backups are needed.
- Disadvantage: Larger than incremental backups as changes accumulate over time.

✓ **Backup Strategies**

- **Hot Backup (Online Backup):** Performed while the database is still running and accessible by users.
- **Cold Backup (Offline Backup):** Performed when the database is shut down and not accessible by users.
- **Warm Backup:** A combination where parts of the database may be online, but most operations are paused or limited.
- **Frequency of Backups:** Regular backups are essential, and organizations determine frequency based on factors like database size, transaction volume, and criticality of data.

✓ **Backup Storage Locations**

- **On-site:** Backups stored on the same premises as the database.
- **Off-site:** Backups stored in a different location for disaster recovery.
- **Cloud Storage:** Backups stored on cloud services for easy access and recovery.

- **Restore**

✓ **Types of Restores:**

- **Full Restore:**
  - Restores the entire database from a full backup.
  - Suitable when the entire database is damaged or lost.
- **Point-in-Time Restore:**
  - Restores the database to a specific point in time, typically by applying transaction logs after a full or incremental backup.
  - Useful when rolling back to a state just before a failure or error.
- **Incremental Restore:**
  - Restores from the last full backup, followed by applying all incremental backups in sequence.
  - Used when incremental backups were taken to minimize downtime and data loss.
- **Differential Restore:**
  - Restores the last full backup followed by the latest differential backup.
  - Faster than an incremental restore because only two backups are applied (full and differential).

✓ **Restore Process Steps:**

- **Identify the Backup:** Choose the correct backup file(s) based on the recovery point and type of failure.
- **Restore Data:** Load the backup file into the database system.

- **Apply Transaction Logs (Optional):** If necessary, apply logs to roll the database forward to a specific point in time.
- **Verify Data Integrity:** Ensure the restore was successful and that the database is functioning as expected.
- **Resume Normal Operations:** Once restored, resume normal database operations and monitor for any further issues.

✓ **Difference between backup and restore**

| Aspect | Backup | Restore |
|---|---|---|
| **Purpose** | To create a copy of the database for safety | To recover and reapply the backup data |
| **Action** | Saving the database data | Rebuilding the database from backup |
| **Timing** | Performed regularly | Done in response to data loss or corruption |
| **Data Used** | Original database data | Backup files |
| **Outcome** | Backup files are generated | Data is restored and available |

**Practical Activity 4.4.2: Configuring Database Backup and Restore**


**Task:**

1: Read key reading 4.4.2

2: Read the below task:

As the database administrator (DBA) for City General Hospital, you have been tasked with setting up a reliable backup and restore strategy for the SQL Server-based HMS database. The IT Director highlights the importance of ensuring data integrity, particularly due to regulatory compliance requirements, which mandate that medical records be protected against accidental loss and must remain accessible during critical times, such as patient treatment and billing processing.

- Use SQL Server Management Studio (SSMS) to schedule and automate the full, differential, and transaction log backups. Ensure backup job logs are maintained for auditing purposes.
- Set up SQL Server Agent alerts to monitor backup job status and configure notifications for job failures or storage issues.
- Document a comprehensive disaster recovery plan with step-by-step instructions for restoring data, including point-in-time restores and handling backup failures.

- Conduct restore tests quarterly, recording the time required for each restore.

3: Configure database backup and restore as required in the above task

4: Present your work to your trainer and a whole class

**Key readings 4.4.2: Configuring Database Backup and Restore**

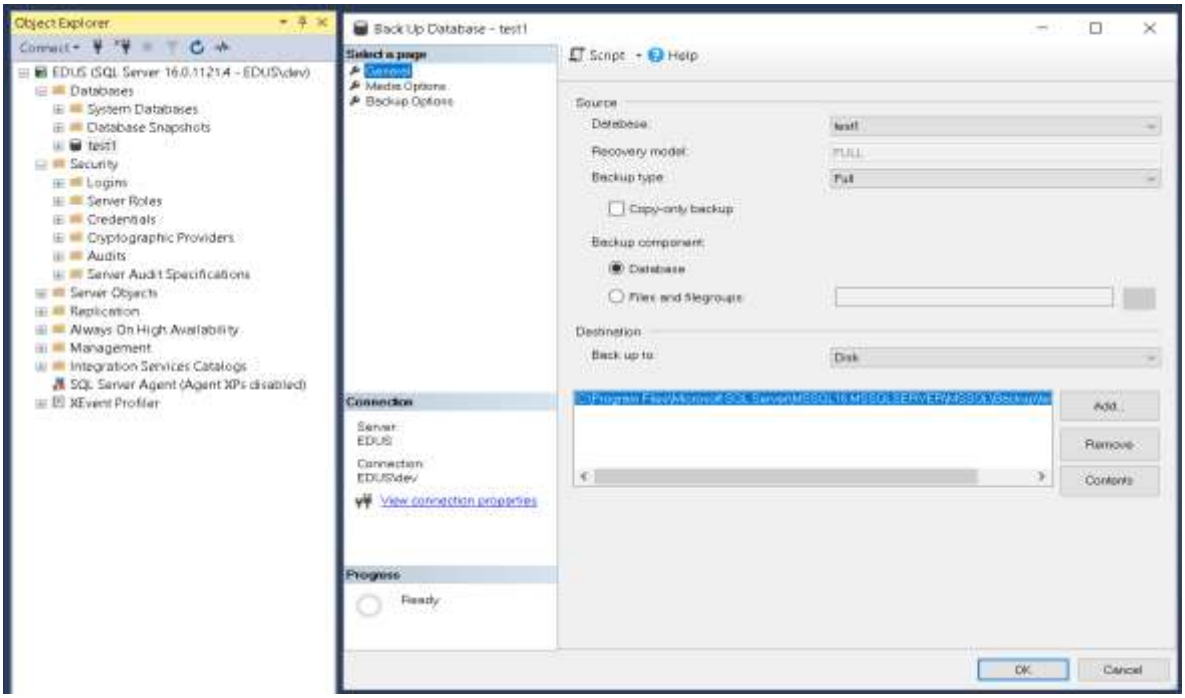
- **Configuration of Database Backup and Restore in SQL Server Using SSMS**
  - ✓ **Create Backup Using SSMS**

To create backups in SSMS, follow these steps:

**Full Backup**

- **Open SSMS** and connect to your SQL Server instance.
- In **Object Explorer**, expand the **Databases** folder and select the database you want to back up.
- Right-click on the database, hover over **Tasks**, and select **Back Up...**.
- In the **Back Up Database** window:

- Set the **Backup Type** to **Full**.
- Choose the **Backup Component** (usually **Database**).
- Specify the **Backup Destination** by clicking **Add...** and selecting a location for the backup file.




- Click **OK** to start the full backup.

**Differential Backup**

- Right-click on the database, hover over Tasks, and select Back Up....
- In the Back Up Database window:

- Set the **Backup Type** to **Differential**.
- Ensure the **Backup Destination** is set (it can be the same as the full backup).

  Click **OK** to start the differential backup.

**Transaction Log Backup (Incremental Backup)**

- Right-click on the database, hover over Tasks, and select Back Up....
- In the Back Up Database window:
- Set the Backup Type to Transaction Log.
- Ensure the Backup Destination is set.
- Click OK to start the transaction log backup.

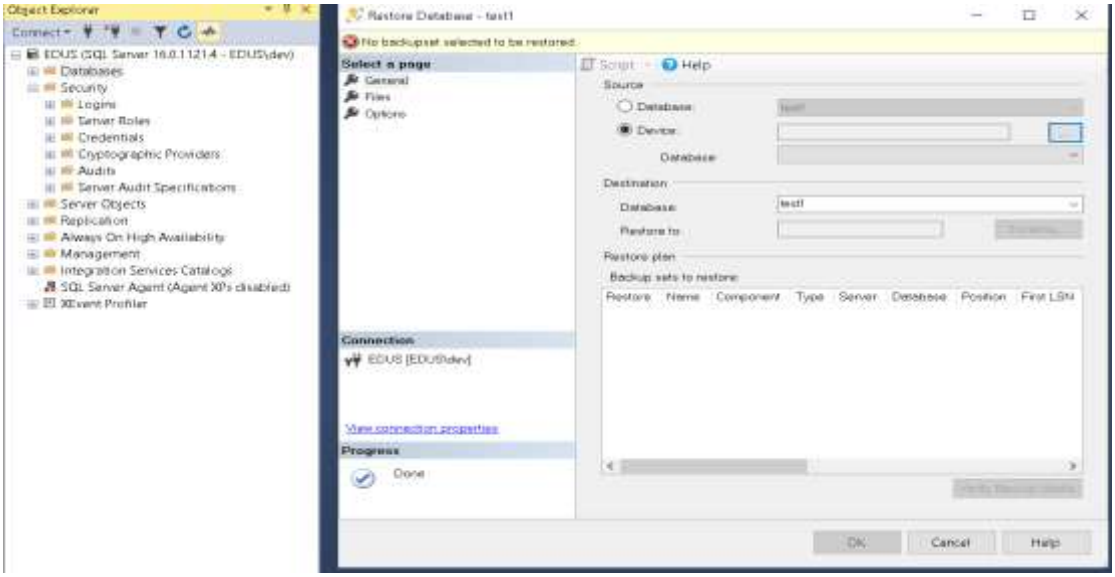
✓ **Perform Recovery Methods**

SQL Server supports several recovery methods, depending on the type of backup used and the recovery scenario:

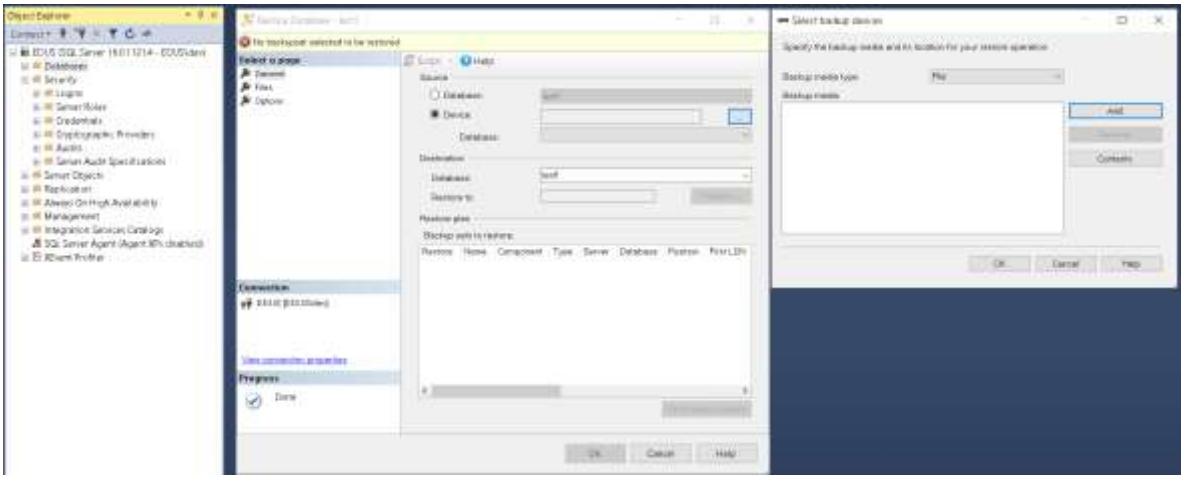
**Full Database Recovery**

To restore a full database backup:

- Right-click on the **Databases** folder and select **Restore Database...**.




- In the **Restore Database** window:
  - Select the **Source** as **Device** and choose the backup file.



- Ensure the **Restore** options are set correctly.
- Check the **Restore Options** tab to select options like **Overwrite the existing database**.

- Click **OK** to start the restoration process.

**Rollback Recovery**

Rollback recovery involves reverting the database to a specific point before unwanted transactions or errors.

- Use the **Transaction Log Restore** options in the **Restore Database** window.
- Select **Transaction Log Backups** to apply logs up to a specific point.

**Point-in-Time Recovery**

Point-in-time recovery is useful when you need to restore the database to a specific time to undo unwanted changes.

- In the **Restore Database** window, select the **Timeline** option.
- Choose **Point-in-time restore** and specify the exact date and time.
- Click **OK** to restore the database to that specific point.

✓ **Test Your Backup and Recovery Plan**

Regularly testing your backup and recovery plan is crucial to ensure it works correctly when needed.

**Test Restores**:

- Periodically restore backups to a test environment to verify they are working as expected.
- Perform full, differential, and transaction log restores to test all scenarios.

**Validate Backups**:

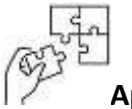
- Use the **Verify Backup Integrity** option in the **Back Up Database** window to check the validity of the backup files.

**Document Backup and Recovery Procedures**:

- Maintain documentation of backup schedules, types, and recovery steps to ensure consistent and accurate recovery in an emergency.

**Points to Remember**

- The process of creating and maintaining a copy of the database, its data, and its configuration in different locations is known as **backup**.
- Recognize that the process of retrieving data from a backup and applying it to a database to make it operational again is referred to as **restoring database data**.
- Establish a recommended backup schedule that includes weekly full backups, daily differential backups, and hourly transaction log backups for databases using the Full recovery model.
- Use SQL Server Management Studio (SSMS) to create full, differential, and transaction log backups.
- Restore databases from full backups, roll back changes using transaction log backups, and perform point-in-time recovery to revert to a specific moment in time.
- Regularly test backups and recovery plans, validate backup integrity, and document procedures to ensure consistent and accurate recovery processes.



**Application of learning 4.4.**

The Database Administrator for BrightFuture Academy, you are responsible for setting up a reliable backup and recovery strategy for the SchoolDB database, ensuring that sensitive data, protected with encryption techniques, is regularly backed up and can be restored in the event of data loss or corruption. Your tasks involve:

- Configuring various backup methods
  Propose and configure a backup schedule using SQL Server Management Studio (SSMS).
- Perform a full database recovery from a full backup, a rollback recovery to revert unwanted changes using transaction log backups, and a point-in-time recovery to restore the database to a specific time prior to unwanted modifications.
- Test the backup and recovery plan

## Learning outcome 4 end assessment

**Theoretical assessment**

**Question1: Respond True /False to the following statements:**

1. **Database security** involves only controlling physical access to the server. (T/F)
2. **Access control policies** define the conditions under which user access is allowed or denied. (T/F)
3. The **authentication system** in SQL Server is only concerned with creating user accounts. (T/F)
4. **Authorization** involves assigning roles to users and granting them specific permissions. (T/F)
5. **Hashing** is used for data encryption in SQL Server to provide confidentiality. (T/F)

**Question2: Read carefully the below statements and choose the correct answer based on the provided options (10 Marks)**

1. What is the purpose of **data access control** in a database?

    A. To monitor server performance

    B. To define roles and permissions

    C. To provide high availability

    D. To compress data files

2. Which type of **encryption** uses the same key for both encryption and decryption?

    A. Symmetric encryption

    B. Asymmetric encryption

    C. Hashing

    D. Data masking

3. What is a **full back up**?

    A. A backup of only changed data since the last backup

    B. A complete copy of the entire database

    C. A backup of log files only

    D. A backup of user accounts

4. **Auditing** helps in:

    A. Performance tuning of queries

    B. Tracking and recording database activities

    C. Creating database schemas

    D. Encrypting sensitive data

5. What does **logging** in SQL Server involve?

    A. Configuring backup schedules

    B. Monitoring and analyzing server activities

    C. Creating user accounts

    D. Performing database normalization

**Question3: Match the Columns (10 Marks)**

Match the items in Column A with their appropriate descriptions in Column B.

| **Answer:** | **Column A** | **Column B** |
|---|---|---|
| 1… | 1. Symmetric Encryption | A. Uses different keys for encryption and decryption |
| 2… | 2. Full Backup | B. Checks data integrity |
| 3… | 3. Role-Based Access Control (RBAC) | C. Complete snapshot of the database |
| 4… | 4. Auditing | D. Same key for encryption and decryption |
| 5… | 5. Point-in-Time Recovery | E. Tracks and logs events |
| 6… | 6. Incremental Backup | F. Restores to a specific point in time |
| 7… | 7. Hashing | G. Tracks only changes since the last full or incremental backup |
| 8… | 8. Authentication | H. Verifying user identities |
| 9… | 9. Logging | I. Capturing and storing system events and actions |
| 10… | 10. Discretionary Access Control (DAC) | J. Owner of the data decides who gets access |

**Question4: One-Word Response Questions (10 Marks)**

1. ……is the process of verifying a user's identity in SQL Server called?
2. Which backup type captures all the changes made since the last full backup?
3. What kind of policy allows access based on the discretion of the data owner?
4. Which method is used to ensure data integrity and is irreversible?
5. Which component in SQL Server allows you to monitor server activities and detect unauthorized access?

**Practical assessment**

As a newly hired Database Security Administrator for "Secure Fin Solutions", a medium-sized financial company, your role focuses on setting up and configuring security features for their recently implemented SQL Server database that manages sensitive customer information, transaction records, employee data, and other critical information. Using SQL Server Management Studio (SSMS), you are required to complete various tasks:

- Starting with a security overview and setup by identifying potential vulnerabilities
- Define user's roles and assigning permissions
- Establish access control policies based on these roles and classify data as Public, Confidential, or Highly Confidential.
- Create user accounts for the Finance Team and Support Team, assign roles, and test by logging in as these users to verify permissions, while regularly reviewing authentication logs for unauthorized access. Authorization will involve creating server-level roles such as DatabaseAdmin, assigning permissions at server and database levels, and testing by logging in as a DatabaseAdmin user to ensure proper functionality.
- Configure logging settings to capture events and archive log data regularly.
- Configure auditing by setting up server and database audit specifications to track key activities. Data encryption techniques will be applied.
- Configuration of database backups and test recovery methods

**END**

## References


Bertino, E. S. (2005). Database Security: Concepts, Approaches, and Challenges. Boston, Massachusetts, USA: Addison-Wesley.

Davis, R. (2013). SQL Server Security. Indianapolis, Indiana, USA: Wiley Publishing.

https://intellipaat.com/blog/database-security/
https://www.geeksforgeeks.org/database-recovery-techniques-in-dbms/
https://www.infosecurity-magazine.com/blogs/how-to-backup-and-restore-database/
https://www.splunk.com/en_us/blog/learn/audit-logs.html
Vacca, J. R. (2009). Database Security: A Practical Approach. Burlington, Massachusetts, USA: Elsevier.

RTB | RWANDA TVET BOARD

October, 2024