RQF LEVEL 4

SWDDD401

SOFTWARE DEVELOPMENT

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 Software Development, specifically for the module **"SWDDD401: 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

DUSABIMANA Emmanuel

Validation

NIYITURAGIYE Vedaste

KWIZERA Jean Paul

Conception, Adaptation and Editorial works

HATEGEKIMANA Olivier

GANZA Jean Francois Regis

HARELIMANA Wilson

NZABIRINDA Aimable

DUKUZIMANA Therese

NIYONKURU Sylvestre

MANIRAKIZA Jean de Dieu

Formatting, Graphics, Illustrations, and infographics

YEONWOO Choe

SUA Lim

SAEM Lee

SOYEON Kim

WONYEONG Jeong

MANISHIMWE Marc

Financial and Technical support

KOICA through TQUM Project

TABLE OF CONTENT

ACRONYMS

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

KOICA: Korean Cooperation International Agency

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.

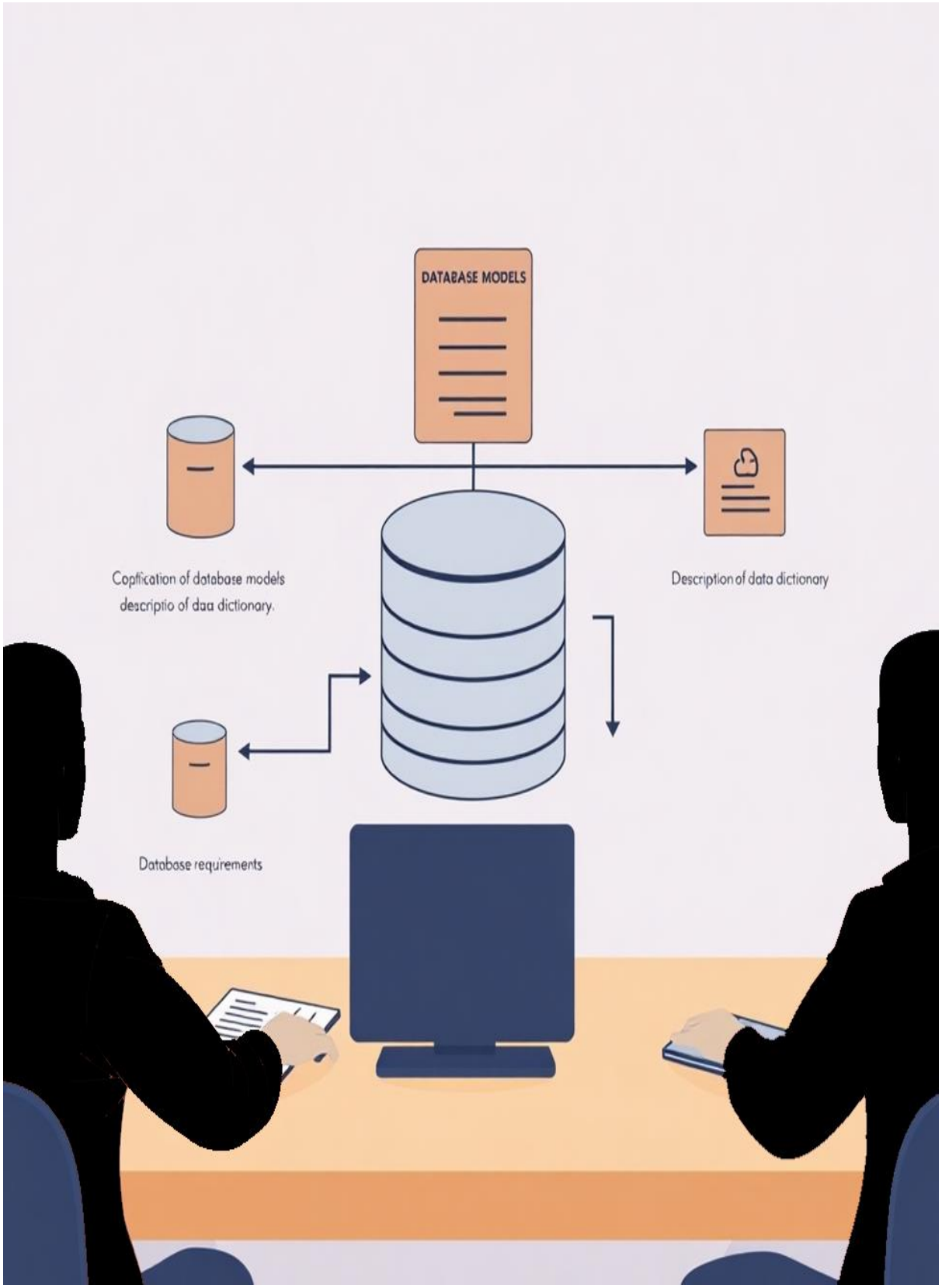
The trainee's manual is organized into Learning Outcomes, which is broken down into indicative content that includes both theoretical and practical activities. It provides detailed information on the key competencies required for each learning outcome, along with the objectives to be achieved.

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: SWDDD401 DATABASE DEVELOPMENT

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


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

Indicative contents
1.1 Description of database fundament
1.2 Description of data dictionary
1.3 Identification of database requirements

Key Competencies for Learning Outcome 1 : Analyse Database

Knowledge	Skills	Attitudes
• Description of database Concepts • Identification of database models • Identification of database relationships • Determination of database data types • Description of data dictionary • Identification of database requirements • Identification of data collection methods	• Creating data dictionary • Gathering data based on user requirement • Analysing database requirements	• Being Innovative • Having adaptability skills • Being Flexible • Having teamwork skills • Having self confidence • Being Time manager • Being Organizer

Duration:19 hrs

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. Creating properly data dictionary according to the user requirements
7. Identify properly database requirements based on user requirements
8. Analyse effectively database requirements based on user requirement

Resources

Equipment	Tools	Materials
● Computer	● MS office	● Internet

Indicative content 1.1: Description of Database Fundamental

Duration: 7 hrs

Theoretical Activity 1.1.1: Introduction to database

Tasks:

1: Answer the following questions related to the 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. Give examples of where a database is applied

iii. Give advantages and disadvantages of database

2: Present your findings to your classmates and trainer

3: Ask questions where necessary

4: For more clarification, read the key readings 1.1.1.

Key readings 1.1.1: Introduction to database

1. Definition of key terms

a. 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.

Example: websites databases, mobile apps databases, business applications databases, and more, to handle large volumes of information.

b. Data

- **Definition:**

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

- **Characteristics:**

  Raw: Data is unprocessed and unorganized.

  Objective: Data is neutral and does not carry any meaning.

  Individual Pieces: Data can be individual pieces of information.

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

c. Information

- **Definition:**

  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.

- **Characteristics**:

  Processed: Information is processed data that has been analyzed or organized.
  Meaningful: Information provides context and understanding.
  Useful: Information helps in decision-making, problem-solving, or gaining knowledge.

- **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.

d. 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

e. 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 (**Customer ID, Name, Email, Phone Number)

➔ **Product entity may have attributes like: Product (**Product ID, Name, Description)

f. 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

g. 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

Customer table

h. 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.

i. 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.

1. MySQL
2. PostgreSQL
3. Oracle
4. Microsoft SQL Server
5. MongoDB
6. Cassandra
7. Redis
8. SQLite
9. IBM DB2
10. Amazon AuroraSQL

j. SQL

Structured query language (SQL) is a programming language for storing and processing information in a relational database.

You can use SQL statements to store, update, remove, search, and retrieve information from the database. You can also use SQL to maintain and optimize database performance.

2. Applications of database

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

1. **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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

3. Advantages and Disadvantages of Database

3.1. Advantages of Databases:

1. Data Centralization: Databases provide a centralized and structured repository for storing and managing data. This allows for efficient data organization, easy access, and improved data integrity.

2. Data Consistency: Databases enforce data integrity rules, ensuring that data remains consistent and accurate. This helps to prevent data duplication and inconsistencies that can occur in file-based systems.

3. Data Security: Databases offer robust security features, such as user authentication, access control, and encryption. This helps protect sensitive data from unauthorized access and ensures data privacy.

4. Data Sharing and Collaboration: Databases enable multiple users to access and work with the same data simultaneously. This promotes collaboration, data sharing, and concurrent data processing, leading to increased productivity and efficiency.

5. Data Integrity and Reliability: Databases provide mechanisms for data backup, recovery, and transaction management. This ensures that data remains intact and reliable, even in the event of system failures or errors.

3.2. Disadvantages of Databases:

1. Complexity: Designing and managing a database system can be complex and require specialized skills. It may involve understanding data modeling, database languages, and performance optimization techniques.

2. Cost: Implementing and maintaining a database system can be costly. It involves expenses for hardware, software licenses, database administration, and ongoing maintenance.

3. Performance Overhead: Database systems introduce some performance overhead due to the additional layers of abstraction and data management processes. Poorly designed databases or inefficient queries can lead to performance bottlenecks.

4. Scalability Challenges: Scaling a database system to handle increasing amounts of data and user load can be challenging. It may require careful planning, performance tuning, and potentially costly hardware upgrades.

5. Data Dependencies: In a database system, changes to the structure or schema can have implications on existing applications and data dependencies. This can require careful management and coordination to ensure compatibility and minimize disruptions.

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 models

2: Present your answers to the whole class

3: Ask questions where necessary.

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

Key readings 1.1.2.: Identification of database models

1. Definition of database model

A database model defines the logical structure of a database, determining how data is stored, organized, and accessed.

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

Here are four major database models:

2. Types of Database models

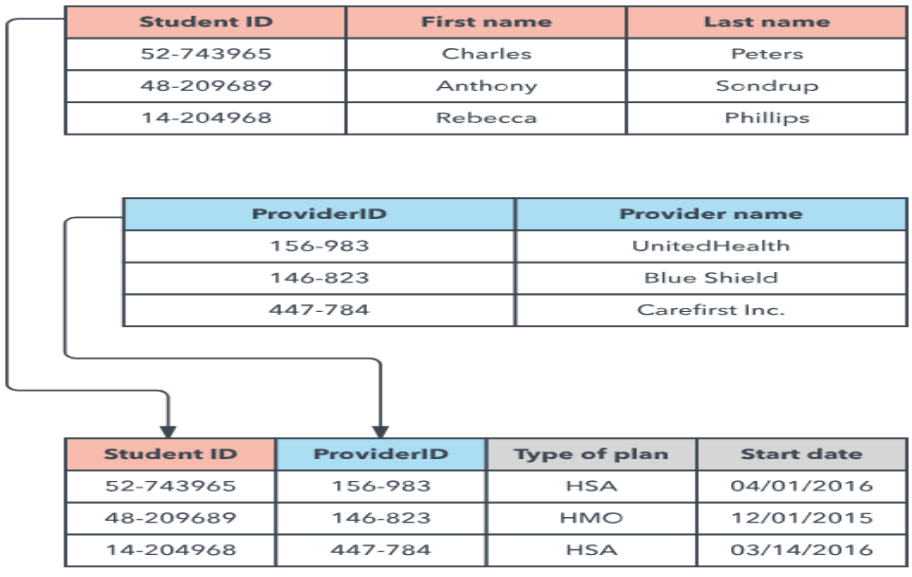
a. Relational Model

The most common model, the relational model store data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity in question, such as price, zip code, or birth date. Together, the attributes in a relation are called a domain. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, when it's called a foreign key.

Each row, also called a tuple, includes data about a specific instance of the entity in question, such as a particular employee.

The model also accounts for the types of relationships between those tables, including one-to-one, one-to-many, and many-to-many relationships.

Here's an example:



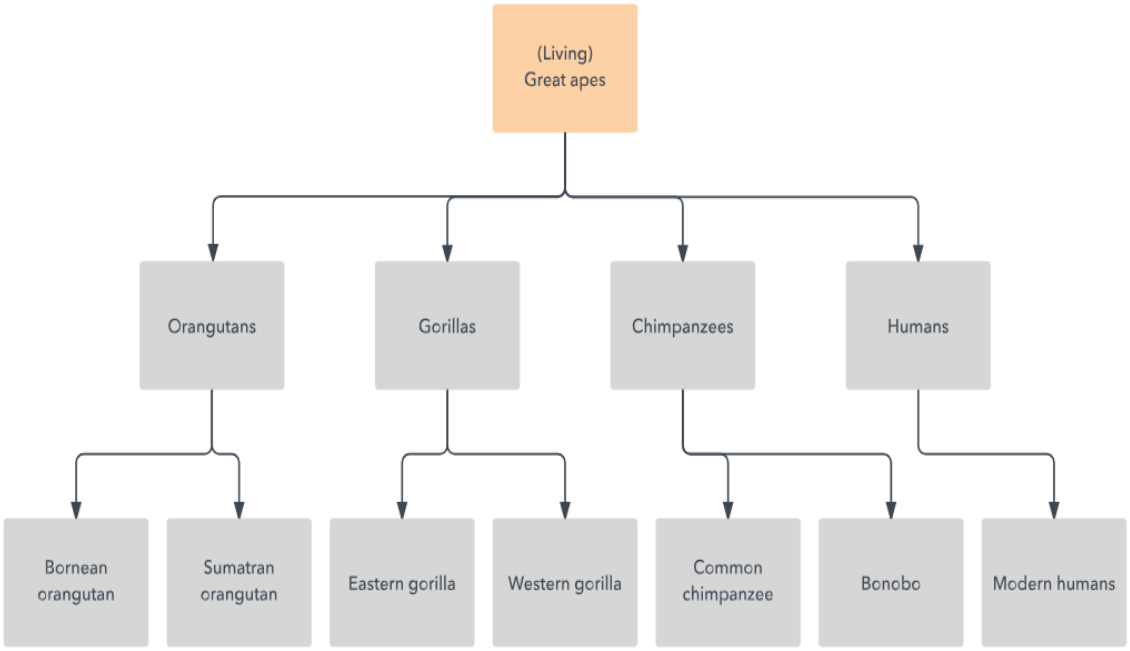
Student ID	First name	Last name
52-743965	Charles	Peters
48-209689	Anthony	Sondrup
14-204968	Rebecca	Phillips

ProviderID	Provider name
156-983	UnitedHealth
146-823	Blue Shield
447-784	Carefirst Inc.

Student ID	ProviderID	Type of plan	Start date
52-743965	156-983	HSA	04/01/2016
48-209689	146-823	HMO	12/01/2015
14-204968	447-784	HSA	03/14/2016

b. Hierarchical Model

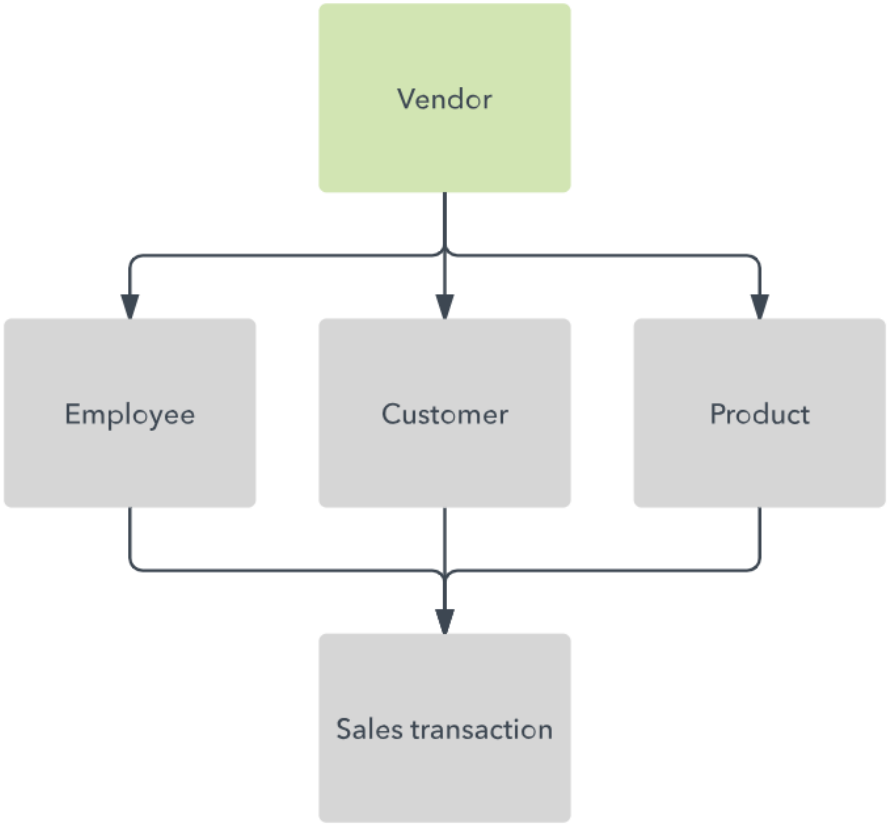
The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root. Sibling records are sorted in a particular order. That order is used as the physical order for storing the database. This model is good for describing many real-world relationships.



This model was primarily used by IBM's Information Management Systems in the 60s and 70s, but they are rarely seen today due to certain operational inefficiencies.

c. Network Model

The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records. Based on mathematical set theory, the model is constructed with sets of related records. Each set consists of one owner or parent record and one or more member or child records. A record can be a member or child in multiple sets, allowing this model to convey complex relationships.



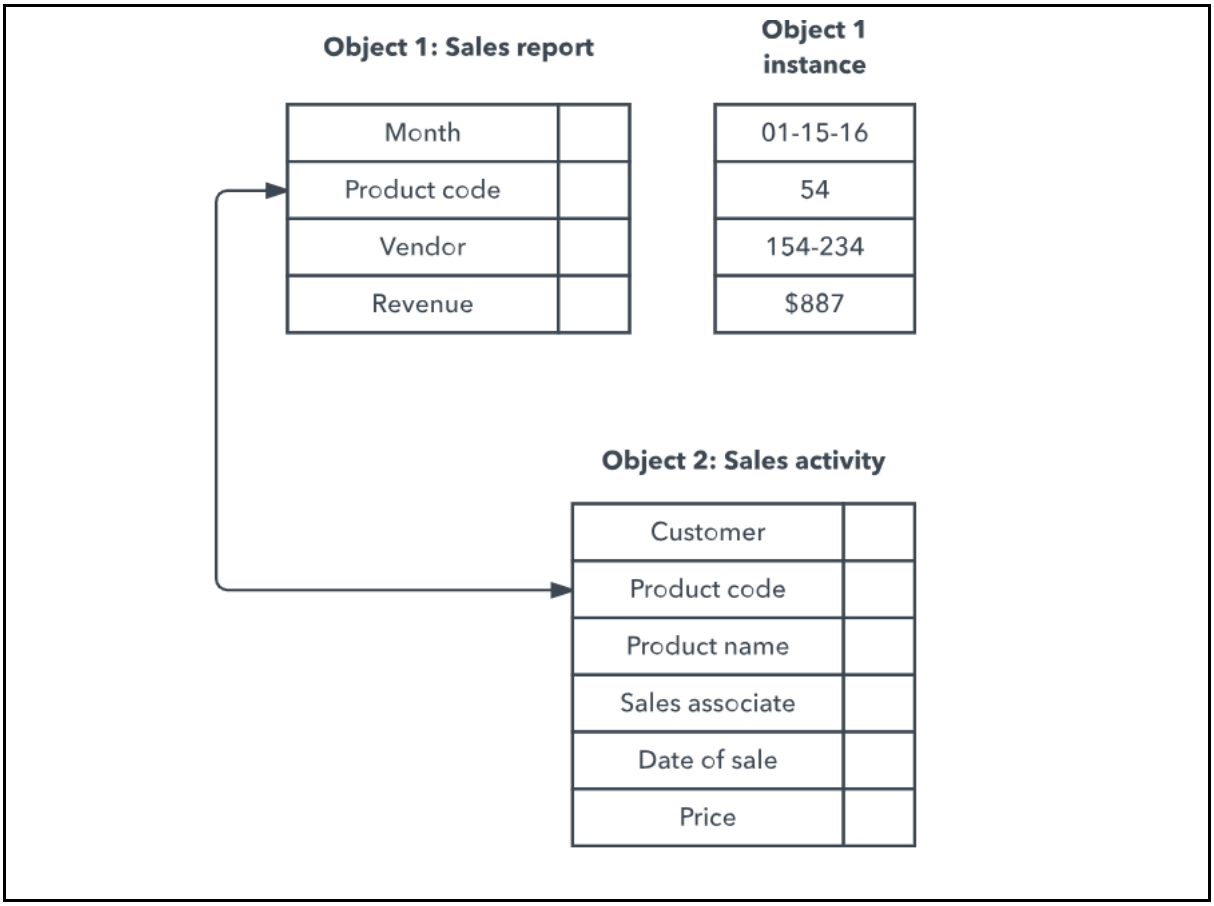
It was most popular in the 70s after it was formally defined by the Conference on Data Systems Languages (CODASYL).

d. Object-Oriented Model

This model defines a database as a collection of objects, or reusable software elements, with associated features and methods. There are several kinds of object-oriented databases:

- **A multimedia database** incorporates media, such as images, that could not be stored in a relational database.
- **A hypertext database** allows any object to link to any other object. It's useful for organizing lots of disparate data, but it's not ideal for numerical analysis.

The object-oriented database model is the best known post-relational database model, since it incorporates tables, but isn't limited to tables. Such models are also known as hybrid database models.



Theoretical Activity 1.1.3: Identification of database Relationships

Tasks:

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

i. What do you understand by relationship in database?

ii. Identify types of relationships

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

1. Definition of Relationship

Relationship is an association between entities.

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:

2. Types of relationship

1. 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.

2. 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.

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

4. 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: Determination of data types

Tasks:

1: Answer the following questions:

i. What do you understand by datatype?

ii. Explain main categories of datatype?

2: Present the findings to the whole class

3: Ask questions where necessary.

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

Key readings 1.1.4.: Determination of data types

1. **Definition of Datatype**

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

2. **Categories of datatypes**

   a. **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
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data

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

b. 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
DOUBLE PRECISION(*size*, *d*)	double precision refers to a data type used to store floating-point numbers with double precision, which means higher accuracy and a larger range of values compared to single-precision types like FLOAT. Double precision numbers are typically used when calculations require greater precision.
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.

c. 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.

Example:

Determine the datatype and size for each column in the below student table:

REGNO	**LAST_ NAME**	**FIRST_ NAME**	**MODULE**	**S E X**	**Yearofbirth**	**MARKS**
S002	Mutesa	Gad	SFTDB401	M	01	70
S023	Berwa	Hope	SFTDP401	F	09	85
S012	Minani	Charles	SFTPF401	M	99	74
S034	Mukesha	Dyna	SFTWD401	F	00	81
S025	Teta	Betty	SFTWA401	F	04	74
S041	Keza	Alice	SFTIP401	F	07	90

Student(REGNO varchar(4), LAST_NAME varchar(12), FIST_NAME varchar(8), MODULE varchar(6), SEX vrchar(1), YEAROFBIRTH year, MRKS int(2))

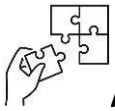
Points to Remember

- **A database** is a structured collection of data that is organized and stored in a way that can be easily accessed, managed, and updated.
- **Data are raw** facts materials that needs to be processed in order to have meaning while **information** is data after being processed
- **Entity:** Is something that exist distinguishable from others
- **Attributes/Field:** Properties of Entity
- **Record:** Raw of information
- **Table:** A table is a data structure used to organize and store data in a relational database management system (RDBMS)
- **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.
- **DBMS**: A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner.
- **SQL:** Structured query language (SQL) is a programming language for storing and processing information in a relational database.
- Database can be applied in Business, E-commerce, Banking and Finance, Healthcare, Education, …
- To store data in database, offer more advantages like:
  - ✓ Data become Centralized
  - ✓ Store consistency data
  - ✓ Secure data
  - ✓ facilitate Sharing and Collaboration
  - ✓ Data Integrity and Reliability

  But also, there are disadvantages like:
  - ✓ Complexity
  - ✓ Cost
  - ✓ Performance Overhead
  - ✓ Scalability Challenges
- A database model defines the logical structure of a database, determining how data is stored, organized, and accessed.
- Database models are:
  - ✓ Relational Model
  - ✓ Hierarchical Model
  - ✓ Network Model
  - ✓ Object-Oriented Model
- Relationship is an association between entities
- Types of relationship in database are:

- ✓ One to one
- ✓ One to many
- ✓ Many to one
- ✓ Many to many

- It is a keyword used in database to identify type of data a column will store
- Categories of data types are:
  - ✓ Characters
  - ✓ Numbers
  - ✓ Dates



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 identify entities, attribute for each entity, datatype for each attribute and identify the relationship between each pair of entities

Indicative content 1.2: Description of Data Dictionary

Duration: 7hrs

Theoretical Activity 1.2.1: Description of data dictionary

Tasks:

1: You are requested to 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

1. Definition of 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.

2. Elements of data dictionary

a. **Field Name:** Specifies the name of the column in the table.
b. **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).
c. **Length:** Represents the maximum length of the data element, applicable to string data types.
d. **Constraints:** Describes any constraints applied to the column (e.g., Primary Key, Foreign Key, Not Null, and Check).
e. **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.

3. Types of data dictionary

a) Active Data Dictionary

♦ **Definition:** An active data dictionary is automatically managed by the database management system (DBMS). It is integrated into the DBMS and is updated

automatically whenever changes are made to the database schema (e.g., adding tables, altering columns).

- **Characteristics**

— Directly tied to the database.
— Reflects the current state of the database schema.
— Used by the DBMS for tasks like query optimization and integrity checking.
— Ensures that all changes made to the database are immediately reflected in the dictionary.

- **Example:** In SQL-based databases, when a table or column is created, the DBMS automatically updates its internal metadata to reflect the change.

b) Passive Data Dictionary

- **Definition:** A passive data dictionary is maintained manually, and changes to the database schema are not automatically reflected. It is external to the DBMS and must be updated manually whenever changes are made.
- **Characteristics**

— Not directly integrated with the DBMS.
— Requires human intervention to update the dictionary when the database structure changes.
— Can become outdated if not regularly maintained.
— Used for documentation and reference purposes, rather than for internal database operations.

- **Example:** A separate document, spreadsheet, or third-party tool that stores database metadata but requires manual updates.

4. Sample passive data dictionary

Table	Field Name	Data Type	Length	Constraints	Description
Employees	EmployeeID	INT	4	Primary Key	Unique identifier for each employee
	FirstName	VARCHAR	50	Not Null	First name of the employee
	LastName	VARCHAR	50	Not Null	Last name of the employee
	BirthDate	DATE	-	Not Null	Date of birth of the employee
	Gender	CHAR	1	Not Null	Gender of the employee (M/F)
	IsActive	BOOLEAN	1	Not Null	Indicates if the employee is currently active (1 for active, 0 for inactive)

Practical Activity 1.2.2: Creating a data dictionary

Task:

1: Read key reading 1.2.2

2: Read carefully the following task:

> The library has various sections such as Fiction, Non-Fiction, Reference, and Children's Books, each containing thousands of books, for that reason to manage the books it is difficult because their information is stored in Microsoft excel. The library needs a system to efficiently manage its inventory of books, track member borrowing history, and handle day-to-day operations. The system will hold information of books: Each book has specific attributes, including title, author, and publication year, along with availability status. Additionally, the library needs to keep track of members, including their personal details, membership status, and borrowing history. To ensure that the database is well-structured and can meet the library's operational needs, you are asked to prepare a comprehensive data dictionary.

3: Referring to the key reading 1.2.2 consider the provided task above to create a data dictionary

4: Present your work to the trainer and whole class

Key readings 1.2.2Steps to prepare a Data Dictionary

1. Identify the Entities

Description: Determine the main entities (or tables) that will be part of the database. These entities represent objects or concepts relevant to the system you are designing, such as Customers, Orders, or Products.

Example: For a library system, entities could include Books, Members, and Borrowing Transactions.

2. Define Attributes for Each Entity

Description: List all the attributes (or columns) for each entity. For each attribute, define what type of data it will hold, such as Integer, Varchar, Date, etc.

Example: For the Books entity, attributes might include BookID, Title, Author, PublicationYear, Genre, and ISBN.

3. Assign Data Types

Description: For each attribute, assign the appropriate data type that defines the kind of data it will store. Common data types include Integer, Varchar, Date, Boolean, etc.

Example: The PublicationYear attribute might be assigned the Year data type,

while Title might be Varchar(255).

4. Specify Constraints

Description: Define any constraints for the attributes, such as Primary Key, Foreign Key, Not Null, Unique, Default, etc. These constraints enforce rules on the data to maintain integrity and accuracy.

Example: The BookID attribute in the Books entity would typically be a Primary Key and Auto-increment, while ISBN might have a Unique constraint.

5. Establish Relationships Between Entities

Description: Identify and define the relationships between different entities. This includes specifying foreign keys and the type of relationships (one-to-one, one-to-many, many-to-many).

Example: The Borrowing Transactions entity might have a foreign key relationship with the Books and Members entities.

6. Create a Data Dictionary Table

Description: Organize all the information into a table format that is easy to understand. Each row should represent an attribute, and the columns should include details like the attribute name, data type, constraints, and a description.

Points to Remember

- Data dictionary is a document that hold data to describe database
- Elements of data dictionary are:

  Table name

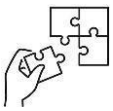
  Attributes name

  Data type and length

  Constraints

  Descriptions

- **To create a data dictionary, follow the below steps:**
  - ✓ Identify the Entities
  - ✓ Define Attributes for Each Entity
  - ✓ Assign Data Types to attributes
  - ✓ Specify Constraints for each attribute
  - ✓ Establish Relationships Between Entities
  - ✓ Create a Data Dictionary Table



Application of learning 1.2

NST is a university that need to expand its digital infrastructure and needs to develop a comprehensive database system to manage its student information, course offerings, faculty details, and enrollment records. The university administration wants to ensure that all aspects of student and course management are streamlined, including tracking student profiles, course registrations, grades, and faculty assignments. To achieve this, you, as the database developer, are tasked with creating a detailed data dictionary that will serve as the foundation for the database design. This data dictionary should define all the entities involved, such as Students, Courses, Faculty, and Enrollments, along with their respective attributes, data types, and relationships. Additionally, it should specify any constraints, such as primary and foreign keys, to ensure data integrity and consistency across the database. By preparing this data dictionary, you will provide a clear and organized blueprint that guides the development process, ensuring that the database meets the university's operational and academic requirements.

Indicative content 1.3: Identification of Database Requirements

Duration: 6 hrs

Theoretical Activity 1.3.1: Identification of data collection methods

Tasks:

1: You are requested to answer the following questions related to the data collection methods:

1. What is data collection?
2. Identify data collection methods?

2: Present your answers to your classmate and your trainer

3: Ask questions where necessary.

4: For more clarification, read the key readings 1.3.1

Key readings 1.3.1: Identification of data collection methods

1. Definition of Data collection

Data collection is the process of systematically gathering information from various sources to gain insights, make informed decisions, and answer research or business-related questions.

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.

2. Methods to collect data

a. Interview

Interview: Interviews in software development 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 interviews**

1) **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.

2) **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.
3) **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.

b. Documentation

It involves gathering by 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.

c. 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.

Here 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.

d. 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.

Practical Activity 1.3.2: Collecting data

Task:

1: Read key reading 1.3.2

2: Read carefully the below task:

In your role as a database analyst, you have been tasked with gathering essential information for the development of a website for Bright Future Secondary School. This project aims to create a platform that will make important school information accessible to students, parents, and stakeholders. To ensure the database is accurately designed and captures all necessary data, you are required to collect detailed information on student records, parent contacts, academic performance, staff details, and other relevant data points such as school events and notices. You will need to engage with various stakeholders, including school administrators, teachers, and parents, utilizing methods such as interviews, questionnaires, observation checklists, and online forms to ensure comprehensive data collection. The goal is to create a database that can be easily integrated into the website and meets the functional and reporting requirements of the school.

3: Referring to the key reading 1.3.2, Collect data as requested in the above task

4: Present your work to the trainer.

Key readings 1.3.2 Collecting data

1.Collecting data by using an interview

To collect raw data by using an interview pass throughout the below steps:

1) **Preparation:**

- Review the research objectives and familiarize yourself with the interview guide or questions.
- Ensure that all necessary materials, such as consent forms and recording devices, are ready.
- Choose a suitable location for the interview, considering factors like privacy and minimal distractions.

2) Introduction:

— Begin the interview by introducing yourself and thanking the interviewee for their time.

— Establish a friendly and comfortable atmosphere to make the interviewee feel at ease.

3) Build Rapport:

— Establish a comfortable and friendly atmosphere to encourage open communication.

— Engage in small talk to help the participant feel at ease before delving into the main questions.

4) Consent and Permissions:

— If required, obtain informed consent from the participant to use their responses for research purposes.

— Explain how the data will be used, stored, and anonymized, and address any concerns the participant may have.

5) Review Interview Process:

— Briefly explain the structure of the interview, the approximate duration, and the types of questions you will be asking.

6) Follow the Interview Script:

— Use the interview tool as a guide, following the questions and prompts in the predetermined order.

— Encourage participants to provide detailed responses, and use follow-up questions when necessary.

7) Active Listening:

— Practice active listening throughout the interview, demonstrating attentiveness to the participant's responses.

— Use non-verbal cues, such as nodding, to show understanding and engagement.

8) Probing and Clarification:

— When needed, use probing techniques to encourage participants to elaborate or clarify their responses.

— Seek additional information to ensure a comprehensive understanding of the participant's perspective.

9) Record Responses:

— Record the participant's responses accurately and consistently. This may involve taking notes, audio recording, or using digital tools for transcription.

10) Handle Technical Issues:

— If using technology for recording or note-taking, ensure that it is functioning properly. Be prepared to address any technical issues that may arise during the interview.

11) Respect Participant's Pace:

- Allow the participant to answer at their own pace. Avoid rushing or interrupting, and create a space for thoughtful responses.

12) Address Participant Questions:

- Be prepared to answer any questions the participant may have about the interview process or the research itself.

13) Closing the Interview:

- Thank the participant for their time and contribution.
- Reiterate the confidentiality of their responses and provide any relevant information about the next steps in the research process.

2. Steps to collect data using the documentation method

1) Identify the Objectives

- Clearly define the purpose of the data collection.

2) Determine Relevant Documents

- Identify the types of documents that will provide the necessary information.

3) Obtain Access to Documents

- Ensure you have permission to view, analyze, and extract necessary data from official records.

4) Review and Analyze the Documents

- Systematically go through the documents and note key details relevant to the project.
- Look for patterns, gaps, or data inconsistencies in the documents

5) Extract and Organize Data

- Extract relevant data
- Organize the extracted information into categories

6) Cross-Verify Data

- Cross-check the data with other available documentation or records to ensure accuracy and consistency.

7) Document Findings

- Maintain detailed records of the findings, ensuring all relevant data is captured.
- Use appropriate formats to document the findings (e.g., tables, summaries, reports).

3. Collecting Raw data by using questionnaire

To collect raw data by using questionnaire flow the below steps:

1) Prepare for Distribution:

- If distributing online, use a reliable survey platform (e.g., Google Forms, SurveyMonkey) or prepare printed copies for in-person distribution.
- Ensure you have a plan for reaching your target audience, whether it's through email, social media, or other channels.

2) Distribution:

- Distribute the questionnaires to your target audience.
- Clearly communicate instructions for completing the questionnaire.
- If using online platforms, share the survey link and consider using incentives to encourage participation.

3) Monitor Responses:

- Regularly check for incoming responses and monitor for any issues or incomplete submissions.
- Consider sending reminders to boost response rates if necessary.

4. Collecting raw data by using observation method

1) Prepare for Data Collection:

- Ensure that all necessary materials for data collection are ready. Check that observation tools are functioning properly and that observers are prepared.

2) Start Observation:

- Begin the observation according to your plan. Record observations systematically and consistently. Be attentive to details and record data promptly.

3) Conduct the Observation:

- Carry out the observation according to your plan. Be adaptable to unexpected situations while staying focused on the objectives. Minimize interference with the natural flow of the observed activity.

4) Maintain Objectivity and Limit Bias:

- Remind observers to maintain objectivity and minimize bias. Objectivity ensures that the data collected accurately reflects the observed behavior or event.

Theoretical Activity 1.3.3.: Identification of database requirements

Tasks:

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

1. What do you understand by database requirements?
2. Differentiate Functional requirements from non-functional requirements?

2: Present your answers to your classmates

3: Ask questions where necessary

4: For more clarification, read the key readings 1.3.1.

Key readings 1.3.3: Identification of database requirements

1. Database requirements

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

2. Types of database requirements

2.1. 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.

2.2. 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.

Practical Activity 1.3.4: Analysing database requirement

Task:

1: Read keys 1.3.4

2: Read carefully the below task:

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.

3: Refer to the key reading 1.3.4, consider the above task and analyse database requirements

4: Present your work to the whole classroom

Key readings 1.3.4 Steps to analyse database requirements.

1. **Define the System's Purpose:** Clearly articulate the goals and objectives of the database system.
2. **Identify Stakeholders:** Determine who will use the system and what their needs are.
3. **Gather Information:** Collect data about the system's functionalities, processes, and data elements.
4. **Define Functional Requirements:** Specify the actions the system must perform.
5. **Define Non-Functional Requirements:** Specify the system's qualities and characteristics.
6. **Prioritize Requirements:** Rank requirements based on importance and feasibility.
7. **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.

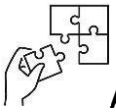
- **Security and privacy:** Protect sensitive information.
- **Performance optimization:** Consider factors affecting system speed and responsiveness.
- **Cost-effectiveness:** Balance requirements with budget constraints.
- **Maintainability:** Design for ease of modification and updates.

Points to Remember

- Data collection is the process of systematically gathering information from various sources to gain insights, make informed decisions, and answer research or business-related questions.
- Data collection methods are:
  - ✓ **Interview:** is conversation between interviewee and interviewer
  - ✓ **Documentation:** It involves gathering by analyzing existing documentation to understand the system architecture, requirements, design decisions, or codebase.
  - ✓ **Questionnaire:** A questionnaire is a structured set of questions designed to collect specific information from respondents.
  - ✓ **Observation:** Observation is a data collection technique where researchers or developers directly observe and analyze how people or processes behave in real-life situations.
- Collecting data using interview the below steps are used:
  - ✓ Introduce
  - ✓ Follow the Interview Script
  - ✓ Ask questions
  - ✓ Probing and Clarification
  - ✓ Record Responses
  - ✓ Address Participant Questions
  - ✓ Closing the Interview
- Steps to collect data by using documentation method
  - ✓ Determine Relevant Documents
  - ✓ Obtain Access to Documents
  - ✓ Review and Analyze the Documents
  - ✓ Extract and Organize Data
  - ✓ Cross-Verify Data
  - ✓ Document Findings
- To collect data by using questionnaire follow the below steps:
  - ✓ Prepare for Distribution
  - ✓ Distribution

- ✓ Monitor Responses
- To collect data by using observation the below steps are used:
  - ✓ Conduct the Observation
  - ✓ Maintain Objectivity
  - ✓ Record data
- Database requirements are the detailed specifications and criteria that a database system must meet to fulfil the needs of a particular application, organization, or project.
- Functional requirements define the specific actions and operations the database system must perform. While Non-functional requirements describe the qualities and characteristics of a database system, rather than its specific functions.
- Steps to analyse database requirements

✓ Define the System's Purpose
✓ Identify Stakeholders
✓ Gather Information
✓ Define Functional Requirements
✓ Define Non-Functional Requirements:
✓ Prioritize Requirements
✓ Validate Requirements



Application of learning 1.3

TechLearn Academy, an online educational platform, is looking to develop a database system to manage its courses, students, instructors, and learning resources. As the database analyst, your task is to collect data and analyze both the functional and non-functional requirements for this new system.

Learning outcome 1 end assessment

Written assessment

Question1: Match the database terms in column A with their definitions 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) Data after being processed
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.

Question2: Choose the collect 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. Boolean

Question3: Identify the type of relationship below between each pair of entities from the below provided statements:

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.

Question4: Match the data collection methods (in the second column) to their description (in the third column) in the below table and write answers in the column first column of a table.

Answers	Methods	Description
…..	1. Interview	A. Reviewing existing records, files, reports, or other documentation to gather relevant data.
…..	2. Documentation	B. Using a set of predefined questions, often distributed to a large number of people, to collect structured responses.
…..	3. Questionnaire	C. Directly watching or monitoring individuals or processes to gather data.
…..	4. Observation	D. Engaging in face-to-face or virtual conversations to obtain detailed information from participants.

Question5: Which of the following is *NOT* typically included as an element in a data dictionary?

A. Data Type

B. Attribute Description

C. Data Owner

D. Relationship Cardinality

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. **Database Fundamentals Analysis** (Identify the key database terms involved in this scenario, Determine the applications of the HIS database, Analyze the advantages and disadvantages of 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. **Create a data dictionary** for the HIS, defining the entities, attributes, and data types to be used in the database.

References

Books

Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, 2020. Database System Concepts (7th Edition)

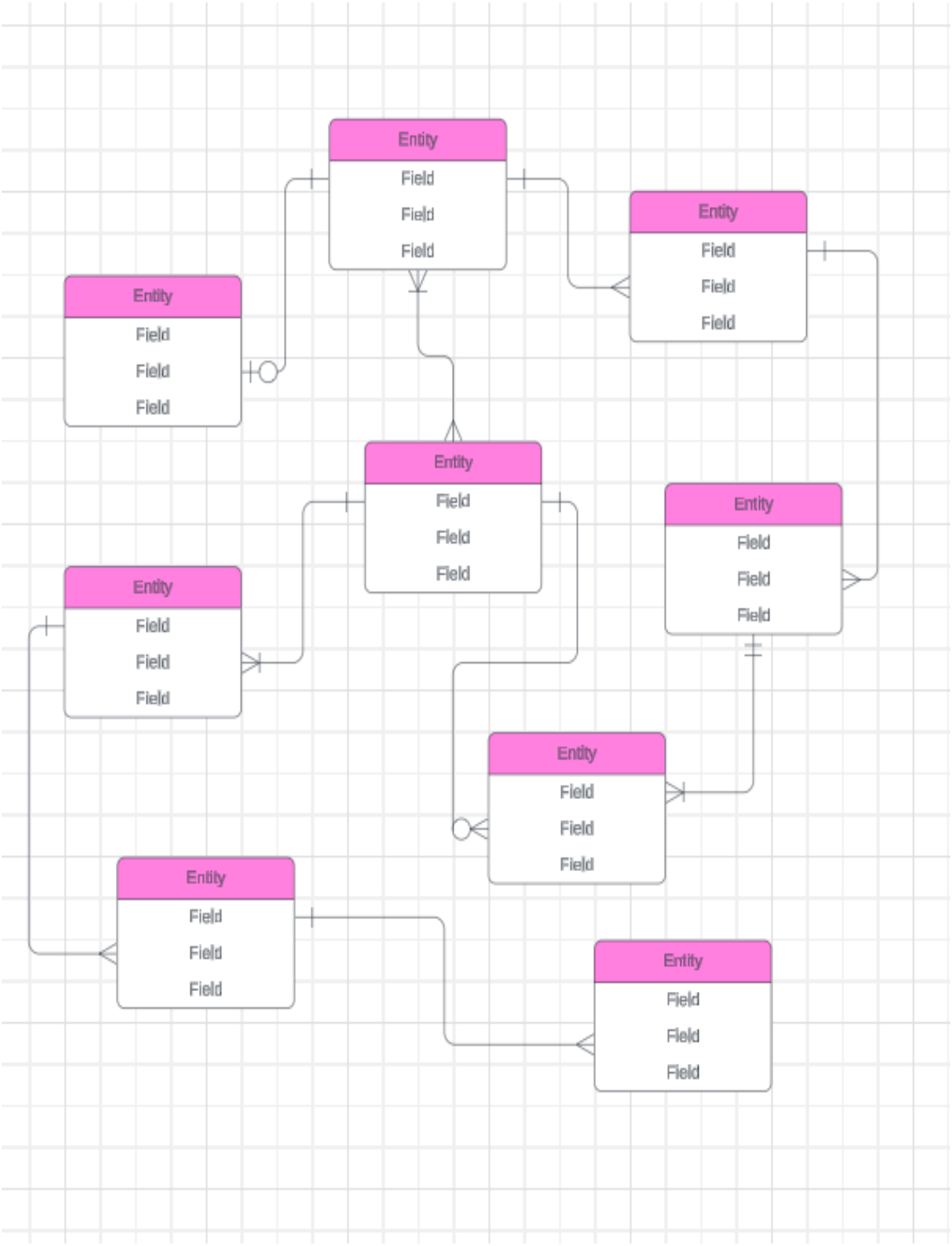
Michael J. Hernandez 2021, Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design (4th Edition)

Steve Hoberman, 2011. Data Modeling Made Simple: A Practical Guide for Business and IT Professionals (2nd Edition)

Web links

LLC, D. (2024, 10 30). RelationalDBDesign. Retrieved from www.relationaldbdesign.com https://www.relationaldbdesign.com/database-analysis/module1/intro-relational-data-analysis.php

Study.com. (2024, 10 30). What is an Entity in a Database? Retrieved from Study.com: https://study.com/academy/lesson/what-is-an-entity-in-a-database.html


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
Entity
Field
Field
Field

Indicative contents

2.1 Description of database schema

2.2 Design of conceptual database schema

2.3 Design of logical database schema

2.4 Optimization of database

2.5 Design of Physical database schema

Key Competencies for Learning Outcome 2: Design Database

Knowledge	Skills	Attitudes
• Description of database schema • Description of conceptual database schema • Description of logical database schema • Description of physical database schema	• Designing conceptual database schema • Designing logical database schema • Optimizing database • Designing Physical database schema	• Having creativity skills • Being Innovative • Having adaptability skills • Being Flexible • Having teamwork skills • Having self confidence • Being Time manager • Being Organizer

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. Optimise effectively Database based on database schema
6. Create appropriately Physical Database Schema based on the Physical Data Model

Resources

Equipment	Tools	Materials
● Computer	● E-Draw max ● MySQL set up	● Internet

Indicative content 2.1: Description of Database Schema

Duration: 2hrs

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

1. 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.

2. 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.

a) **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

b) **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 Conceptual database schema are:

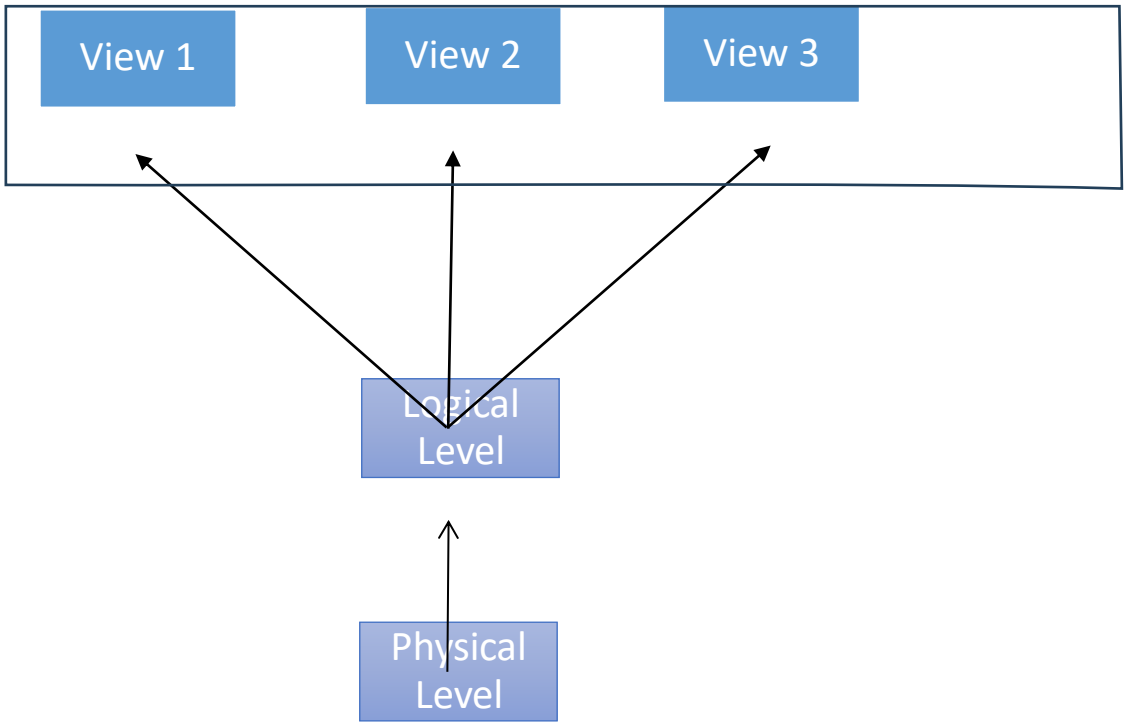
- ✓ Entity Names
- ✓ Entity Relationships
- ✓ Attributes
- ✓ Primary Keys
- ✓ Foreign Keys

c) **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.

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

3. 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:



a) **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.

b) **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.

c) **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.

4. 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.

a) 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.

b) 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 schemas are:

  Conceptual database schema

  Logical database schema

  Physical database schema
- Database abstraction levels are:

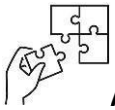
  Physical Level

  Logical Level

  View Level
- Types of data independence:

  Logical Data Independence

  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 enrollment. 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, Explaining the Levels of simplifying the database structure, then advise the university if it is possible to expand the structure of database after its implementation.

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

1. 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.

2. 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 behavior 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.

3. Components of ERD

3.1. 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**

1) 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.

2) 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.

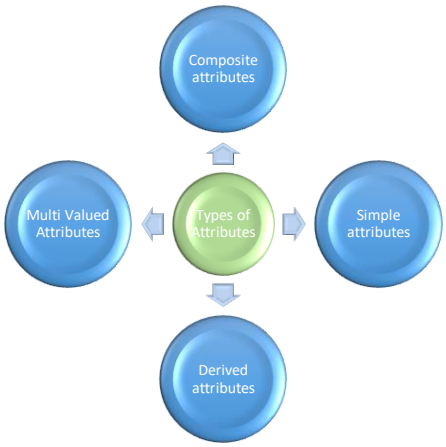
A typical example of a weak entity type is an "order." It has no meaning by itself

- 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.

3.2. Attributes

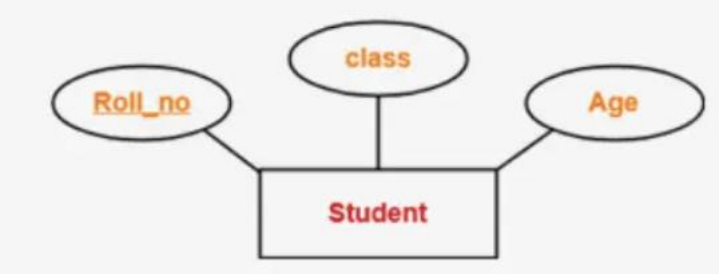
An attribute is a piece of data that describes an entity. For example, in a customer database, the attributes might be name, address, and phone number.

- **Types of Attributes in DBMS**



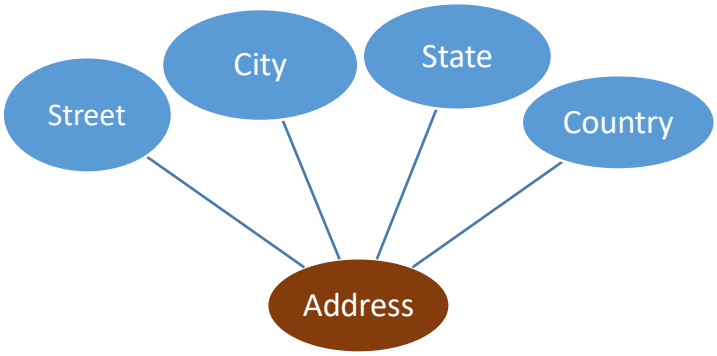
1)Simple Attributes/ Single Valued Attributes

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.



2) 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).

3)Multivalued Attributes

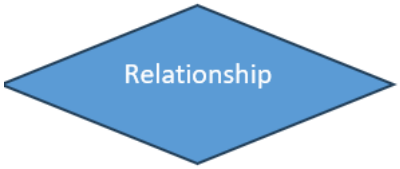
Multivalued attributes can have more than one value. For example, a person may have multiple email addresses or phone numbers. Multivalued attributes in DBMS are often used to store information about relationships between entities.
In ERD a multivalued attribute is represented by double ovals

4) 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

3.3. Relationship

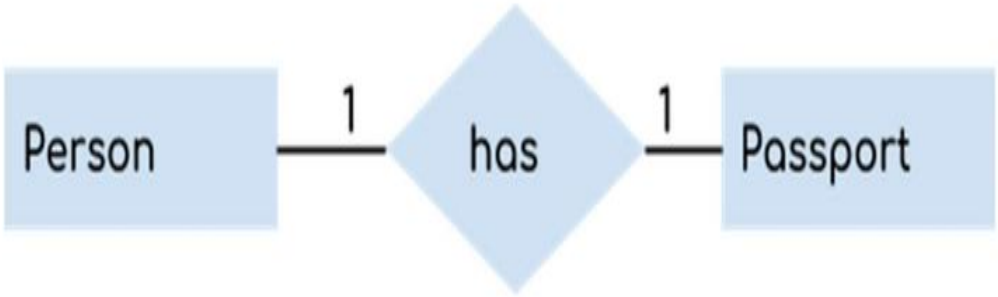
In the context of databases and Entity-Relationship Models (ERMs), relationships refer to the associations between different entities. These relationships define how data is stored and retrieved in a structured manner within a database.
In ERD a relationship is represented in Diamond



1)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.



2)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.

3)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.

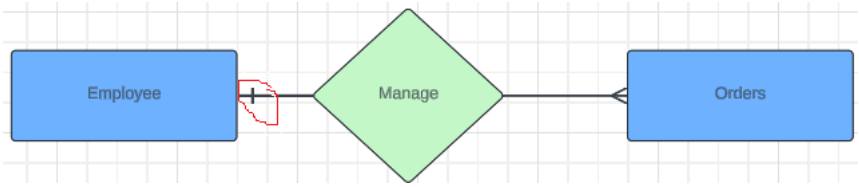
4. Define relationship using Cardinality symbols

Cardinality refers to the maximum number of times an instance in one entity can relate to instances of another entity. Ordinality, on the other hand, is the minimum number of times an instance in one entity can be associated with an instance in the related entity.

Cardinality and ordinality are shown by the styling of a line and its endpoint, according to the chosen notation style.

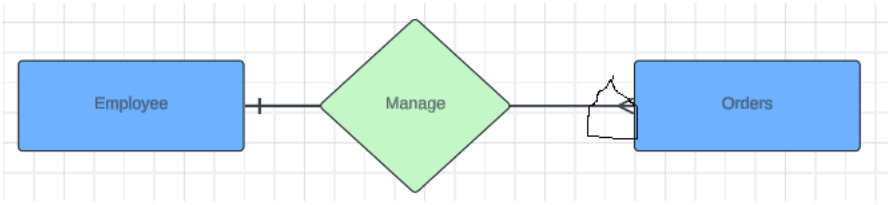
One

Example:



Many

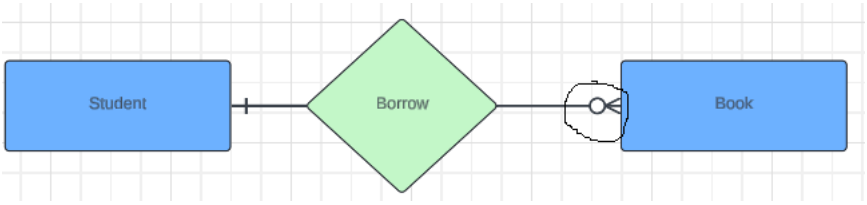
Example:



Zero or many

A zero or many relationships means that it is optional. There might not be any connection between the entities, or there might be one or more.

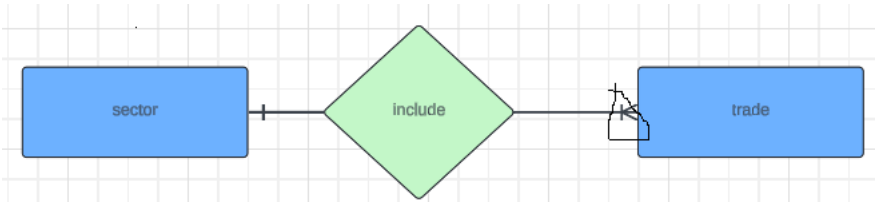
Example:



One or many

The “one” indicates that this is not an optional relationship, although there might be more than one.

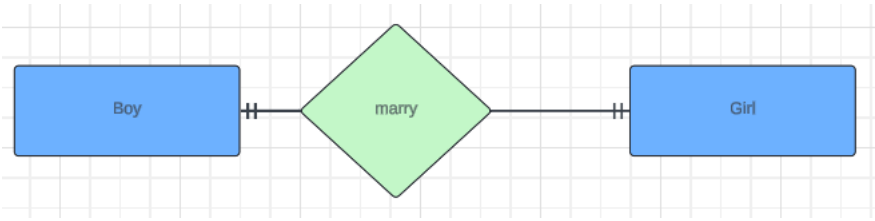
Example:



One and only one

No more than one relationship is possible between instances.

Example:



Zero or one

Another optional relationship, with a maximum of one.

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 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.

Step 1: 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

Step 2: 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:

1. 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.

2. List these key verbs between the entities they connect and draw a diamond around each one.
3. Associate entities with the appropriate relationships by connecting them with lines.
4. Label each relationship to show whether it is 1-1, 1-M, or M-M.

Step 3: 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

Step 4: Defining the Relationship type by using cardinality symbols

Step 5: Refining the Diagram

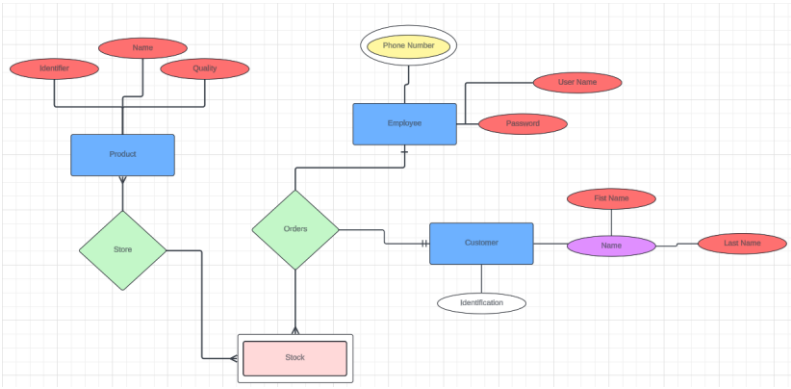
In this final step, we focus on refining the ER diagram to enhance clarity and readability. Organize the entities and relationships in a logical and intuitive manner. Group related entities together and arrange them in a way that reflects their connections.

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

Task:

1: Read key reading 2.2.3.

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



3: Draw ERD Using E-draw max

4: Present your work to the trainer and whole class

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

- **Edraw Max Online**
  This 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:

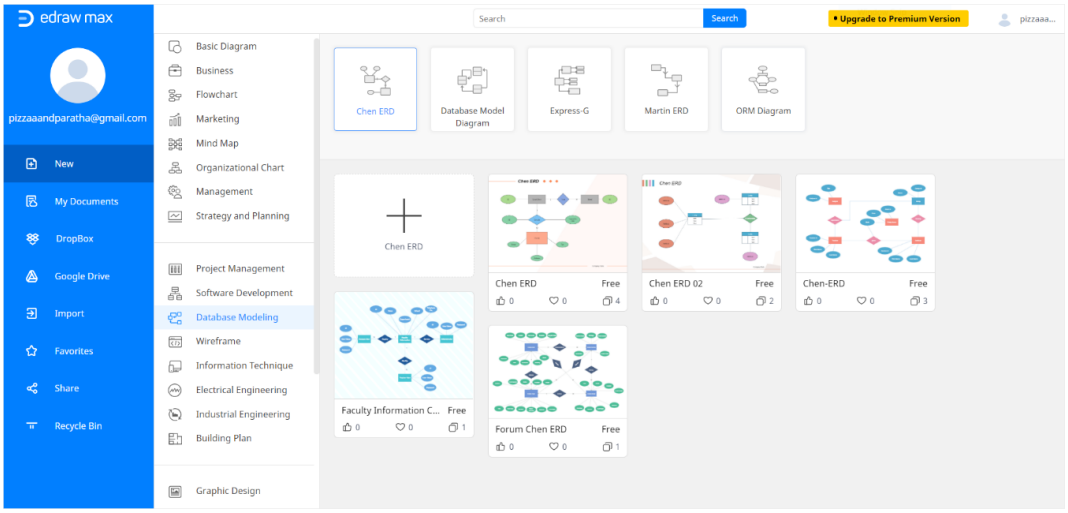
Step 1: Open Edraw Max Online

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

Step 2: 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.

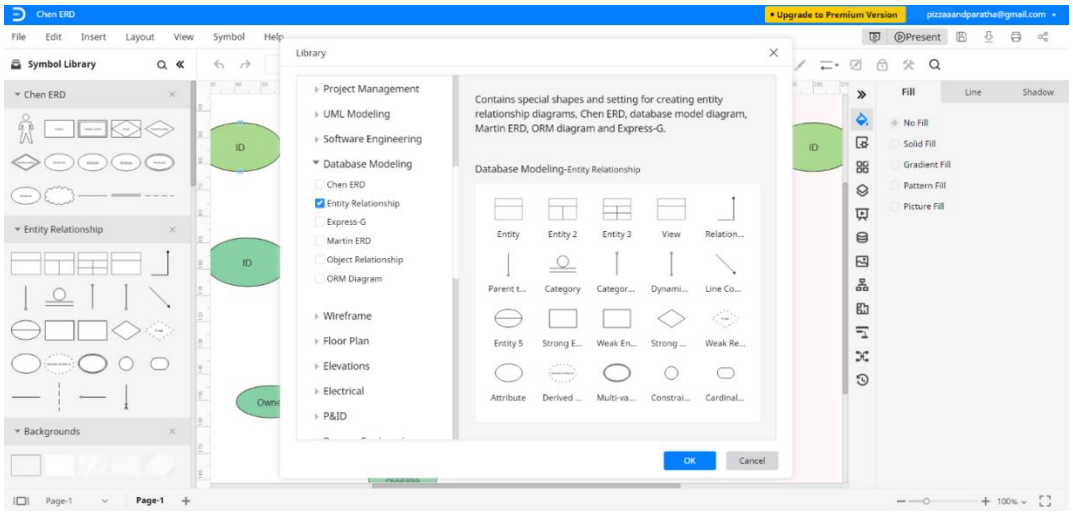
You can also create an ER Diagram from the start all by yourself. For this, you need to select the blank template.



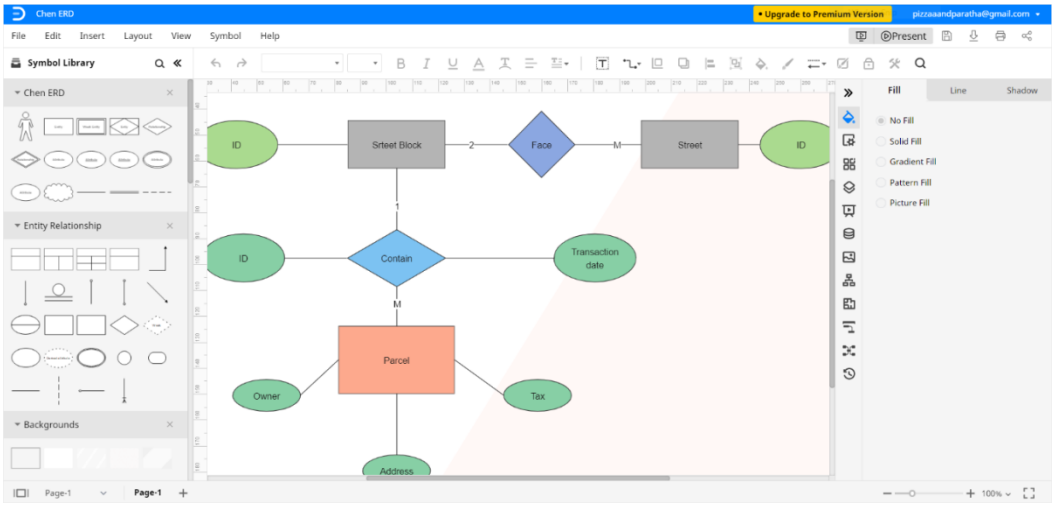
Step 3: Create an ER Diagram

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.

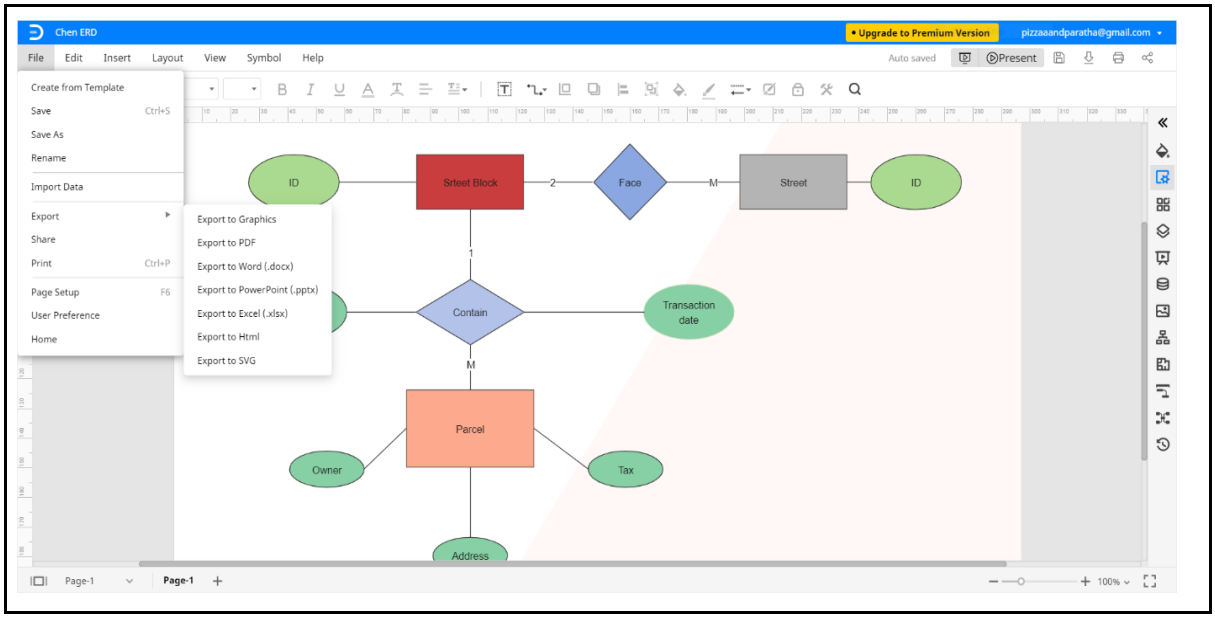


Step 4: Customize Your ER Diagram

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.

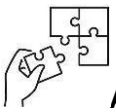
Step 5 – Save File

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

- Conceptual database schema is required when designing database because it 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.
- Components of ERD are:
  - ✓ Entities
  - ✓ Attributes
  - ✓ Relationship
  - ✓ Cardinality symbols
- Steps to create ERD
  - ✓ Defining entities
  - ✓ Establishing relationships
  - ✓ Adding attributes
  - ✓ Defining the Relationship type by using cardinality symbols
- Steps to draw ERD by using Edraw max
  - ✓ Open Edraw Max Online
  - ✓ Choose the ER Diagram
  - ✓ Create an ER Diagram
  - ✓ Customize Your ER Diagram
  - ✓ Save File



Application of learning 2.2

The owner of "Fresh Harvest 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. The database design should include an ERD that clearly models these entities and their relationships, ensuring the system can efficiently manage daily operations and support future business growth. 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

1. Definition of logical database schema

- 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.

2. 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.

a. 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.

b. 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.

c. 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.

d. 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.

e. 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).

f. 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.

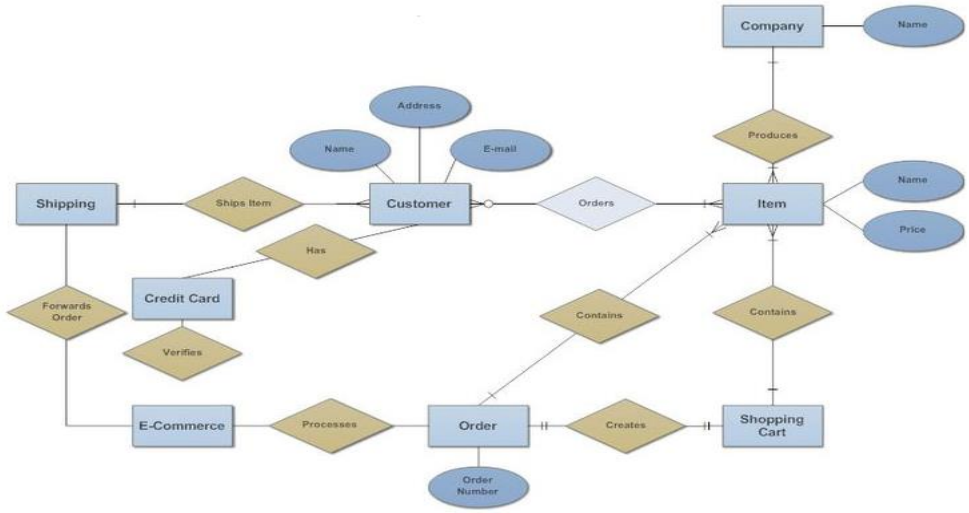
Practical Activity 2.3.2: Convert conceptual database schema to logical

Task:

1: Read key reading 2.3.2

2: 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:



3: Design logical database schema

4: Present your work to the trainer and whole class

5: Ask clarification where necessary

Key readings 2.3.2: To convert a conceptual database schema to a logical database schema in MySQL, you can follow these steps:

1. Review the Conceptual Schema: Understand the entities, attributes, and relationships defined in the conceptual schema.

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

3. 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.

4. 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.

5. 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.

6. 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.

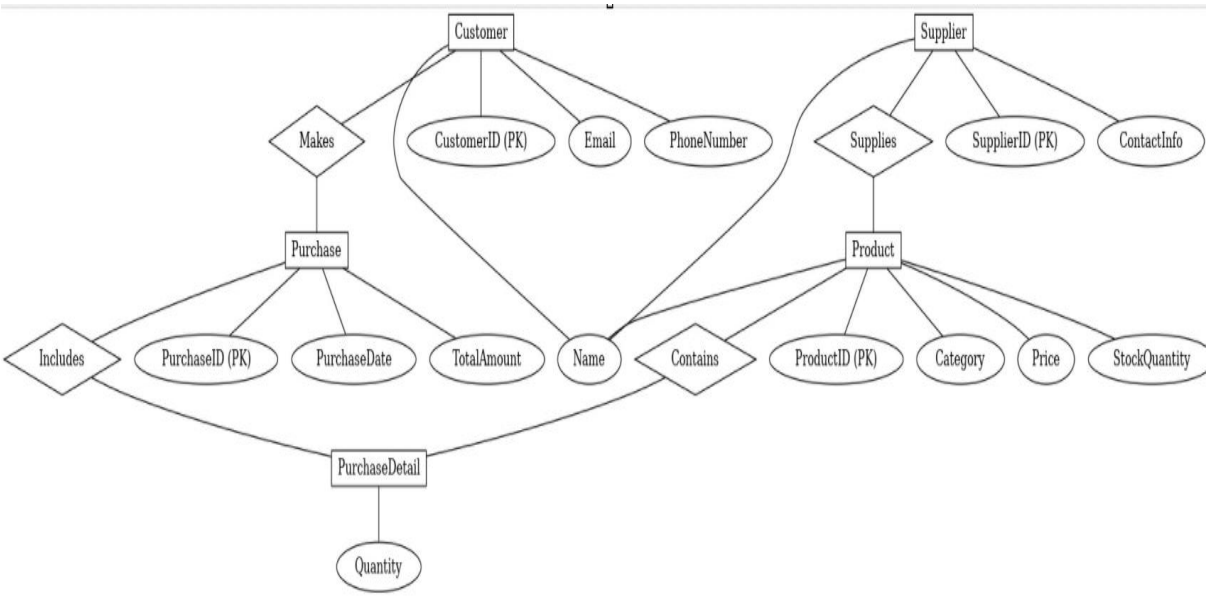
7. Review and Refine: Review the logical schema for completeness, accuracy, and efficiency. Ensure that it meets the requirements and performance goals of the MySQL database. Make any necessary refinements or adjustments based on your analysis.

Points to Remember

- 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.
- Table constraints are:
  - ✓ NOT NULL Constraint
  - ✓ UNIQUE Constraint
  - ✓ DEFAULT Constraint
  - ✓ CHECK Constraint
  - ✓ PRIMARY KEY Constraint
  - ✓ FOREIGN KEY Constraint.
- Steps to convert conceptual schema to logical scheme
  - ✓ Identify Entities and Attributes
  - ✓ Design Tables
  - ✓ Define Primary Keys
  - ✓ Establish Relationships
  - ✓ Define Constraints
  - ✓ Review and Refine

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

1. 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

2. Method of optimising database

1.1. Data Normalization

a) 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.

b) 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.

c) 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.

1.2. 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.

Practical Activity 2.4.2: Normalising database data

Task:

1: Read key reading 2.4.2

2: 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, ProductID, ProductName, Quantity, UnitPrice, and OrderDate. 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

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

4: Present your work to the trainer and whole class

Key readings 2.4.2: Normalising database data

- **Normalising database data**
  - ✓ **Step1:** 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

✓ **Step2: 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

✓ **Step3: 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

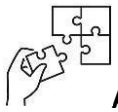
- Database optimization refers to a variety of strategies for reducing database system response time.
- The ways to optimise database are:
  - ✓ Normalization, form of normalisation are: 1NF, 2NF,3NF
  - ✓ Indexing
- **Steps to normalise a table**

✓ Check if column values are atomic if yes jump to the next step; if no make column values atomic

✓ Check if non key column depends on one prime attribute (one prime attribute). If yes jump to the next step if no; separate the columns that are depending on different prime attributes by spiriting a table into other tables

✓ Check if non key attributes are mutual independent. If no you are done. If yes separate the non key columns that are depending to one another

Note: if you a table is spirited into more tables, the relationship must be created between those new tables.



Application of learning 2.4

IBYIWACU restaurant 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

1. Definition of 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.

2. Types of DBMS

a) SQL DBMS (Relational DBMS)

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

- May face challenges with horizontal scaling (distributing data across multiple servers).

5) Use Cases:

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

b) NoSQL DBMS

a) 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.

b) Query Language:

- Uses various query languages or APIs specific to the NoSQL database type (e.g., MongoDB uses BSON, Redis uses commands).

c) Eventual Consistency:

- Often follows an eventual consistency model rather than strict ACID compliance, allowing for higher availability and partition tolerance.

d) Scalability:

- Generally designed to scale horizontally, allowing for easy distribution of data across multiple servers.

e) Use Cases:

- Suitable for applications with large volumes of data, real-time analytics, and those requiring flexibility (e.g., social media, big data applications).

3. Components of DBMS

1. Hardware
2. Software
3. Data
4. Procedures
5. Database Access Language
6. People

This module Will Focus on MySQL DBMS

- **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: Read key reading 2.5.2.

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

As a database designer you are requested to prepare MySQL environment

3: Referring to the key reading, prepare MySQL environment

4: Present your work to the trainer and whole class

Key readings 2.5.2: Preparation of DBMS Environment (MySQL)

To prepare a DBMS environment for MySQL, you can follow these steps:

1. 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.

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

3. 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`.

4. 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.

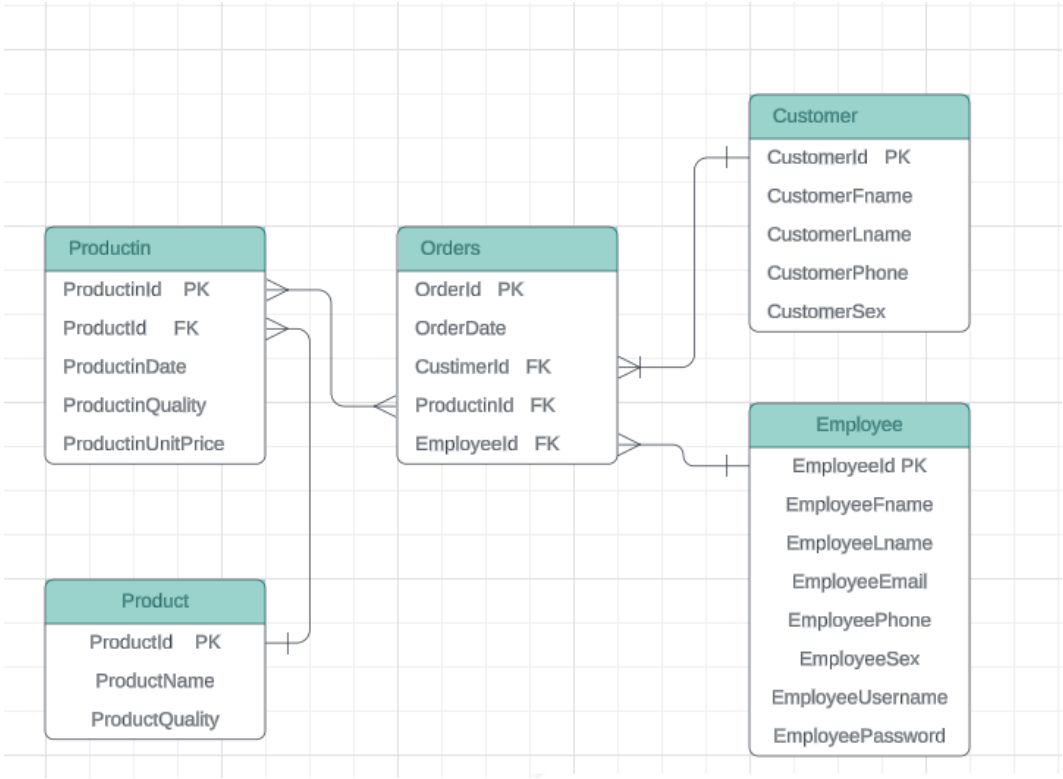
Practical Activity 2.5.3: Converting logic database schema to physical

Task:

1: Read key reading 2.5.3.

2: 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:



3: Referring to the key reading, design a physical database schema

4: Present your work to the trainer and whole class.

Key readings 2.5.3 Steps to convert logical database schema into physical database schema

1. **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.

2. **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.

3. **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.

4. **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

5. **Define Constraints:**

— Primary Key Constraints.

— Foreign Key Constraints.

— Unique Constraints:

— Null and not null

— Check Constraints

6. **Define relationship**: 1:1, 1:N, N:N

7. **Normalize Data (if needed):**

— Apply Normalization Rules: Ensure the database is normalized to the desired level to minimize redundancy and improve data integrity.

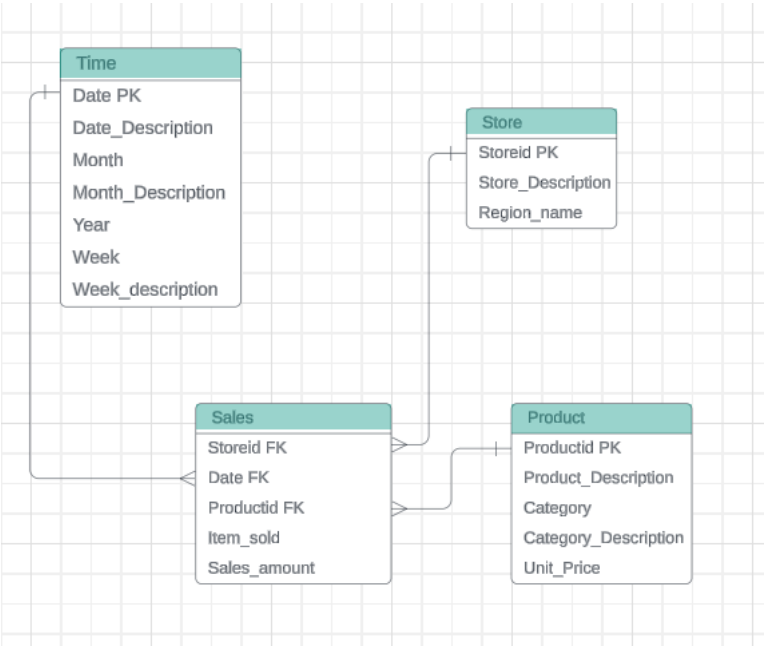
Points to Remember

- 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.
- Components of DBMS are:
    1. Hardware
    2. Software
    3. Data
    4. Procedures
    5. Database Access Language
    6. People
- Steps to prepare MySQL environment
    - ✓ Install MySQL
    - ✓ Configure MySQL
    - ✓ Start MySQL Server
    - ✓ Connect to MySQL

a. Steps to concert logical database schema into physical database schema
    - ✓ Select a Database Management System (DBMS)
    - ✓ Define Tables
    - ✓ Specify Columns and Data Types
    - ✓ Define Constraints
    - ✓ Define relationship
    - ✓ Normalize Data (if needed)

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

Written assessment

Question1: Match the following terms with their correct definitions:

Answers	Terms	Definitions
……	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.Conceptual Schema	B. Clearly defines schema objects with table names, field names, relationships, and integrity constraints but lacks technical requirements.
…….	3.Logical Schema	C. is the lowest level of data abstraction, which deals with the physical storage and representation of data.
…….	4.Physical Schema	D. Provides a high-level representation of the data, defining tables, columns, and relationships without concerning physical storage details.
…….	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**
…….	1.1NF	a. Ensures that a table has no repeating groups and that each column contains atomic (indivisible) values.
…….	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.
…….	3.3NF	c. Requires that a table be in 1NF and that all non-key attributes be fully functionally dependent on the primary key.

Practical assessment

GS KABURINGA 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 task is to create an Entity-Relationship Diagram (ERD) that outlines the relationships between different entities such as Students, Teachers, Classes, and Subjects. After finalizing the ERD, develop a logical data model that details the structure of the database, including tables, primary keys, foreign keys, and relationships between entities. Finally, create a physical data model, translating the logical design into a schema that can be implemented in MySQL database management system. The database should be designed to support the school's day-to-day operations, ensuring data integrity, consistency, and efficiency.

References

Books

Michael J. Hernandez, 2013. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design

Jan L. Harrington, 2026. Relational Database Design and Implementation

Martin Kleppmann, 2017. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems

Web links

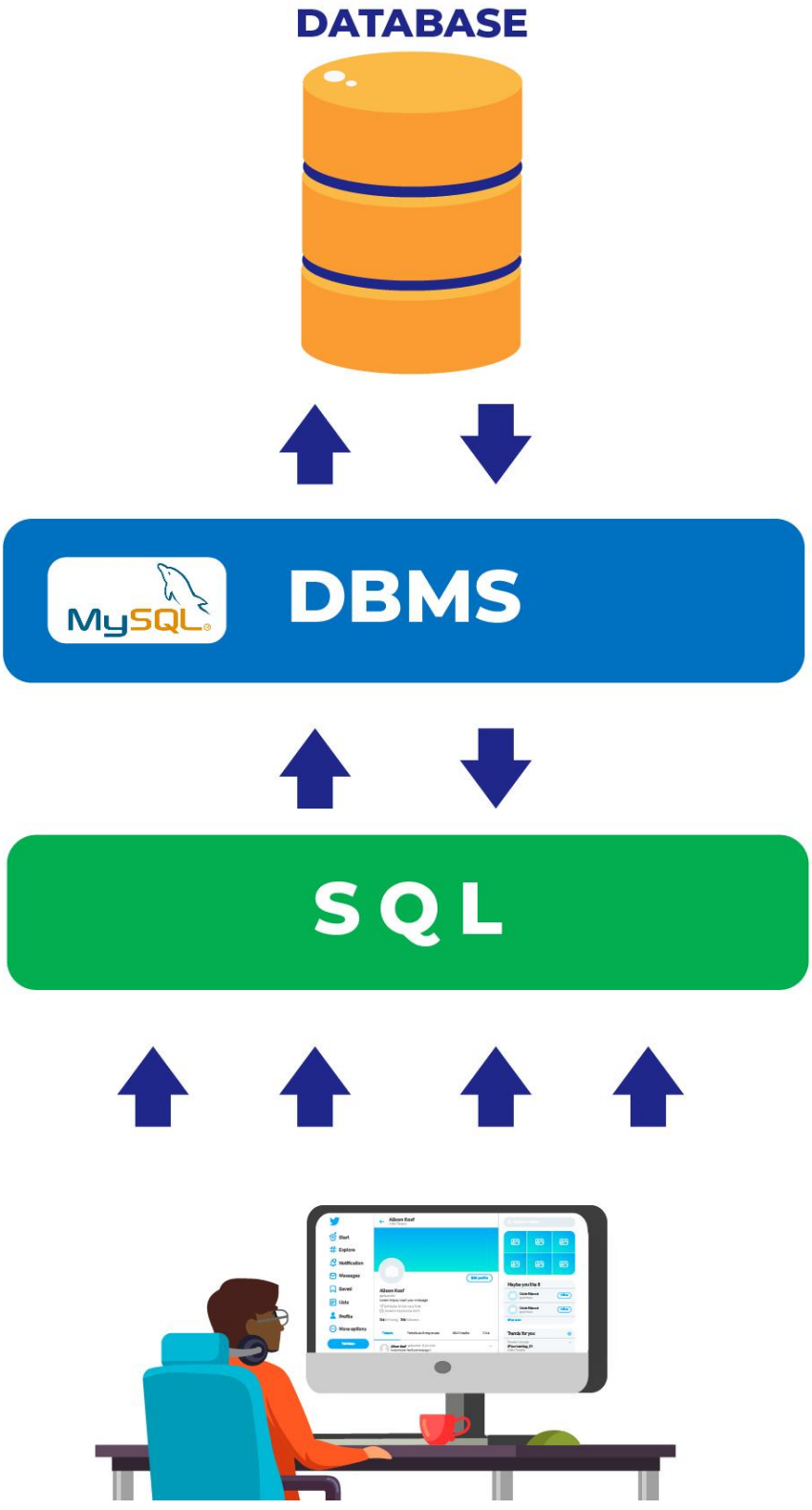
edrawsoft.com. (2024, 10 30). *Entity Relationship Diagram (ERD)*.

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

geeksforgeeks.org. (2024, 10 30). *Database Schemas*. Retrieved from https://www.geeksforgeeks.org/database-schemas/: https://www.geeksforgeeks.org/database-schemas/

ibm.com. (2024, 10 30). *What is a database schema?* Retrieved from https://www.ibm.com/: https://www.ibm.com/topics/database-schema

miro.com. (2024, 10 30). *How to draw an ER diagram*. Retrieved from https://miro.com/: https://miro.com/diagramming/how-to-draw-an-er-diagram/


DATABASE
MySQL
DBMS
SQL

Indicative contents

3.1 Description to SQL

3.2 Application of DDL commands

3.3 Application of DML commands

3.4. Application of DQL Command

3.5. Application of DCL commands

3.6. Application of TCL commands

Key Competencies for Learning Outcome3 : Implement Database

Knowledge	Skills	Attitudes
• Description of SQL • Description of DDL Command • Description of DML Commands • Description of DQL Commands • Description of DCL Commands • Description of TCL Commands	• Applying DDL Commands • Applying DML Commands • Applying DQL Commands • Applying DCL Commands • Applying TCL Commands	• Having self-Confident • Having problem solving skills

Duration: 36 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 DDL commands in SQL
3. Apply correctly DDL Queries based on database schema
4. Apply correctly DML Queries based on database schema
5. Apply correctly DQL Queries based on database schema
6. Apply correctly DCL queries based on database schema
7. Apply correctly TCL queries based on database schema

Resources

Equipment	**Tools**	**Materials**
● Computer	● MySQL ● Apache ● XAMPP ● Browsers	● Internet

Indicative content 3.1: Description to SQL

Duration: 3hrs

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. Differentiates SQL sub-Languages
3. Discuss on four SQL operators

2: Present your answers to your classmate and your trainer

3: Ask questions where necessary.

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

Key readings 3.1.1. Description of SQL

1. Introduction of SQL

Structured Query Language (SQL)

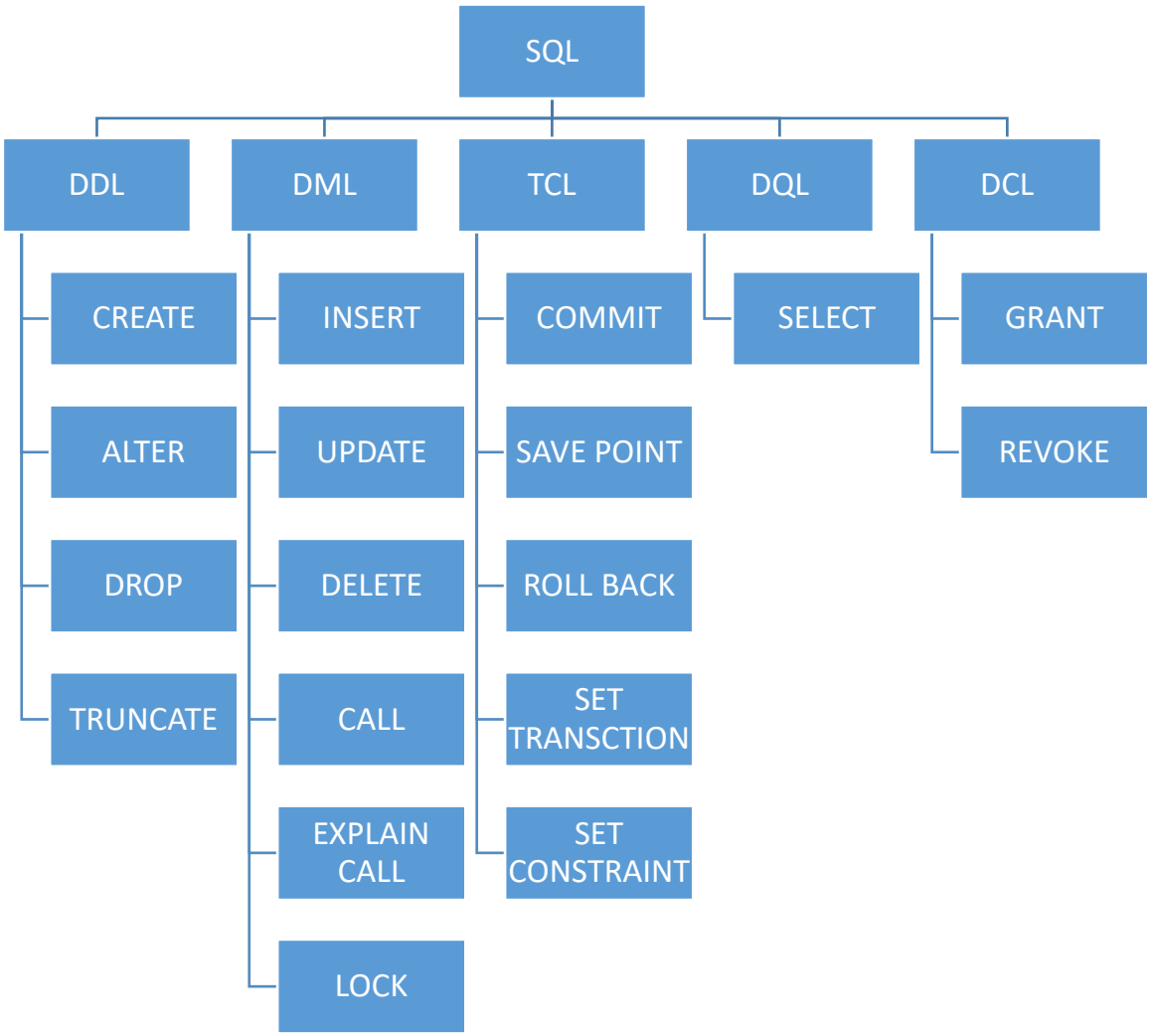
As we all know, is the database language by which we can perform certain operations on the existing database, and we can also use this language to create a database. SQL uses certain commands like CREATE, DROP, INSERT, etc. to carry out the required tasks.

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:

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

Now, we will see all of these in detail.



2. SQL sub-languages

a. 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

b. 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

c. 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

d. 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.

e. 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

3. 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.

a. 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;

b. 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**.

c. 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.

d. **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;

e. 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 alues 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 condiion becomes true.	(a != b) is true.

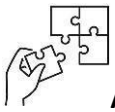
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand 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

- SQL stands for Structured Query Language. It's a specialized programming language designed to manage and manipulate data stored in relational databases.
- SQL sub-languages are:
  - ✓ DDL
  - ✓ DML
  - ✓ DCL
  - ✓ DQL
  - ✓ TCL

- **SQL operators are:**
  1. Arithmetic operators
  2. Bitwise operators
  3. Compound operators

4. Logical operators
5. Comparison operators



Application of learning 3.1

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

Indicative content 3.2: Application of DDL Commands

Duration: 6hrs

Theoretical Activity 3.2.1: Description of DDL Commands

Tasks:

1: Answer the following questions related to the DDL commands

i. Describe DDL Create Commands
ii. Describe DDL Drop Commands
iii. DDL Alter Commands
iv. 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

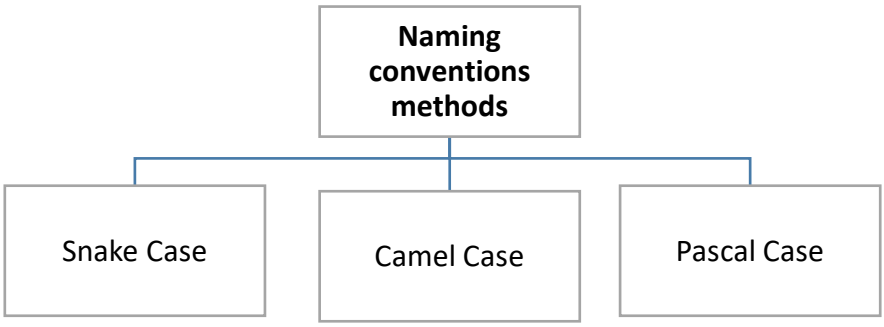
a. DDL Commands

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

a) 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 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 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 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.

b) 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

1. **Description of CREATE Commands**

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

a) Creating Database

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

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

The command CREATE DATABASE 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;

b) 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,**
    1. The name of the column. Column names should be unique within the table.
    2. **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.
    3. **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:**
    1. **Composite PRIMARY KEY:** A primary key that consists of more than one column.
    2. **FOREIGN KEY constraint:** A constraint that links a column (or combination of columns) in this table to a primary key in another table.
    3. **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)

);

c) 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 ;
```

d) Adding an Index to a MySQL Table

You can add index on a table in two ways:

⇒ **Creating a table with an Index:**

```
CREATE TABLE(
 column1 datatype PRIMARY KEY,
 column2 datatype,
 column3 datatype,
 ...
 INDEX(column_name)
);
```

In this example, we are create a new table **CUSTOMERS** and adding an index to one of its columns using the following CREATE TABLE query

```
CREATE TABLE CUSTOMERS (
  ID INT NOT NULL,
  NAME VARCHAR (20) NOT NULL,
  AGE INT NOT NULL,
  ADDRESS CHAR (25),
  SALARY DECIMAL (18, 2),
  INDEX(ID)
);
```

⇒ **Adding an index on a created table:**

To create an index on existing table, we use the following SQL statements −

- **With CREATE INDEX Statement**

General syntax: CREATE INDEX index_name ON table_name;

Example: CREATE INDEX NAME_INDEX ON CUSTOMERS (Name);

- **With ALTER Command**

General syntax: ALTER TABLE tbl_name ADD INDEX index_name (column_list);

Example: ALTER TABLE CUSTOMERS ADD INDEX AGE_INDEX (AGE);

2. Describing DROP Commands

a) Drop Database

General syntax: **Drop database** 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;

b) 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;

3. 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.

a) 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;

b) 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;

c) 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;

d) 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;

e) 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;*

4. 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;

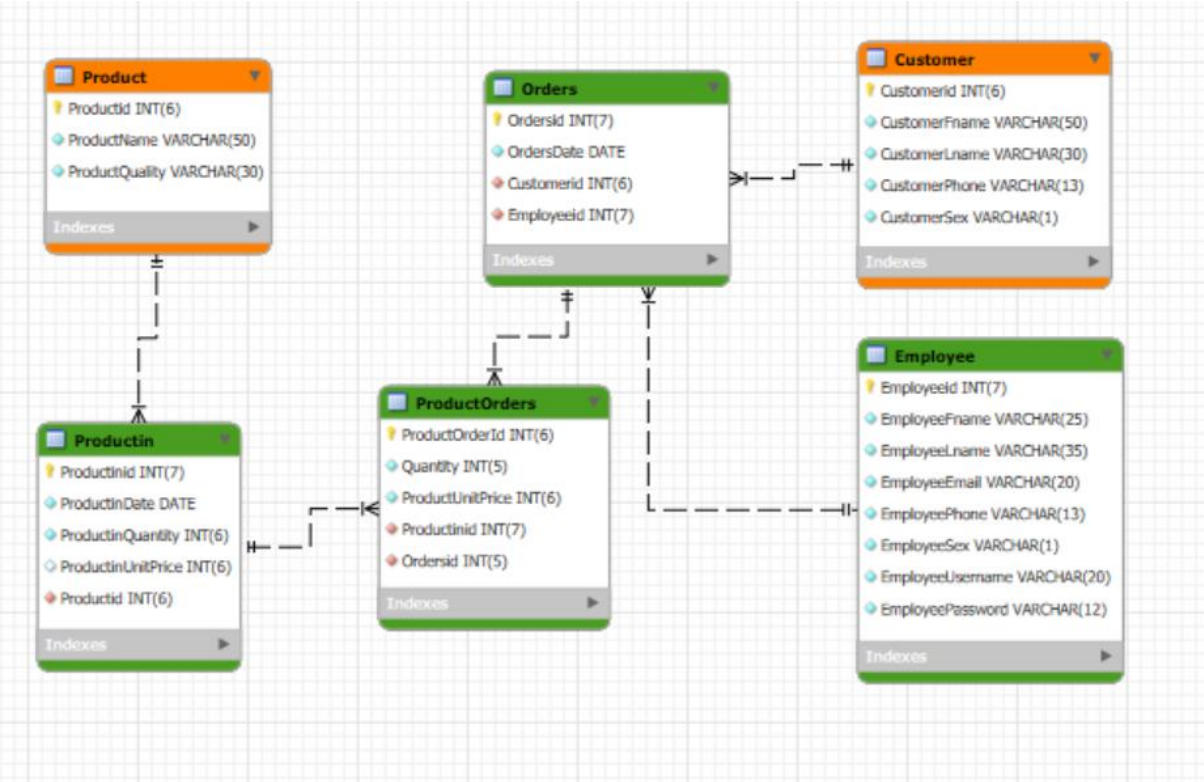
Practical Activity 3.2.2: Applying DDL Commands

Task:

1: Read key reading 3.2.2.

2: Read the below task:

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



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

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

Key readings 3.2.2:
Applying DDL Commands

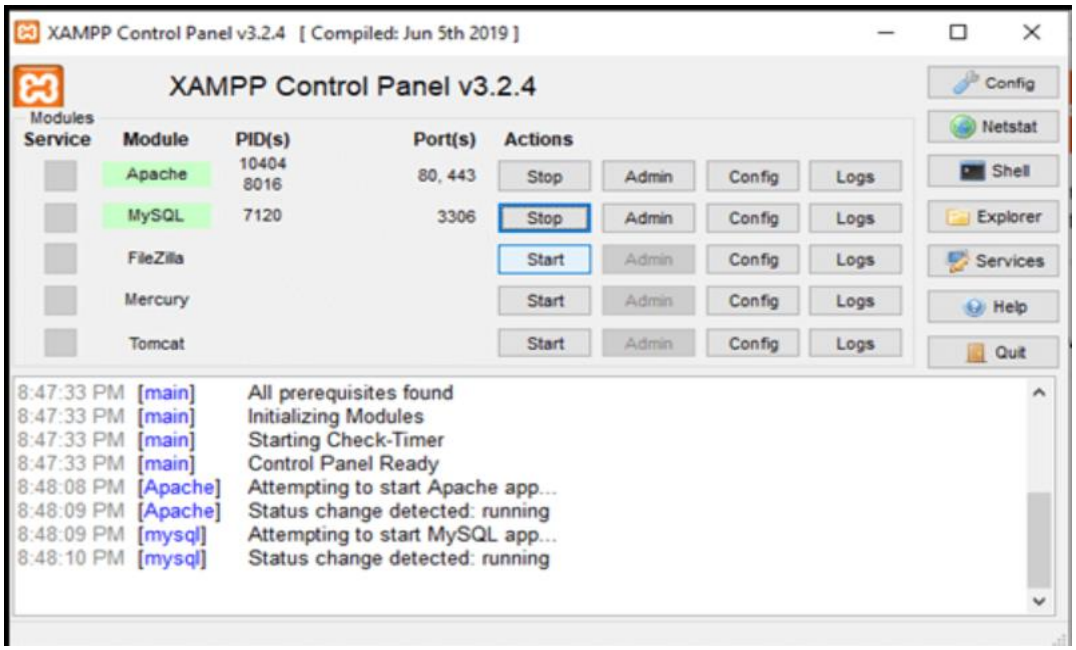
Creating database in MYSQL by using XAMPP Follow the below steps:

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:

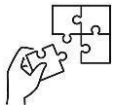
Step 5: Creating a Table

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

Points to Remember

- 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;
- To create database in the DBMS, use the following steps:
  - ✓ Run XAMPP control panel
  - ✓ Start Apache and MYSQL
  - ✓ Run MySQL Admin
  - ✓ Select SQL menu
  - ✓ Write a statement to create Database
  - ✓ Run the statement

- To add tables in the created database, use the following steps:
  - ✓ Select the database
  - ✓ Select SQL menu
  - ✓ Write a statement to create Table
  - ✓ Run a statement
- To remove unwanted tables from database, use the below steps:
  - ✓ Select the database
  - ✓ Select SQL menu
  - ✓ Write a statement to modify table structure
  - ✓ Run a statement
- To Truncate tables, use the below steps:
  - ✓ Select the database
  - ✓ Select SQL menu
  - ✓ Write a statement to truncate table
  - ✓ Run a statement



Application of learning 3.2

As a database developer, a 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: 8hrs

Theoretical Activity 3.3.1: Description of DML Commands

Tasks:

1: You are asked to answer the following questions related to the DML commands

1. Describe DML INSERT Commands
2. Describe DML UPDATE Commands
3. Describe 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

1. 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.
  1) **(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.
  2) **(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.
  3) **(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)**:

1) These are the arguments or parameters that the stored procedure takes.
2) If the procedure does not require any parameters, the parentheses are still included but left empty: CALL procedure_name();.
3) 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);

5. 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:

1) The EXPLAIN command will show you the execution plan of the queries inside the stored procedure, providing insight into how the procedure executes.

2) 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.

6. 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:

1. **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.
2. **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.
3. **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**:

    1) READ: Allows reading but prevents writing.

    2) 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:

1: Read Key reading 3.3.2

2: You are requested to read careful 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:

1. **Student**(StudentID INT(10)PK, StudentName varchar(100), StudentEmail varchar(100), StudentPhone varchar(20), StudentBirthDay date)
2. **Subject**(SubjectID int(7)PK, SubjectName varchar(100))
3. **Professor**(ProfessorIDint(7)PK,ProfessorNamevarchar(100),ProfessorEmail varchar(100), ProfessorPhone varchar(20))
4. **ProfessorSubject**(ProfessorSubjectID Int(10), ProfessorID INT(10)FK, SubjectID INT(10)FK)
5. **Class** (ClassID INT(10)PK, ClassCode varchar(20),BeginDate date, EndDate Date, SubjectID int(7)FK, ProfessorID int(7)Fk)
6. **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 tables as the task 2 above requested

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

Key readings 3.3.2:

Apply DML Commands

1. **Steps to insert data in a database table**

a) Select the Database

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

b) 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.

c) Select SQL Menu

d) Insert Data into the Table

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

e) Run the statement

To run the statement, click on GO or RUN

f) Verify the Data Insertion

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

2. **Steps to update database data**

a) Select the Database by clicking its name

Choose the database where the table is located.

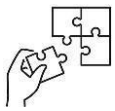
1) Select table by clicking table name to Check the Table Structure (Optional)
2) Select SQL menu
3) Write a command to update database data
4) Click on go to run the command
5) Verify if the updated data is applied to database by clicking on database table name>browse

b) Steps to delete database data

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

Points to Remember

- **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

You are a database developer 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.

Your task is 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):
    - student_id (Primary Key, INT)
    - first_name (VARCHAR)
    - last_name (VARCHAR)
    - date_of_birth (DATE)
    - enrollment_date (DATE)
    - grade_level (INT)
2. **Teachers Table** (teachers):
    - teacher_id (Primary Key, INT)
    - first_name (VARCHAR)
    - last_name (VARCHAR)
    - subject (VARCHAR)
    - hire_date (DATE)
3. **Courses Table** (courses):
    - course_id (Primary Key, INT)
    - course_name (VARCHAR)
    - teacher_id (Foreign Key, INT)
4. **Enrollments Table** (enrollments):
    - enrollment_id (Primary Key, INT)
    - student_id (Foreign Key, INT)
    - course_id (Foreign Key, INT)
    - 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: 8hrs

Theoretical Activity 3.4.1: Description of DQL Commands

Tasks:

1: Answer the following questions:

1. Describe DQL SELECT Command
2. What is the use of aggregation functions in SQL?
3. Explain the use of where, order by, group by, having, and join SQL clauses

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

1. 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.

2. 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.

Lets 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

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

Example: Determine the lowest salary

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

Output:

LowestSalary
403

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

Example: Get the highest salary

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

Output:

HighestSalary
802

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

Example: count the number of employees

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

Output:

TotalEmployees
6

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

Example: Calculate the total salary

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

Output:

TotalSalary
3120

5) 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()).

3. 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

1. SELECT * FROM Customers ORDERBY Country;
2. SELECT * FROM Customers ORDERBY Country DESC;

ORDER BY Several Columns

1. SELECT * FROM CustomersORDERBY Country, CustomerName;
2. 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
GROUP BY column_name(s)
ORDER BY column_name(s);

Example:

1. 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

2. 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	XXXXXXXXXX	18	1
2	PRATIK	BIHAR	XXXXXXXXXX	19	2
3	RIYANKA	SILGURI	XXXXXXXXXX	20	2
4	DEEP	RAMNAGAR	XXXXXXXXXX	18	3
5	SAPTARHI	KOLKATA	XXXXXXXXXX	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

1. SQL INNER JOIN

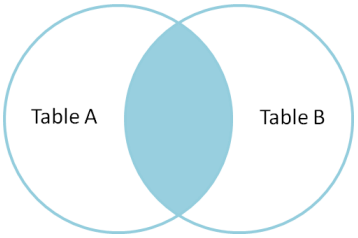
The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

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

2. 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;
```

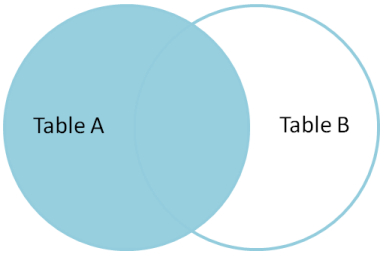
Here,

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*

3. 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

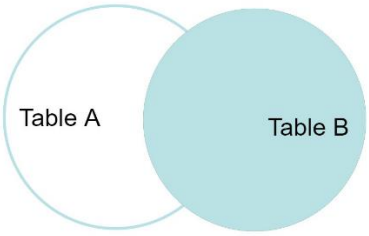
JOIN For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

➔ **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

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;

Output:

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 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;

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

3: Referring to the key reading 3.4.2, perform the task 2 above

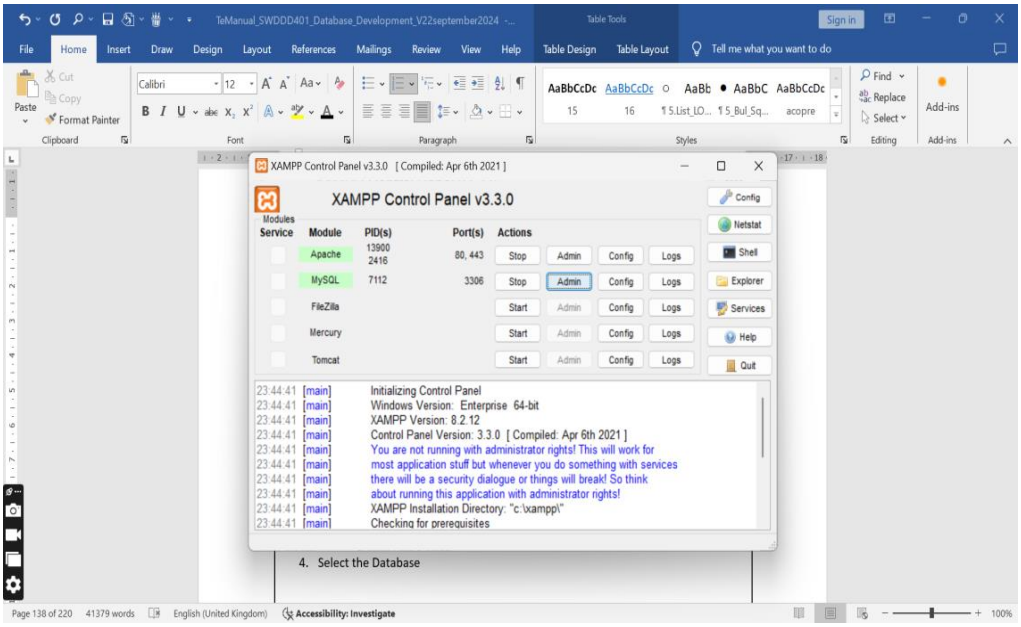
4: Present your work to your trainer and a whole class

Key readings 3.4.2: Apply DQL Commands

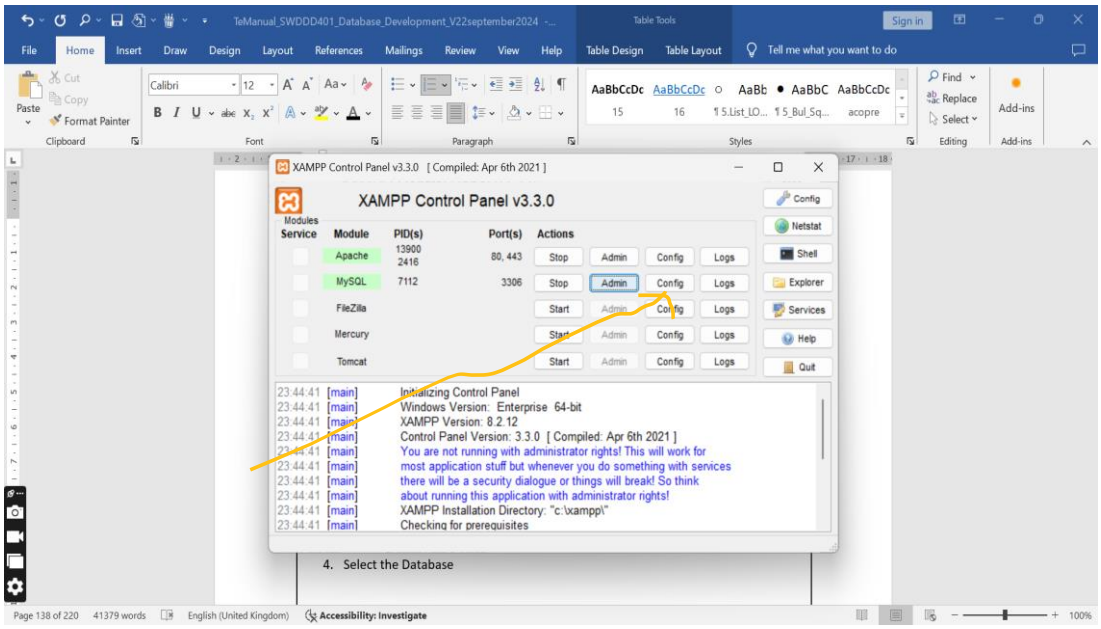
To select data from database table, follow the steps below:

1. Open xampp by double clicking on its icon

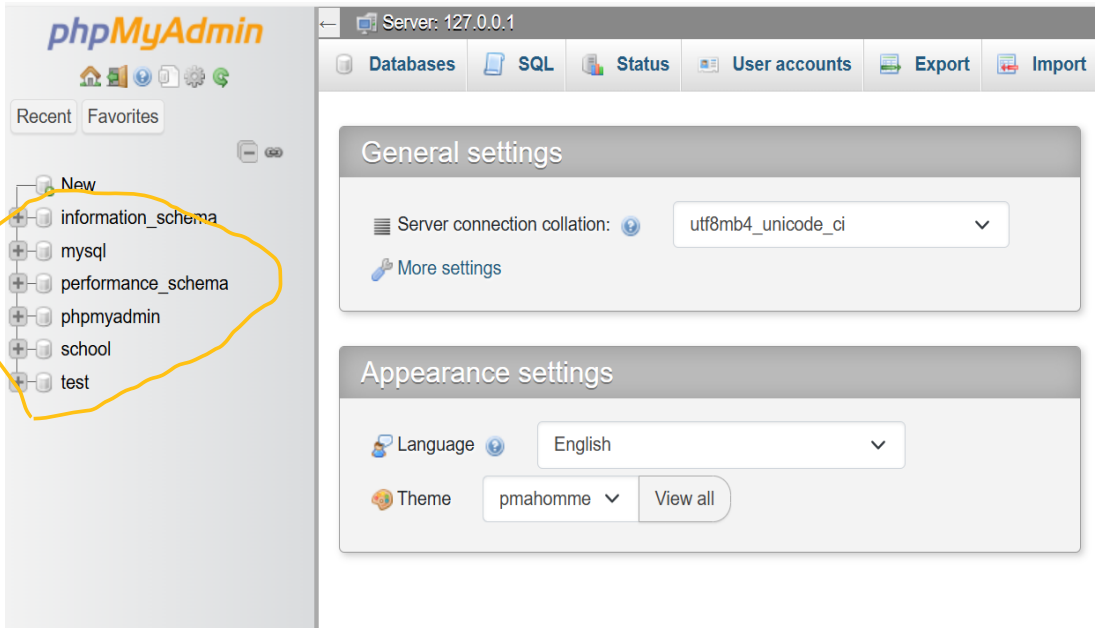
2. Start MySQL and Apache



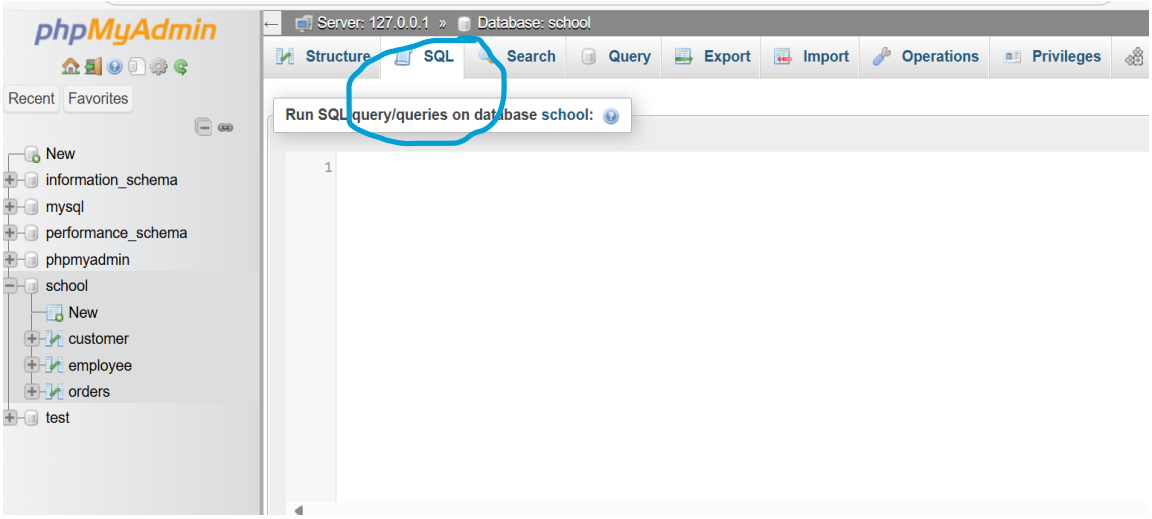
3. Run MySQL as Admin



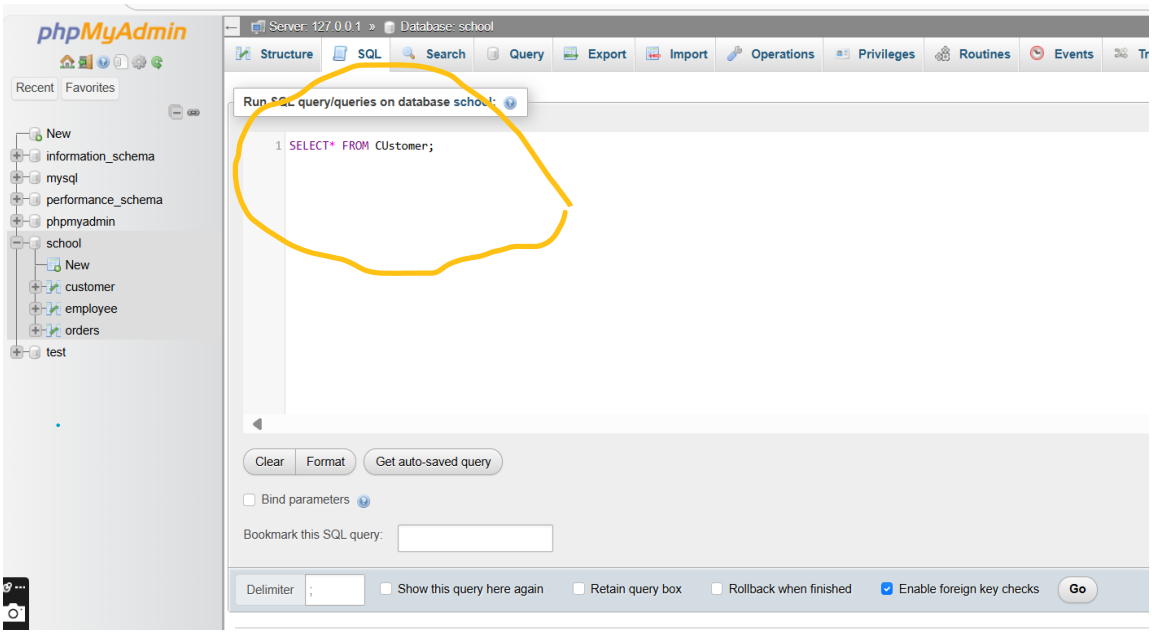
4. Select the Database



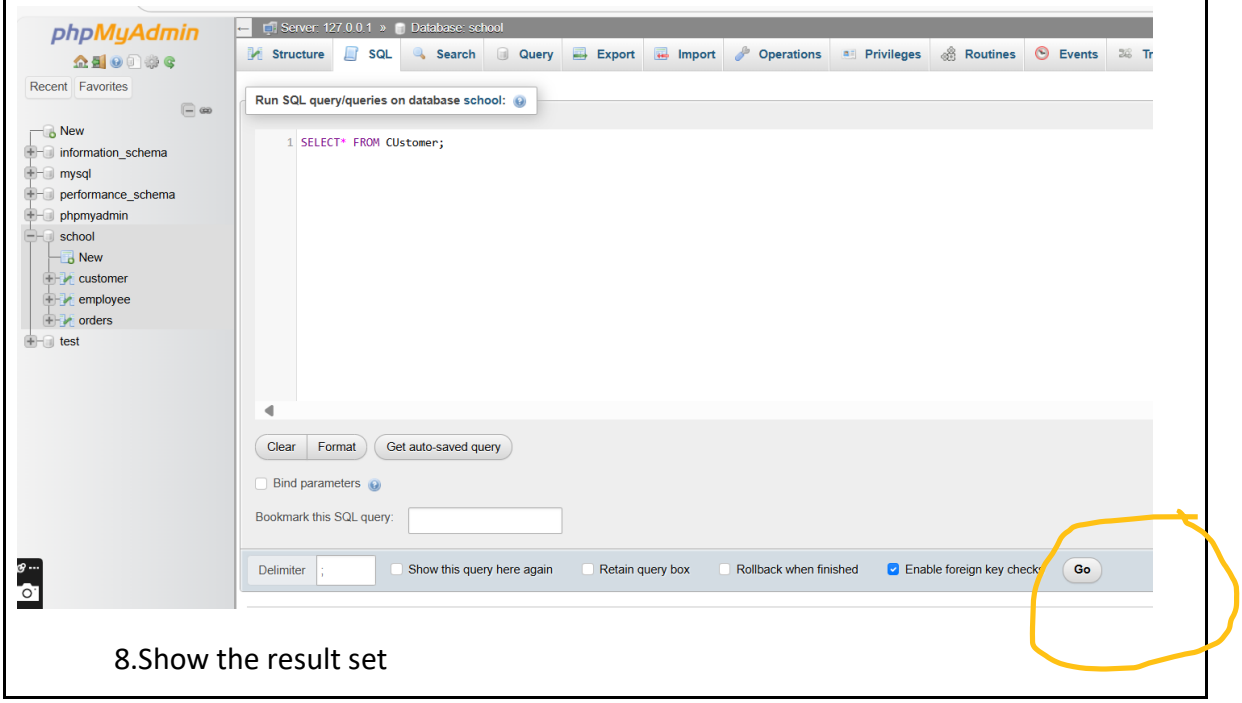
5. Select SQL Menu



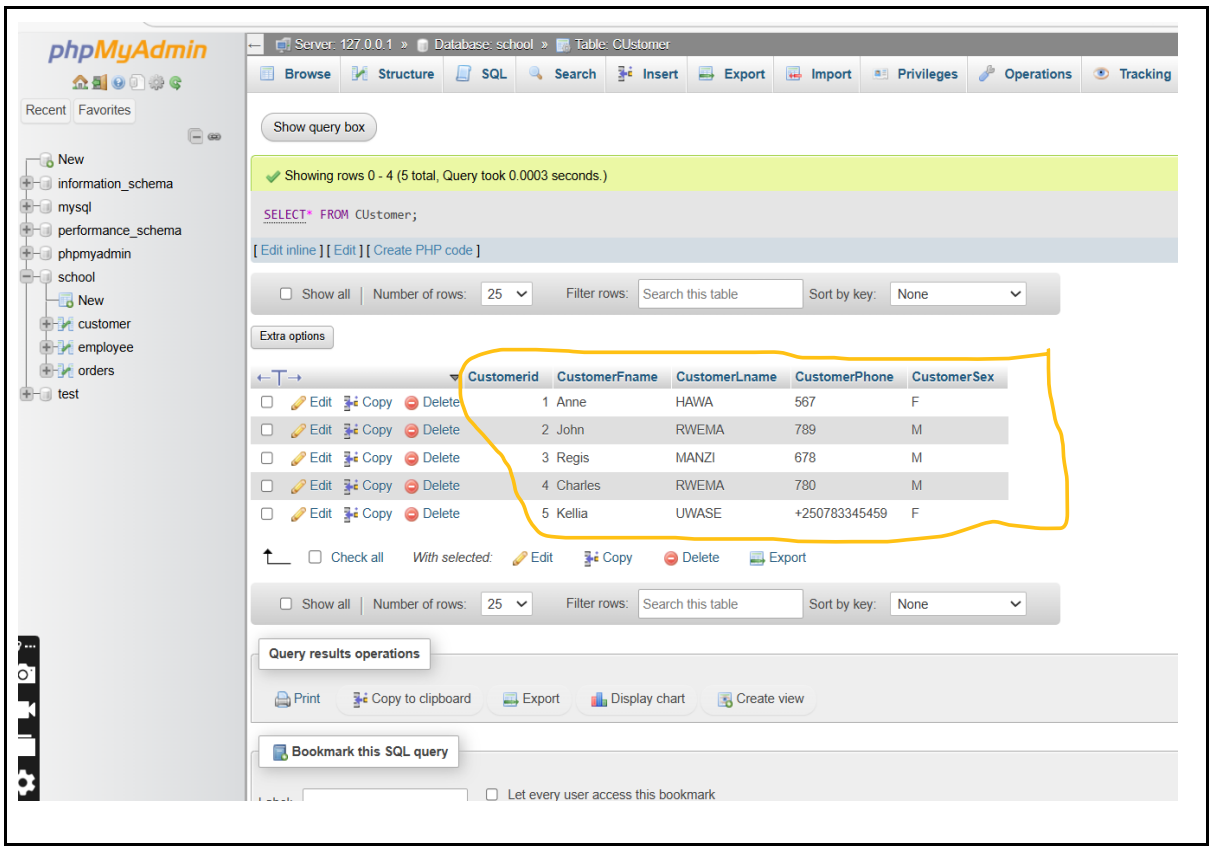
6. Type a statement to select data



7. Run it by clicking on GO

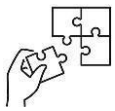


8.Show the result set



Points to Remember

- **General syntax for select command:** SELECT column1, column2, ... FROM table_name;
- An aggregate function is a function that performs a calculation on a set of values, and returns a single value
- SQL Clauses are:
  - ✓ Where clause
  - ✓ Group By
  - ✓ Order By
  - ✓ Having
  - ✓ Join
- Select command is used to generate report
- Select can display information from multiple tables.



Application of learning 3.4

You are a database developer 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: 4hrs

Theoretical Activity 3.5.1: Description of DCL Commands

Tasks:

1: You are requested to answer the following questions:

i. Describe SQL Grant Command

ii. Describe SQL Revoke Command

2: Present the findings to the whole class

3: Ask questions where necessary.

4: For more clarification, read the key readings 2.5.1

Key readings 3.5.1:

DCL commands 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 owner's of the database object can provide/remove privileges on a database object.

1. 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.

2. 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:**

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

1) System privileges - This allows the user to CREATE, ALTER, or DROP database objects.

2) 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.

b) **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: Read key reading 3.5.2

2: 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

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

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

Key readings 3.5.2

•To grant privileges to a user in **MySQL**, follow these steps:

1. 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

2. 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.

3. 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;

4. Apply Changes:

Run the following command to reload the privilege tables and apply changes.
FLUSH PRIVILEGES;

5. Verify User Privileges (Optional):

To check the privileges granted to the user: SHOW GRANTS FOR 'username'@'host';

6. 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;

1. Revoke Privileges (If Necessary):

If you need to revoke privileges later, use the REVOKE command:

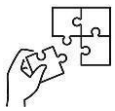
REVOKE privilege ON database.table FROM 'username'@'host';

For example, to revoke INSERT privilege from the user on the employees table:

REVOKE INSERT ON company.employees FROM 'username'@'localhost';

Points to Remember

- 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}
  - **To grant and revoke privileges to/from user follow the below steps:**
    - ✓ Login to MySQL as an Administrator
    - ✓ Create a User
    - ✓ Grant Privileges to the user
    - ✓ Revoke Privileges from user(If Necessary)



Application of learning 3.5

You are a database developer 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. The database is not secured everyone can access information in database, this may cause the data loss and falsified data. To avoid that issue the school need to hire a database developer to add the database users that are:

- ✓ 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

- Withdraw adding a new teacher from a teacher user

Indicative content 3.6: Applying TCL Commands

Duration: 7hrs

Theoretical Activity 3.6.1: Description of TCL Commands

Tasks:

1: You are requested to answer the following questions:

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:

Apply 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.

1. 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;

2. 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;

3. 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.

3. 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.

- **Use Cases**:

♦ **Data Consistency**: Ensuring consistent reads in financial transactions where even a minor inconsistency can cause significant issues.

♦ **Read-only Reports**: Using READ ONLY transactions for generating reports that should not alter the data.

♦ **Performance Optimization**: Adjusting isolation levels to balance between performance and consistency, especially in high-traffic applications.

5. 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 key reading 3.6.2

2: Read the task below:

You manage a retail database with Employees, Orders, and Customers tables. Perform the following tasks using TCL commands to control the transaction flow:

1. **Insert Records and Commit:**

Insert records into Employees with:
(EmpID: 1, Name: 'John Doe', Department: 'HR')
(EmpID: 2, Name: 'Jane Smith', Department: 'Finance')

- Use COMMIT to save the changes.

2. **Create Savepoint and Insert:**
   - Insert (OrderID: 101, ProductName: 'Laptop', Quantity: 2) into Orders.
   - Set a SAVEPOINT named savepoint1.
   - Insert another record: (OrderID: 102, ProductName: 'Phone', Quantity: 5).
3. **Rollback to Savepoint:**
   - Realize the second order insert is incorrect; ROLLBACK TO savepoint1 to undo it.
4. **Update and Full Rollback:**
   - Update Employees to set Department = 'Marketing' for EmpID = 1.
   - Delete the employee with EmpID = 2.
   - Use ROLLBACK to undo both changes.
5. **Set Transaction and Query:**
   - Set the transaction to READ ONLY.
   - Retrieve all employees from the HR department.
6. **Defer Constraints:**
   - Set constraints to DEFERRED.
   - Insert (OrderID: 103, CustomerID: 201) into Orders.
   - Update Customers to change Customer Name for CustomerID = 201 to 'Michael Johnson'.
   - Enforce constraints immediately after the changes.

3: Perform the task2 above

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

Key readings 3.6.2: Applying TCL Commands

1. COMMIT

Steps Followed:

1. 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.
2. Ensure that the operations are correct and the data is in a valid state before executing COMMIT.
3. 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.

2. SAVEPOINT

Steps Followed:

1. Use SAVEPOINT to create a point within a transaction to which you can later roll back.
2. After executing several DML operations, use SAVEPOINT to mark a specific point.
3. 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.

3. ROLLBACK

Steps Followed:

1. After a series of operations, use ROLLBACK to undo changes made within the current transaction.
2. If you want to undo changes up to a specific point, use ROLLBACK TO SAVEPOINT.
3. 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.

4. SET TRANSACTION

Steps Followed:

1. Before starting a transaction, use SET TRANSACTION to define properties like the isolation level and read/write access.
2. Set the transaction properties by executing a SET TRANSACTION command, for example, SET TRANSACTION READ ONLY;
3. 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.

5. SET CONSTRAINTS

Steps Followed:

1. Use SET CONSTRAINTS to temporarily defer or immediately enforce constraints like FOREIGN KEY or UNIQUE.
2. Execute SET CONSTRAINTS ALL DEFERRED to delay the constraint check until the end of the transaction.
3. Execute SET CONSTRAINTS ALL IMMEDIATE to enforce constraints immediately.

Key Notes:

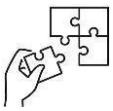
- Deferred constraints are useful when the order of data modifications may temporarily violate a constraint but will be resolved by the end of the transaction.
- It provides flexibility in managing the order of operations in complex transactions.

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

- **COMMIT**: Ensure all operations are validated and correct before using COMMIT to make changes permanent.
- **SAVEPOINT**: Use SAVEPOINT to create a rollback point within a transaction for better error recovery.
- **ROLLBACK**: Apply ROLLBACK to undo changes since the last COMMIT or to a specific SAVEPOINT for maintaining data integrity.
- **SET TRANSACTION**: Define the transaction's isolation level and access mode using SET TRANSACTION to control data consistency and concurrency.
- **SET CONSTRAINTS**: Utilize SET CONSTRAINTS to defer constraint checks until the end of a transaction for more flexibility in data operations.



Application of learning 3.6

You are managing an inventory management system for a warehouse that handles multiple product categories, orders, and customer details. The database includes three key tables: Products, Orders, and Customers. You are responsible for ensuring data integrity and proper transaction flow using TCL commands (COMMIT, SAVEPOINT, ROLLBACK, SET TRANSACTION, and SET CONSTRAINTS). Follow the steps below to complete the task.

Exercise Instructions:

1. **Add New Products and Commit Changes:**
    - Insert new records into the Products table:
        - (ProductID: 1001, ProductName: 'Electric Drill', Category: 'Tools', StockQuantity: 50)
        - (ProductID: 1002, ProductName: 'Safety Goggles', Category: 'Safety Gear', StockQuantity: 150)
    - Ensure the changes are saved permanently using COMMIT.
2. **Create a Savepoint After Adding Orders:**
    - Insert a new record into the Orders table for a recent customer order: (OrderID: 5001, ProductID: 1001, CustomerID: 301, Quantity: 5, Status: 'Pending')
    - Set a SAVEPOINT named order_point1 after this insertion.
    - Insert another order record:
      (OrderID: 5002, ProductID: 1002, CustomerID: 302, Quantity: 3, Status: 'Pending')
3. **Rollback a Mistaken Order Entry:**
    - Realize that the second order quantity is incorrect. Use ROLLBACK TO order_point1 to undo the second insertion while keeping the first order intact.
4. **Update Stock Quantities and Use Rollback:**
    - Update the Products table to decrease StockQuantity of ProductID = 1001 by 5.
    - Update the Products table to decrease StockQuantity of ProductID = 1002 by 3.

- Realize the stock update was premature and use ROLLBACK to undo both stock updates.

5. **Set Transaction Mode for Read-Only Data Analysis:**
   - Set the transaction to READ ONLY mode.
   - Retrieve the list of all products under the Safety Gear category without making any changes.
6. **Defer Constraints for Batch Operations:**
   - Set all constraints to DEFERRED to temporarily disable constraint checking.
   - Insert a new order in Orders:
     (OrderID: 5003, ProductID: 1001, CustomerID: 303, Quantity: 10, Status: 'Pending')
   - Update the Customers table to modify the CustomerName of CustomerID = 303 to 'Alex Brown'.
   - Use SET CONSTRAINTS ALL IMMEDIATE to re-enable and enforce constraints immediately after the changes.

Learning outcome 3 end assessment

Theoretical assessment

1) Match the terms in COLMN A and their definition in COLUMN B

	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) Fill-in-the-Blank Questions

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) Chose the correct answers:

i. **What is the correct syntax for creating a new database named "mydb"?**

a. CREATE DATABASE mydb; b) CREATE DATABASE mydb(); c) DATABASE CREATE mydb; d) NEW DATABASE mydb;

ii. **Which clause is used to specify conditions for selecting rows in a SELECT statement?**

iii. WHERE b) FROM c) GROUP BY d) HAVING

iv. **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')

v. **Which SQL aggregate function calculates the average value of a column?**

a. 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

GS AGAPE 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. Your first task as a database developer is to use DDL commands to create the necessary tables: Students, Teachers, Courses, and Grades, ensuring the correct relationships, are in place. After creating 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. Then, 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. Manage permissions to database by ensuring that only authorized users can view or modify certain data. Lastly, 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

Books

C. J; 2003. Introduction to Database Systems
Jeffrey Ullman and Jennifer Widom, 1997.fundamental concepts of database systems.
Morgan Kaufmann, 2002. Relational database design principles.
Zanini, A. (2024, 10 30). *SQL DDL: The Definitive Guide on Data Definition Language*.

Web links

https://beginner-sql-tutorial.com/sql-grant-revoke-privileges-roles.htm
https://www.dbvis.com/thetable/sql-ddl-the-definitive-guide-on-data-definition-language/
www.geeksforgeeks.org. (2024, 10 30). *SQL Joins*. https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/


SQL

Indicative contents
4.1 Enforcement of data access control
4.2 Management of Auditing and logging
4.3 Implementation of Data encryption
4.4. Configuration of database backup and restore

Key Competencies for Learning Outcome 4: Implement Database Security

Knowledge	Skills	Attitudes
• Description of database security • Description of data control access • Description of auditing and logging • Description of data encryption • Introduction of data backup and restore	• Enforcing data access control • Managing Auditing and logging • Implementing Data encryption • Configuring database backup and restore	• Being Creative • Having Innovative skills • Having Adaptability skills • Being Flexible • Having Teamwork • Having self Confidence

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 • Browsers	• Internet

Indicative content 4.1: Enforcement of Data Access Control

Duration: 4hrs

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.: Introduction to database security

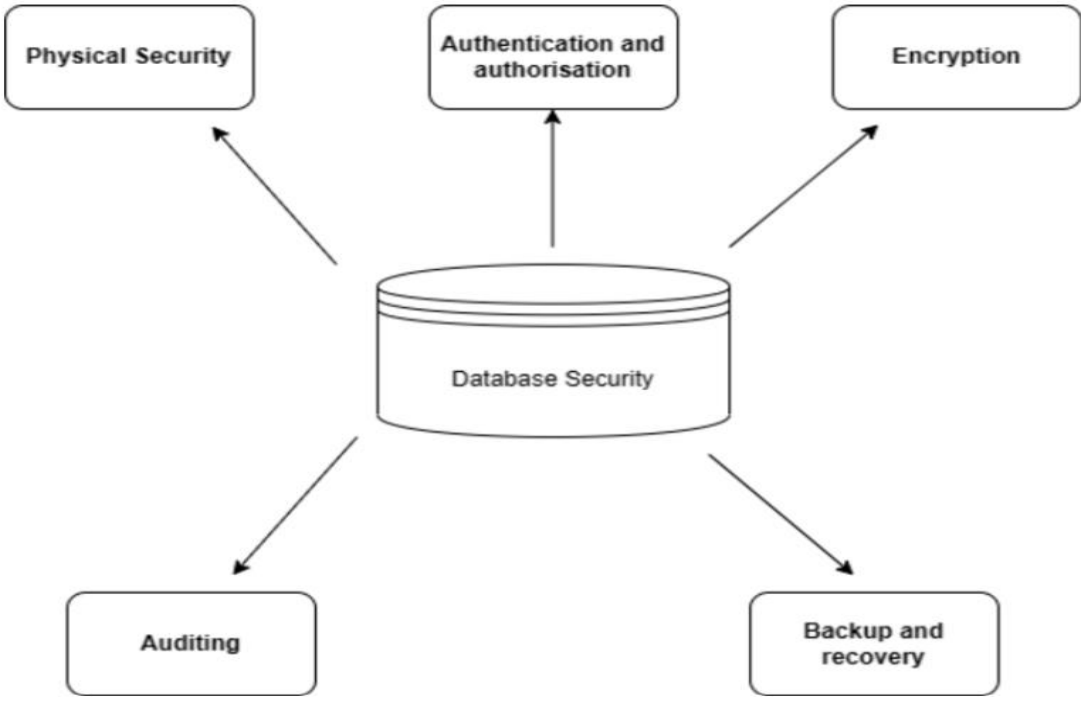
- **Database security**

  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.

I.**Types of database security**



1. 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.

2. 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.

3. 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.

4. 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.

5. 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

I. Data access control

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.**

II. 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**

1. 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.

2. 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.

3. 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.

4. 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:

1. **Public Data**:
   - Information that is freely available to the public.
   - No risk if disclosed.
   - Example: Marketing materials, public reports, press releases.
2. **Internal Data**:
   - Information meant for internal use within an organization.
   - Limited risk if disclosed to outsiders.
   - Example: Internal memos, internal policy documents, employee directories.
3. **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.
4. **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.

1. 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.

I. 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.

II. 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.

2. 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.

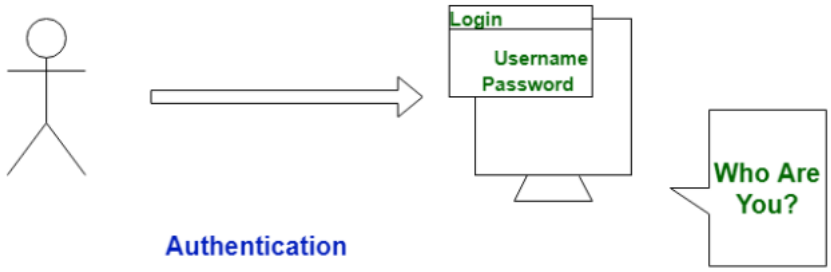
— **ALTER**: Modify an object (e.g., change a table's structure).

— **REFERENCES**: Create foreign key relationships between tables.

— **CONTROL**: Full control over a database object (similar to ownership).

III.Authentication

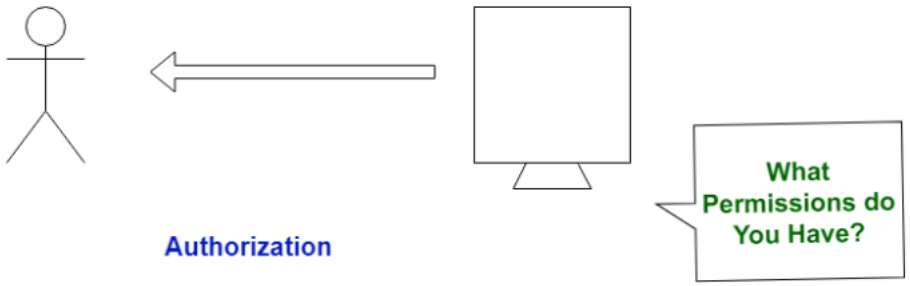
Authentication is the method of verifying the identity of a consumer or system to ensure they're who they claim to be. It involves checking credentials which include usernames, passwords, or biometric information like fingerprints or facial recognition. This step is vital for securing access to systems, programs, and sensitive records. By confirming identities, authentication saves you from unauthorized entry and protects you against safety breaches.



IV.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 in SQL Server

Task:

1: Read key **reading 4.1.3**

2: Read carefully the task described below:

Perform the following tasks using SQL Server Management Studio (SSMS)

XYZ 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 first step is to **identify the existing user accounts** in the system and audit their current permissions to ensure they align with best practices. Next, you will need to **create new logins and users** for additional staff, including the admissions and academic departments, ensuring that each user has appropriate access. You will also need to **assign server-level and database-level roles**, such as db_owner, db_datareader, and db_datawriter, to these users based on their job responsibilities.

Furthermore, you are required to **configure SQL Server authentication settings**, ensuring a mix of Windows and SQL Server authentication for flexibility, while also implementing strong password policies and enforcing the use of Multi-Factor Authentication (MFA). Once the configurations are in place, you will **test the authentication system** by logging in with newly created accounts to verify that access

control is functioning as expected. Finally, ongoing responsibilities include **monitoring and maintaining user accounts** by regularly reviewing their privileges, deactivating accounts for users who no longer need access, and ensuring compliance with the university's data security policies.

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. Below is a step-by-step guide:

Step 1: Authentication in SQL Server

SQL Server supports two types of authentications:

1. **Windows Authentication**: Uses Active Directory (AD) credentials. The user is authenticated by Windows, and SQL Server trusts that authentication.
2. **SQL Server Authentication**: Uses a username and password created within SQL Server. This is independent of Windows credentials.

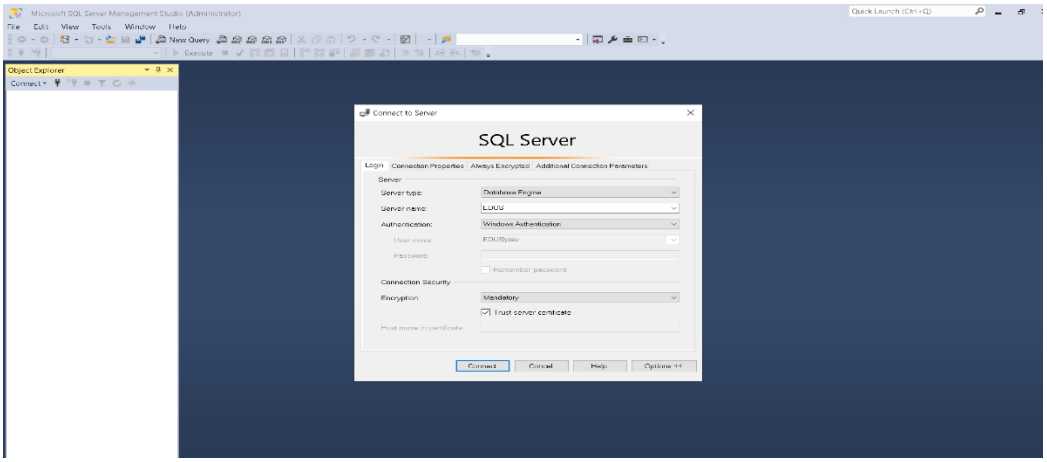
When you connect to SQL Server using SSMS, you choose the authentication mode:

- **Windows Authentication**: Choose this if you're using your Windows credentials.
- **SQL Server Authentication**: Enter the SQL Server username and password.

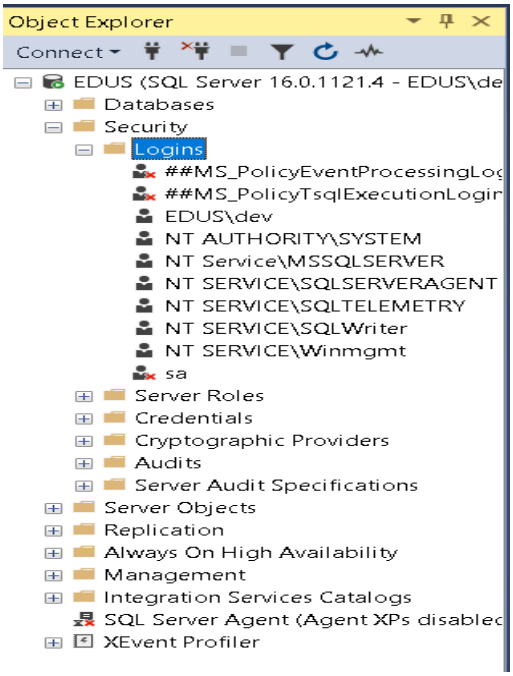
Step 2: Creating a New Login in SQL Server

A **login** is a security principal at the SQL Server instance level. To create a new login in SSMS:

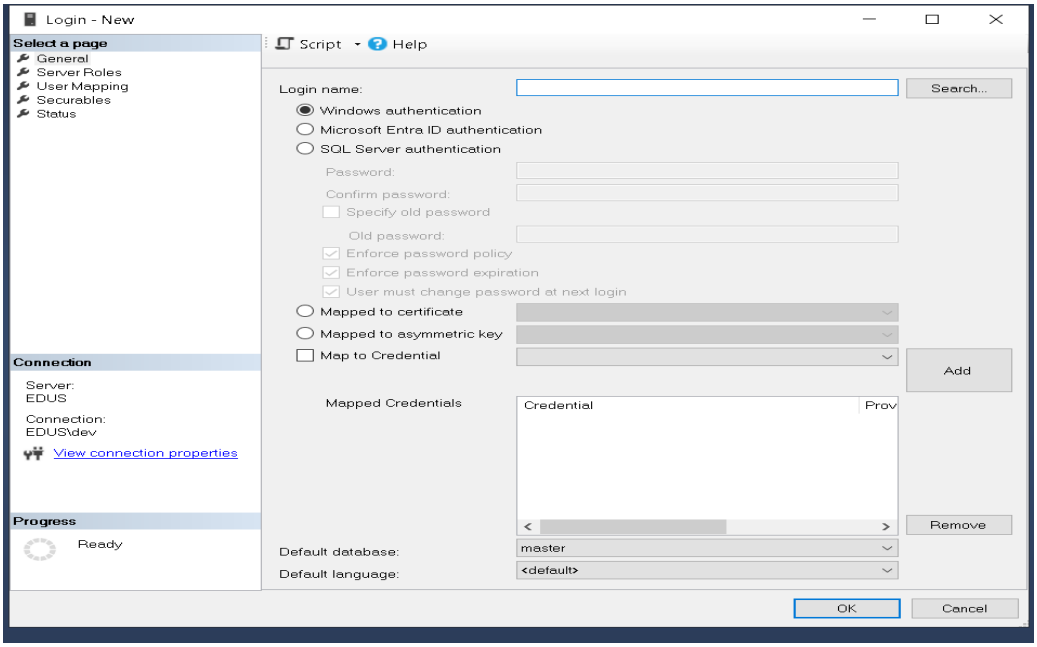
1. **Open SQL Server Management Studio (SSMS)** and connect to your SQL Server instance.



2. In **Object Explorer**, expand the **Security** folder.



3. Right-click on **Logins** and select **New Login...**.



Creating a Login Using Windows Authentication

- In the **Login - New** window, under the **General** tab:

o 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.

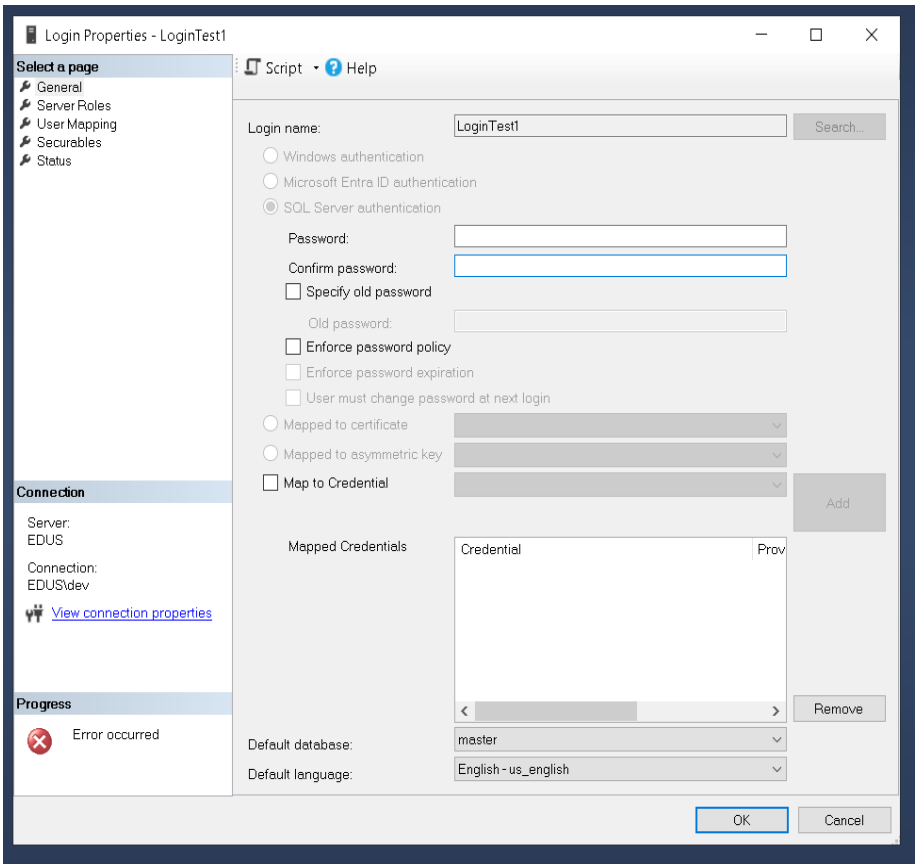
o Select **Windows Authentication**.

- Configure user mapping(check db owner), if needed, and click **OK**.

2.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.

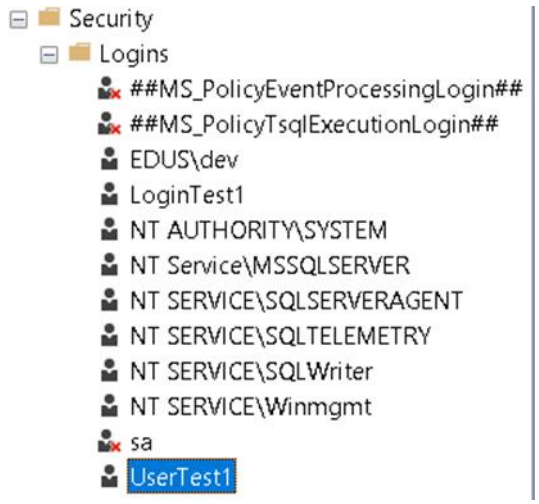


- Configure other settings like **Default Database** (optional), **Server Roles**, and **User Mapping**.
- Click **OK** to create the login.

3.Grant Permissions to the New User

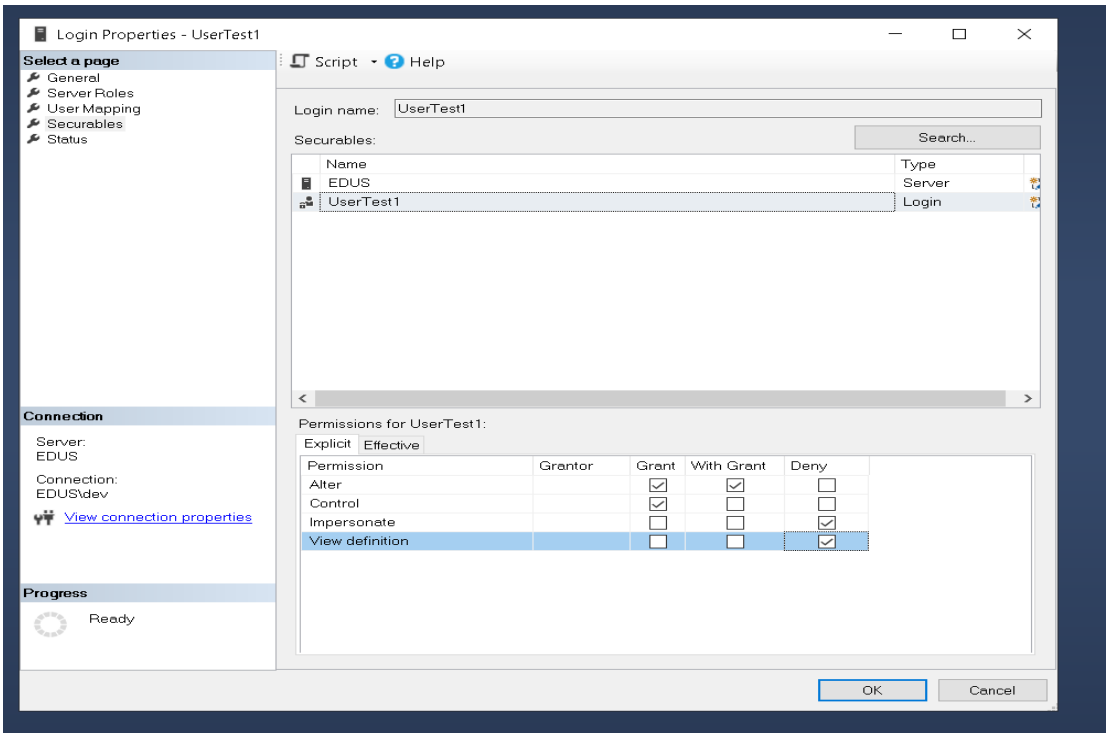
To grant specific permissions to the new user:

✓ **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.

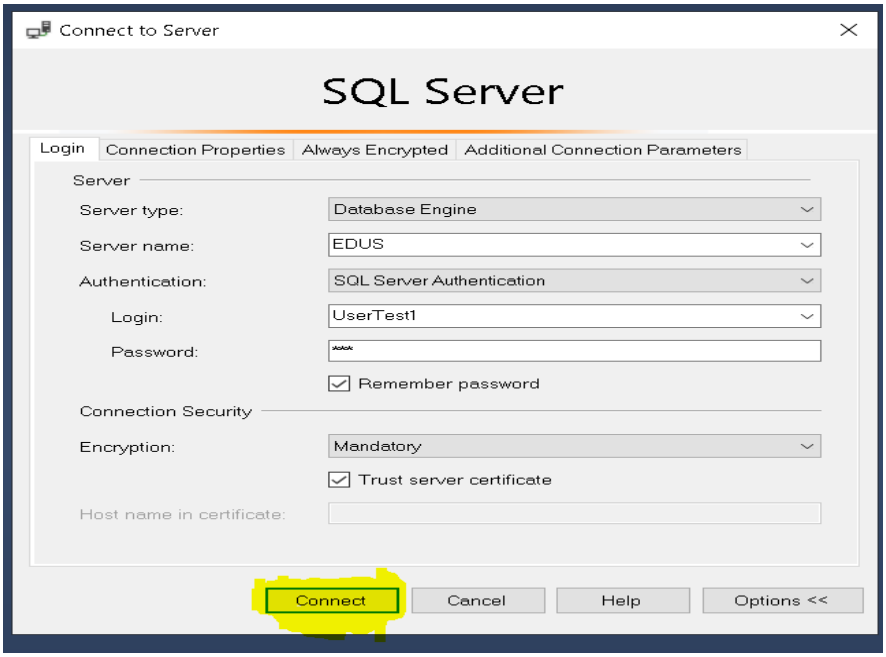


✓ Select the specific permissions (e.g., SELECT, INSERT, UPDATE, DELETE) and click **OK**.

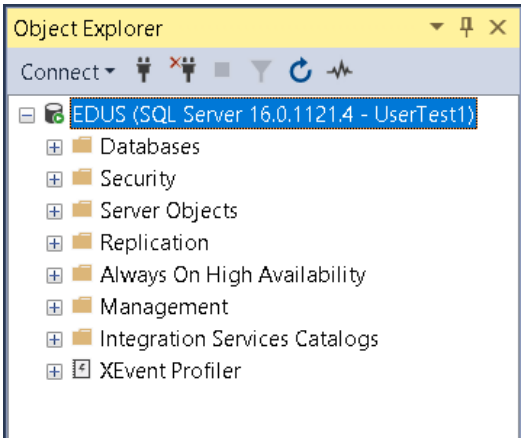
4.Test the Authentication System

To test the authentication system, log in to SSMS using the new login credentials and verify that the permissions are correctly set.

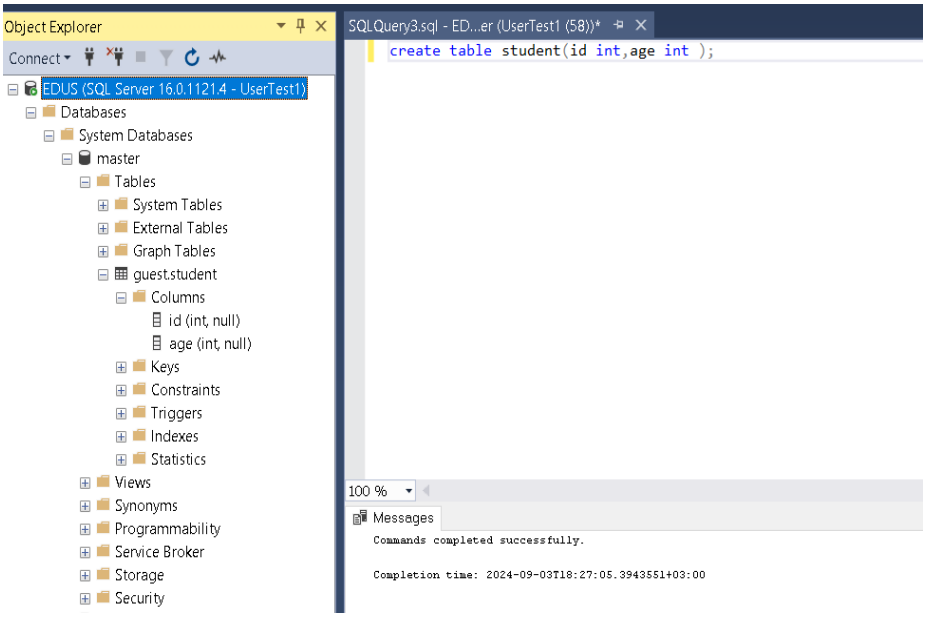
- Test a User Login:
  - ✓ Open **SSMS** and click **Connect**.
  - ✓ Choose **SQL Server Authentication** or **Windows Authentication**, depending on how the login was created.
  - ✓ Enter the **Login Name** and **Password**.



1. 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.



5. 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

2: Read the below 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 first task is to create roles to manage user permissions efficiently. Begin by creating a server-level role named CustomServerRole and assign an appropriate owner. Then, 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 such as ALTER, CONTROL, and CONNECT to CustomServerRole. For Test1DBRole, assign database-level permissions including SELECT, INSERT, UPDATE, and DELETE within the Test1 database.

Next, 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 such as selecting data, updating a table, and attempting to create a database to ensure the user permissions are functioning correctly as defined.

Finally, set up an ongoing process to monitor and maintain these roles and permissions. Regularly review user roles, permissions, and assignments to ensure they remain aligned with organizational needs. Remove any unused or outdated roles and establish auditing to track user activity, helping to detect any unauthorized actions or potential security breaches.

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

1. 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.

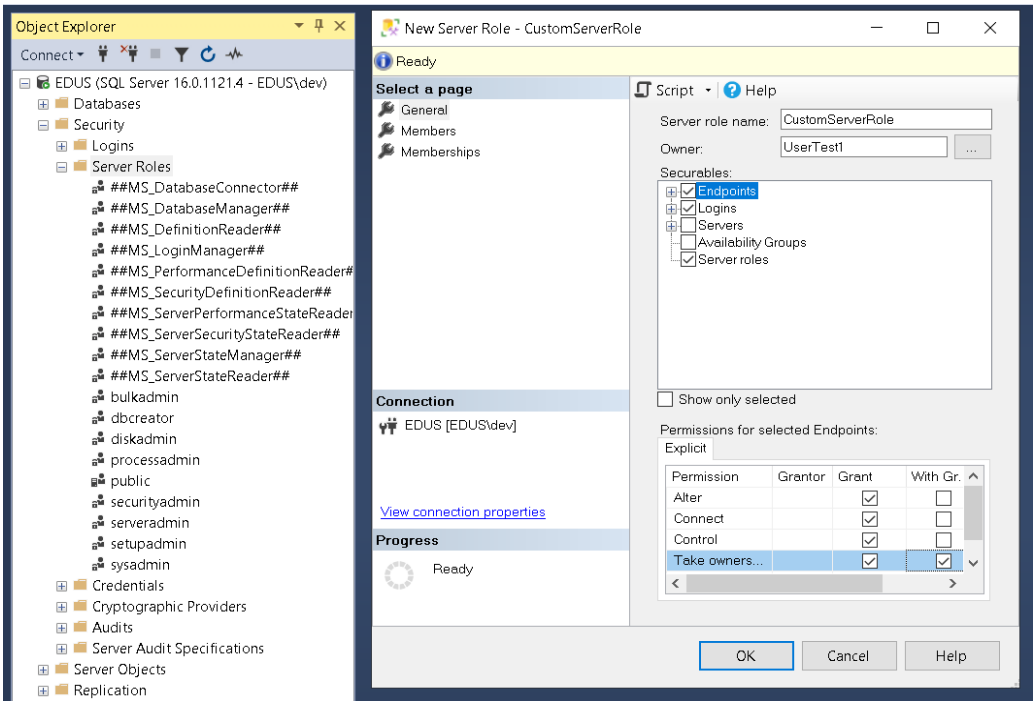
2. Create Roles

You can create roles at both the **server level** (server roles) and **database level** (database roles).

Create a Server Role

1. **Open SSMS** and connect to your SQL Server instance.
2. In **Object Explorer**, expand the **Security** folder.
3. Right-click on **Server Roles** and select **New Server Role...**.
4. 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).



5. Click **OK** to create the server role.

Create a Database Role

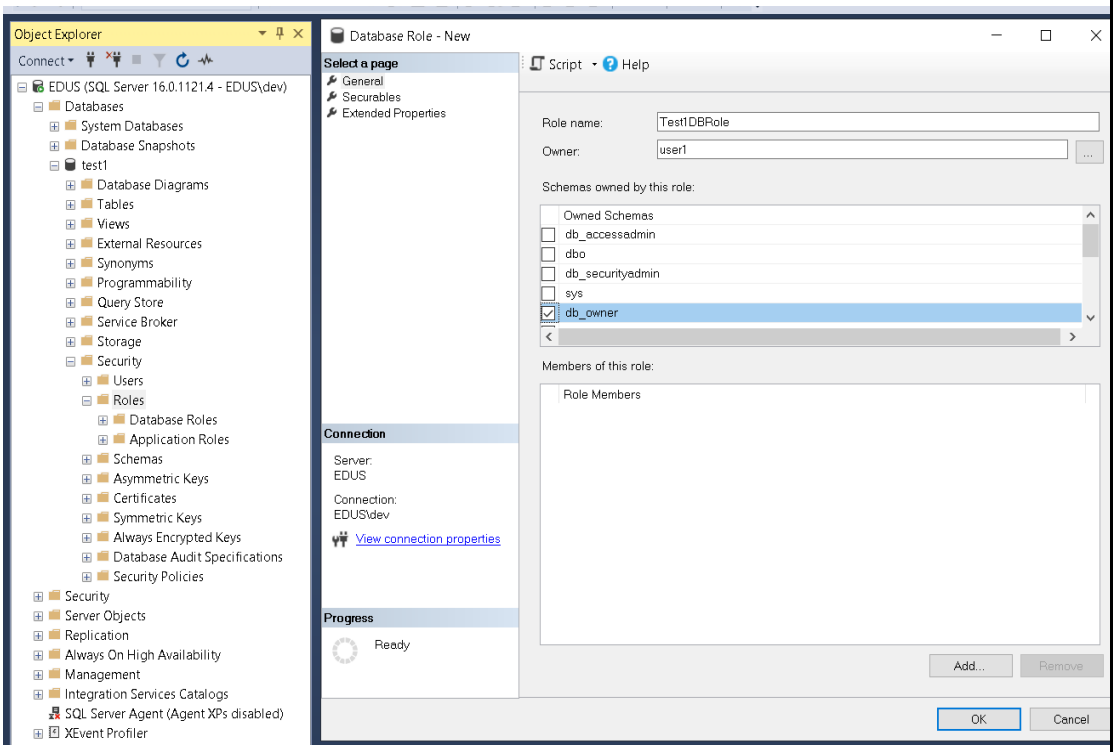
1. Expand the **Databases** folder, then expand the target database (e.g., Test1.)

2. Expand the **Security** folder under the database.

3. Right-click on **Roles > Database Roles** and select **New Database Role...**.

4. In the **Database Role - New** window:

- Enter the **Role Name** (e.g., Test1DBRole).
- Specify the **Owner** of the role.



5. Click **OK** to create the database role.

3. 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.).

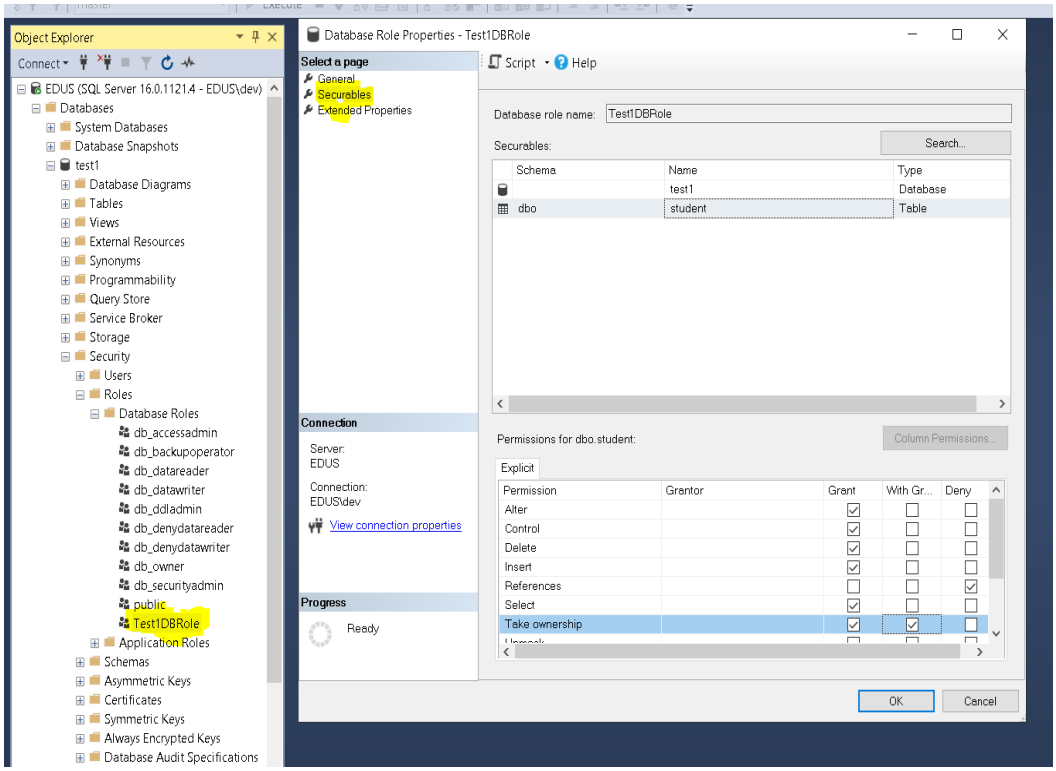
1. **Assign Server-Level Permissions to a Server Role**:

- 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.

2. **Assign 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.

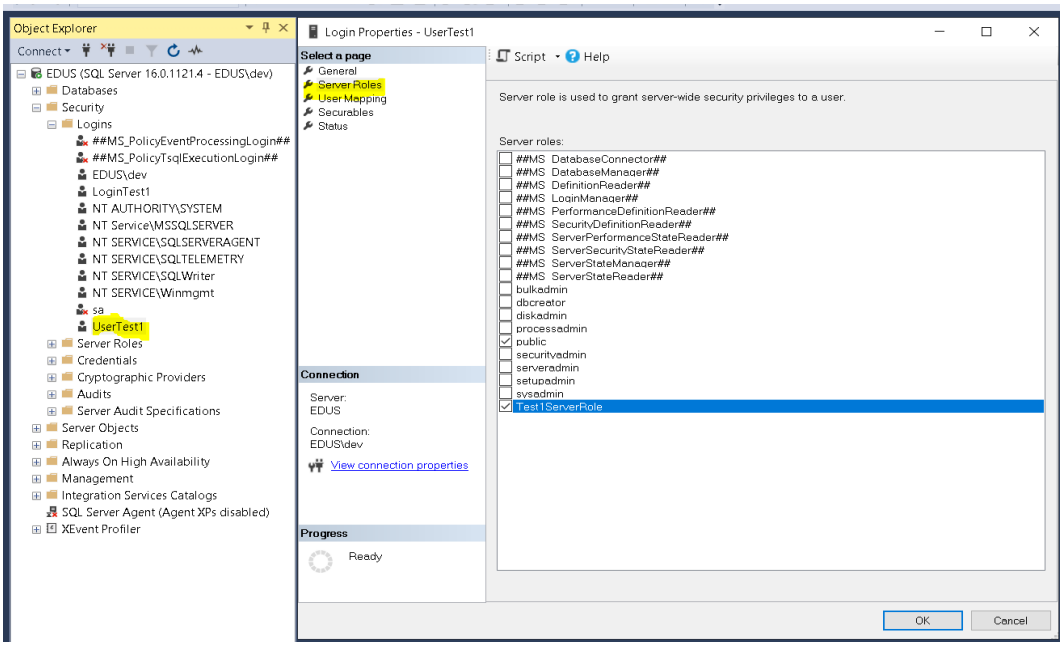
4. Assign Roles to Users

After defining roles and their permissions, assign these roles to users or logins.

1. **Assign a Server Role to a Login**:

- Expand the **Security > Logins** folder.

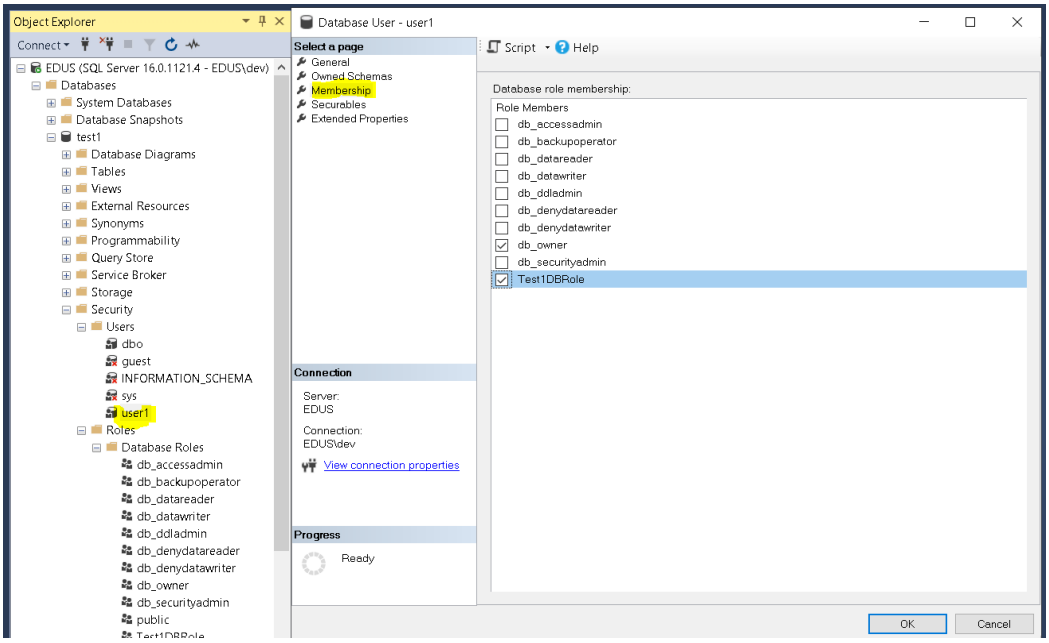
- Right-click on the login you want to assign a role to and select **Properties**.
- Go to the **Server Roles** tab.
- Check the server roles (e.g., Test1ServerRole) that you want to assign.



- Click **OK**.

2. **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**.

5. Test the Authorization System

To test the authorization system, log in using the user credentials and verify the permissions and roles assigned.

1. **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.

6. Monitor and Maintain

Monitoring and maintaining the authorization system involves regularly reviewing roles, permissions, and user assignments to ensure they meet security policies.

1. **Review Roles and Permissions**:

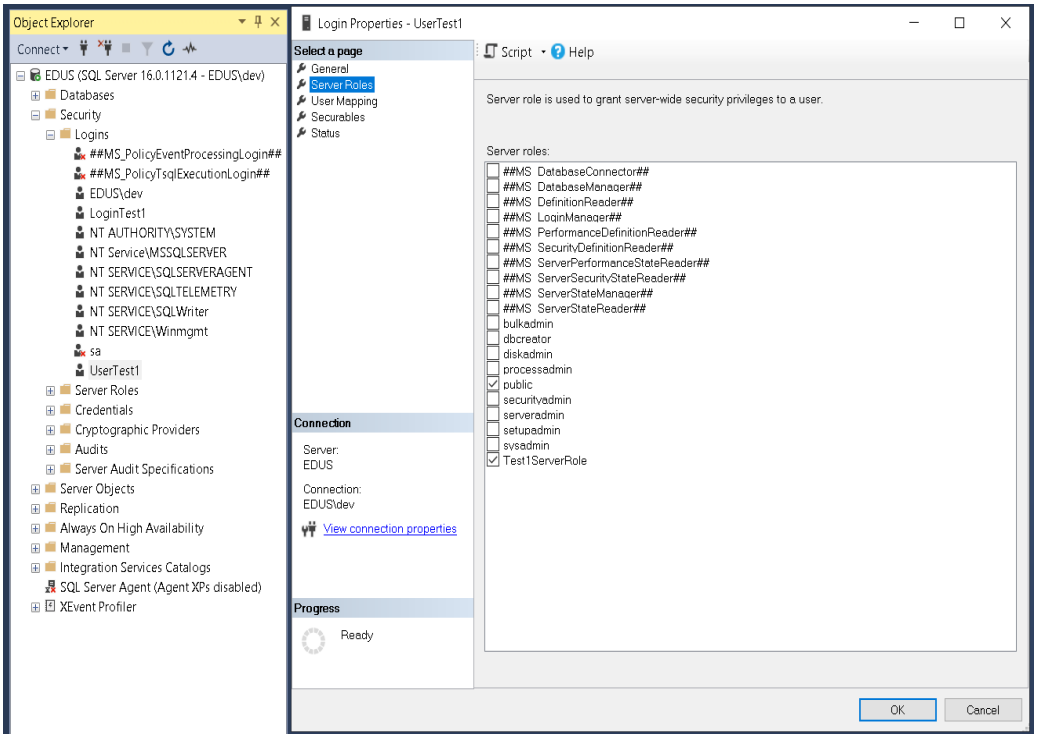
- Periodically review **Security > Server Roles** and **Security > Database Roles** to ensure that only necessary roles exist and that they have appropriate permissions.

- Right-click on roles and select **Properties** to view and adjust permissions if needed.

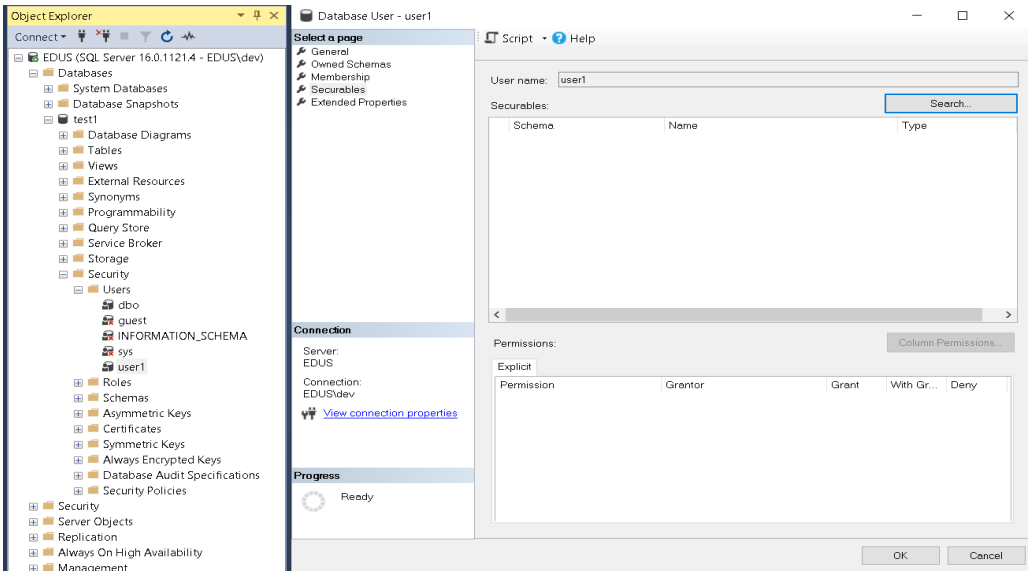
2. **Review User Role Assignments**:

- 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.

3. **Remove Unused or Outdated Roles**:

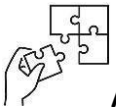
- Right-click on any role under **Server Roles** or **Database Roles** that is no longer needed and select **Delete** to remove it.

- Confirm the deletion.

4. **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.



Application of learning 4.1

You are the Database Administrator for "BrightFuture Academy," a school that 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).

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: 5hrs

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

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 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:

1. **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.
2. **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.
3. **Schema Auditing**:
   - Monitors structural changes to the database schema, such as creating or altering tables, views, procedures, or indexes.
4. **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.
5. **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:

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. To begin, you will identify the key events that need to be audited, such as login attempts, data modifications, permission changes, and schema changes. Next, you will configure the audit settings by creating a SQL Server Audit with a designated target (e.g., file, Application Log, Security Log). This will be followed by setting up a Server Audit Specification to track server-level events like FAILED_LOGIN_GROUP and a Database Audit Specification to monitor specific database-level activities like SELECT, INSERT, and UPDATE operations. Once configured, you will enable the audit and review the audit logs in SSMS to monitor activities and identify any unauthorized access or changes. Using the Audit Logs Viewer, you will analyze the audit data by filtering entries based on user activity, timestamps, and actions performed, paying close attention to any suspicious activities. 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.

4: Perform task2 described above.

5: 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

1. 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.

2. 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:

1. In SSMS, connect to your SQL Server instance.
2. Expand the **Management** folder and right-click on **SQL Server Logs**.
3. Select **Configure SQL Server Logs**.
4. Choose the **Log Destination** (e.g., File, Windows Application Log).
5. Set the **Maximum Number of Error Logs** and **Size** to manage disk space usage.
6. Click **OK** to save the configuration.

- Configure Extended Events for Logging:

7. In SSMS, expand the **Management** folder.
8. Right-click on **Extended Events** and select **New Session...**.
9. In the **New Session Wizard**, provide a **Session Name**.
10. Choose the events to capture (e.g., sql_batch_completed, rpc_completed).
11. Specify **Event Fields** to include additional data such as client_app_name or database_id.
12. Set the **Storage Target** (e.g., file or ring buffer) and specify the file path if

using a file.

13. Click **OK** to create and start the session.

3. Monitor Log Data

- Monitor SQL Server Error Logs:

1. In SSMS, expand the **Management** folder.

2. Right-click on **SQL Server Logs** and select **View SQL Server Log**.

3. Use filters to view specific logs (e.g., errors, warnings, or informational messages).

- Monitor Extended Events Logs:

4. Expand **Management > Extended Events > Sessions**.

5. Right-click on your logging session and select **Watch Live Data**.

6. View real-time data and apply filters for specific events or user activities.

4. Analyze Log Data

- Analyze SQL Server Error Logs:

1. Use the **View SQL Server Log** option to analyze entries based on severity, error codes, and user actions.

2. Identify patterns of repeated errors or security breaches.

- Analyze Extended Events Logs:

3. Export the extended events data to a file or table for detailed analysis.

4. Use SSMS or third-party tools to query and visualize log data for insights into database performance and security.

5. Archive Log Data

- Archive SQL Server Error Logs:

1. In SSMS, expand **Management > SQL Server Logs**.

2. Right-click on **SQL Server Logs** and select **Recycle SQL Server Error Log** to close the current log file and start a new one.

3. Regularly move older log files to a secure archive location to manage disk space and maintain records.

- Archive Extended Events Logs:

4. Set up a job in SQL Server Agent to periodically copy log files to an

archive directory.

5. Compress archived logs to save space and protect sensitive information.

6. 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

2: Read the below task

As the Database Administrator at DataSecure Solutions, you are responsible for configuring and managing auditing for the CompanyDB database using SQL Server Management Studio (SSMS) to ensure security and compliance. Your tasks begin with identifying the data to audit, focusing on key events such as login attempts, data modifications, permission changes, and schema alterations. Next, you will configure the audit settings by creating a SQL Server Audit with a designated target destination (e.g., file, Application Log, Security Log). You will then create a Server Audit Specification to track server-level events such as FAILED_LOGIN_GROUP and a Database Audit Specification to monitor database-level activities like SELECT, INSERT, and UPDATE operations. Once the audit and specifications are configured, you will enable the audit and regularly review the audit logs in SSMS to monitor activities, looking for any unauthorized access or changes. To analyze audit data, you will use the Audit Logs Viewer in SSMS, filtering entries based on user activity, timestamps, and actions performed, with a focus on identifying suspicious or unusual activities. If any unauthorized access or suspicious behavior is detected, you will take corrective actions by adjusting user permissions, strengthening security settings, or modifying the audit specifications to improve monitoring of relevant data.

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

1. Identify the Data That Needs to Be Audited

Determine which events and data need to be audited. This can include:

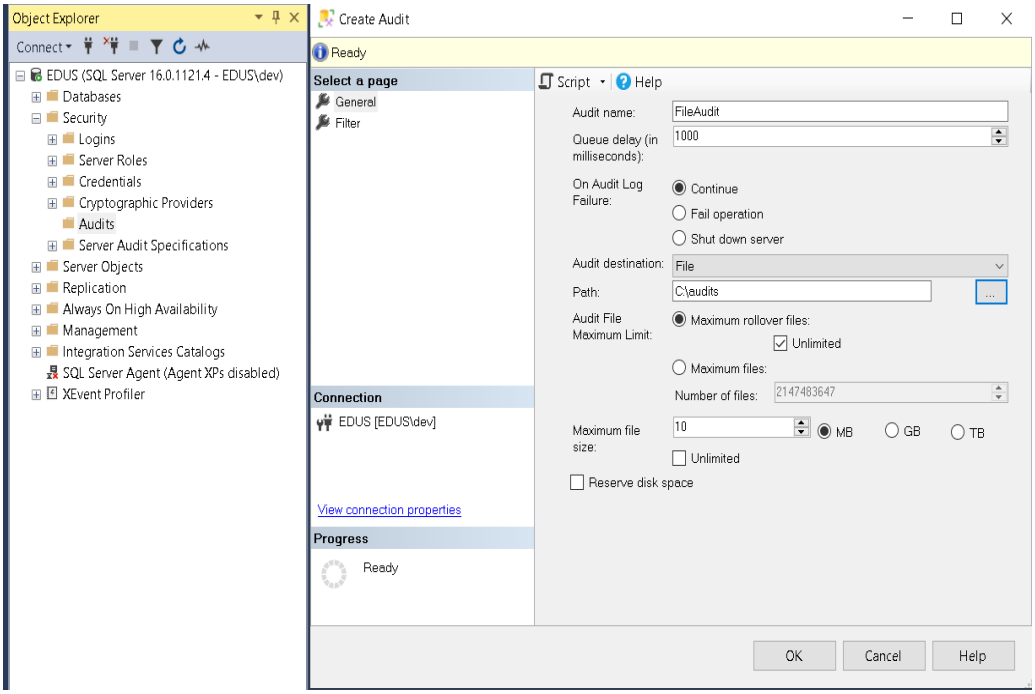
- Login attempts (successful or failed).
- Data modifications (e.g., INSERT, UPDATE, DELETE).
- Permission changes.
- Schema changes.

2. 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.

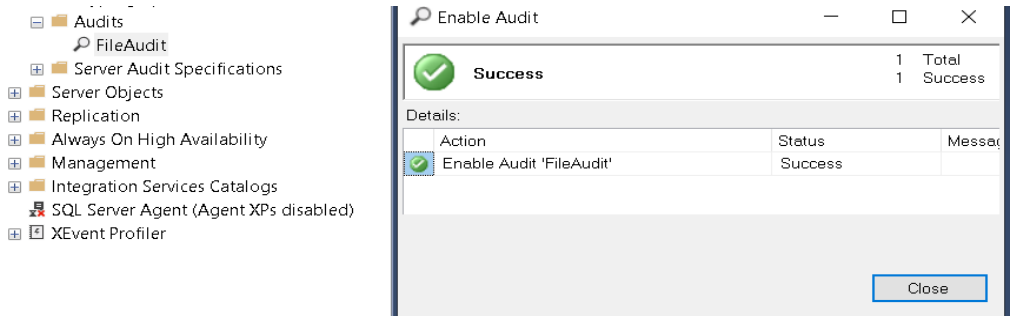
- **Create a SQL Server Audit**:

1. In **SSMS**, expand the **Security** folder.
2. Right-click on **Audits** and select **New Audit...**.
3. 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.



4. Click **OK** to create the audit.

5. Click on created audit (e.g., FileAudit) and select **Enable audit.**



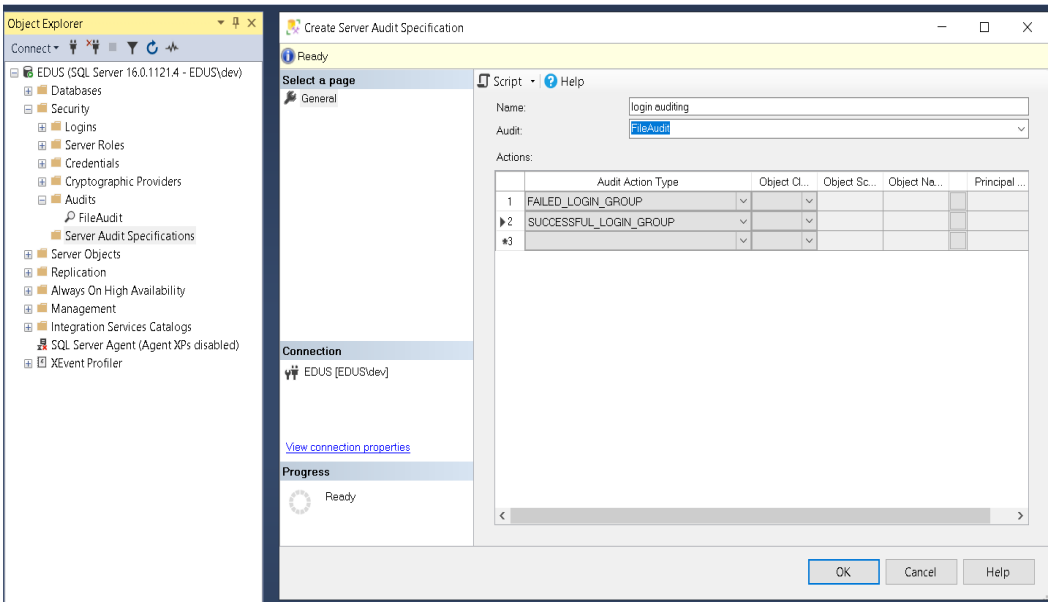
6. Click **close** to enable the audit.

- Create a Server Audit Specification:

7. Right-click on **Server Audit Specifications** under **Security** and select **New Server Audit Specification...**.

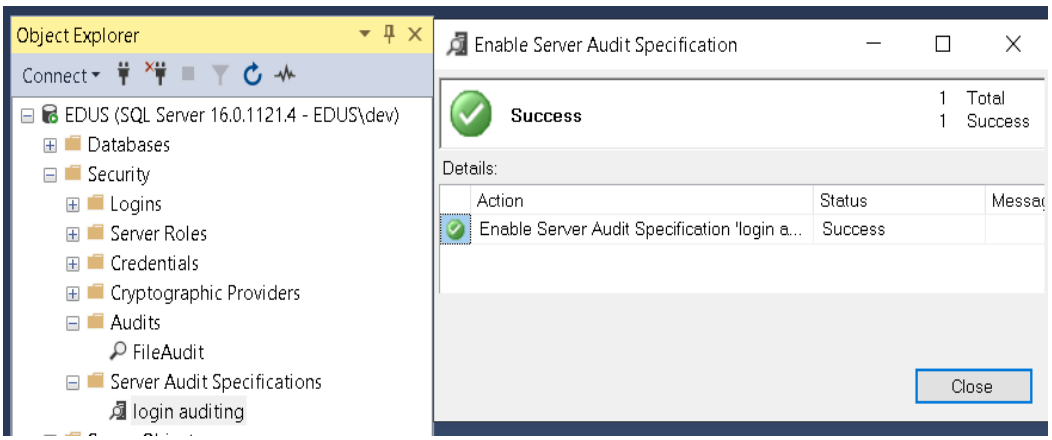
8. In the **Create Server Audit Specification** window:

   - Enter a **Name** for the specification.
   - Select the **Audit** you created earlier.
   - Choose the **Action Types** to audit (e.g., FAILED_LOGI_GROUP).



9. Click **OK**.

10. Click on created Server Auditing Specification (e.g., login auditing) and select **Enable Server Auditing Specification.**



11.

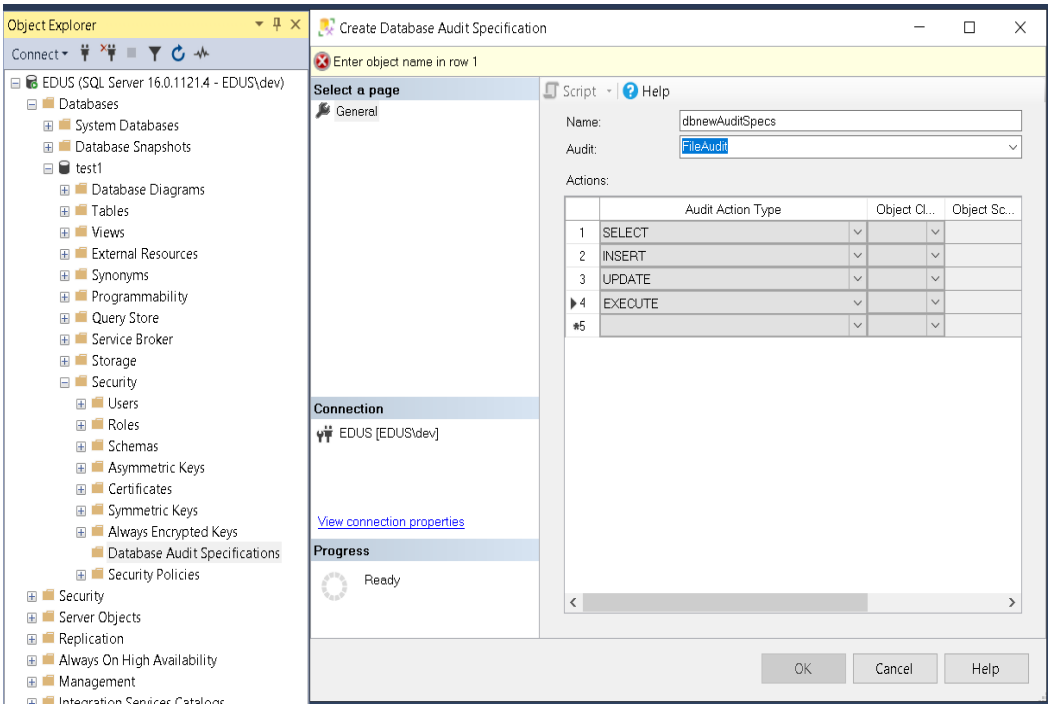
- Create a Database Audit Specification:

12. Expand **Databases > YourDatabase > Security**.

13. Right-click on **Database Audit Specifications** and select **New Database Audit Specification...**.

14. 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).



15. Click **OK**.

3. 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**.

4. Analyze Audit Data

- Analyze Audit Logs:

1. Use the **Audit Logs Viewer** in SSMS to analyze entries based on user activity, timestamps, and actions performed.

2. Look for unauthorized access, changes, or suspicious activities.

5. Corrective Action

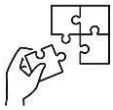
- Take Corrective Actions Based on Audit Data:
  - If unauthorized access or suspicious activity is detected, review user

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

- **Identify Logging Requirements**: Determine which events, like DELETE or UPDATE operations and failed logins, need to be logged.
- **Configure Logging Settings**: Set up SQL Server error logs and Extended Events sessions to capture specific activities.
- **Monitor Log Data**: Use SSMS to view and monitor error logs and Extended Events for tracking logged activities.
- **Analyze Log Data**: Review logs for patterns of unauthorized activities or failed login attempts using SQL queries.
- **Identify Data to Audit:** Determine events to be audited, such as login attempts, data modifications, and permission changes.
- **Configure Audit Settings:** Set up SQL Server Audit, Server Audit Specification, and Database Audit Specification for targeted events.
- **Enable and Review Audits:** Enable audit settings and review audit logs in SSMS for monitoring activities.
- **Analyse Audit Data:** Use SSMS to analyse audit logs for user activities and detect any unusual actions.
- **Take Corrective 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, you will identify logging requirements, focusing on tracking events like DELETE operations on the Grades table, UPDATEs on the Students table, and failed login attempts. Then, you will 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 will help you identify irregularities or suspicious activities, while log data analysis using SQL queries will assist in identifying patterns of failed logins or unauthorized actions. You will 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, you will identify key data to be audited, such as login attempts, data modifications (INSERT, UPDATE, DELETE), and permission or schema changes. Using SSMS, you will configure audit settings by creating a SQL Server Audit with a target destination (e.g., file, Application Log), link it to a Server Audit Specification to track server-level events like FAILED_LOGIN_GROUP, and set up a Database Audit Specification for auditing SchoolDB-level activities such as SELECT, INSERT, UPDATE, and DELETE operations, ensuring that all critical data access and modifications are monitored effectively.

Indicative content 4.3: Implementation of Data encryption

Duration: 4hrs

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

I. Data encryption

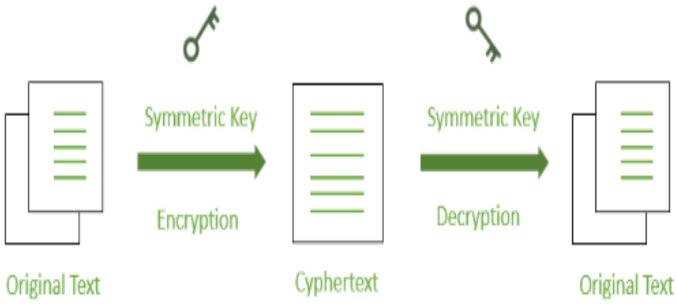
Data Encryption is a method of preserving data confidentiality by transforming it into ciphertext, which can only be decoded using a unique decryption key produced at the time of the encryption or before it. The conversion of plaintext into ciphertext is known as encryption.

II. 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.

1. Symmetric Encryption

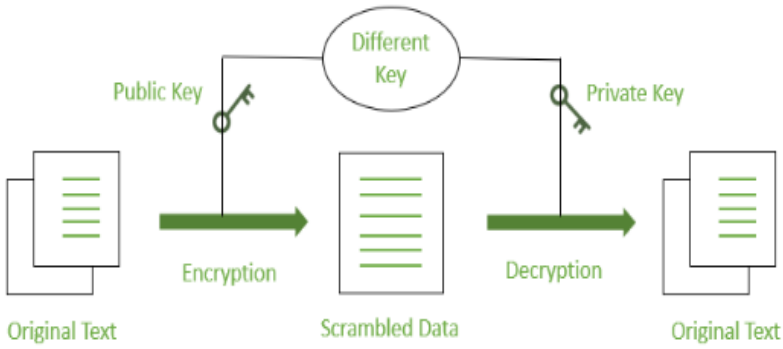
Symmetric Encryption: In symmetric encryption, the same key is used for both encryption and decryption. Popular algorithms include AES (Advanced Encryption Standard), DES (Data Encryption Standard), and Blowfish.



Symmetric Encryption

2. 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

3. 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

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, such as customer records and transaction histories stored in the database. Using **AES (Advanced Encryption Standard)**, encrypt the data to ensure that only users with the correct key can access or decrypt the information.

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.

By applying these encryption techniques, you will help protect sensitive data from unauthorized access and ensure secure communications and data integrity across the organization.

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

1. Description of Data Encryption

Data encryption in SQL Server involves converting data into a secure format that can only be read by authorized users with the correct decryption key. This is essential for protecting sensitive data, ensuring data privacy, and complying with security regulations.

Types of Data Encryption in SQL Server:

- Symmetric Encryption: Uses the same key for both encryption and decryption.
- Asymmetric Encryption: Uses a public key for encryption and a private key for decryption.
- Hashing: Converts data into a fixed-size string of characters, which is irreversible and used for integrity verification.

2. Application of Encryption Techniques

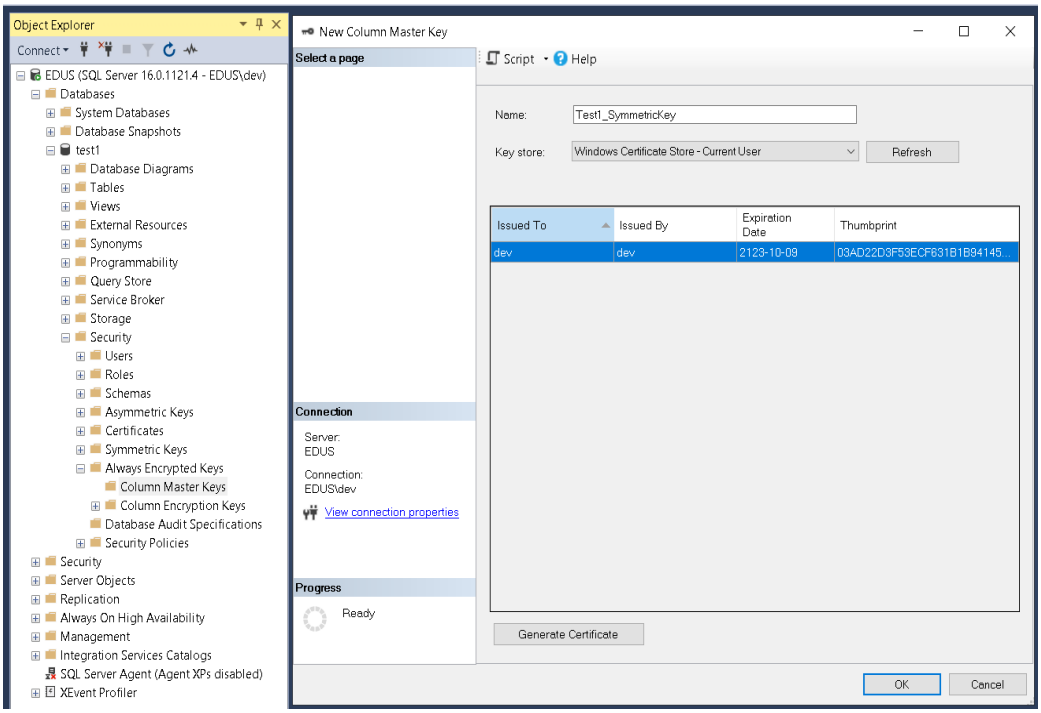
To apply encryption in SQL Server, you need to configure the database master key, certificates, and encryption keys. This can be done using SSMS's graphical interface.

A. Symmetric Encryption

Symmetric Encryption is straightforward and faster than asymmetric encryption. It is suitable for encrypting large amounts of data.

Steps to Implement Symmetric Encryption:

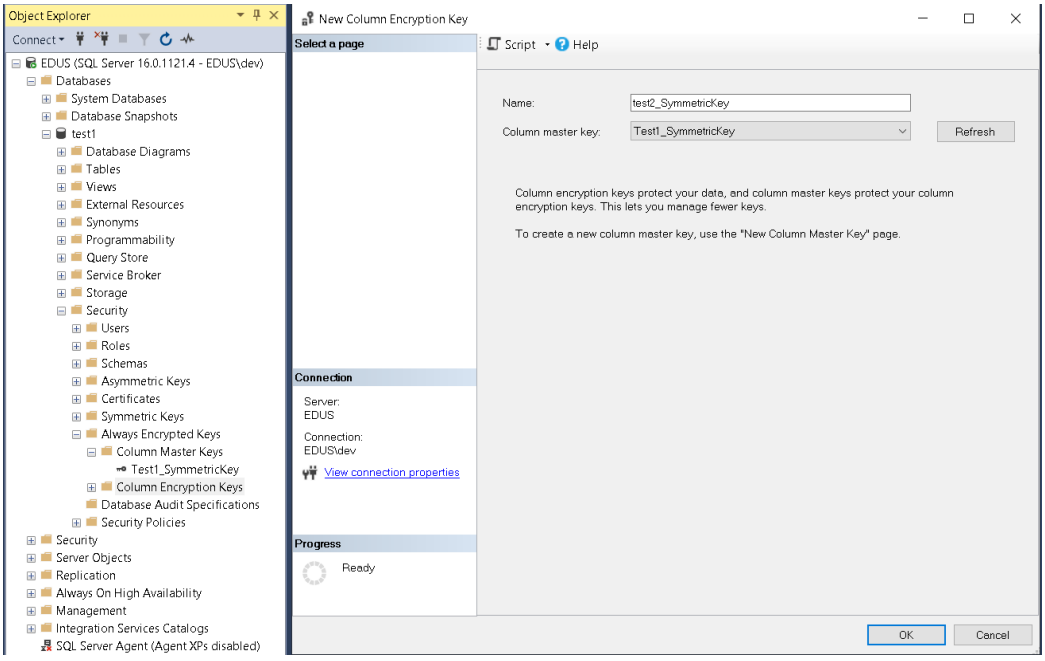
1. **Create a Master Key**:
   - Open SSMS and connect to your SQL Server instance.
   - In **Object Explorer**, expand the **Databases** folder, and then expand your target database.
   - Expand **Security > Always Encrypted Keys > Column Master Keys**.
   - Right-click on **Column Master Keys** and 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.

2. **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.

3. **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.

B. 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:

1. **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.
2. **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.

C. 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:

1. **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.

3. Monitor and Maintain Encryption

After implementing encryption, it's essential to monitor and maintain the encryption keys, certificates, and encrypted data.

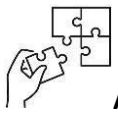
1. **Backup Encryption Keys and Certificates**:
   - Regularly back up your encryption keys and certificates to avoid data loss in case of server failure.

- Right-click on **Always Encrypted Keys** or **Certificates** and choose **Export...**.

2. **Review Encryption Configurations**:
   - Periodically review the encryption settings to ensure compliance with security policies.
3. **Rotate Keys and Certificates**:
   - As a best practice, regularly rotate encryption keys and certificates to enhance security.

Points to Remember

- Describe Data Encryption Methods: Explain the importance of data encryption in SQL Server and describe symmetric, asymmetric, and hashing methods.
- Implement Symmetric Encryption: Use SSMS to create a master key, set up a symmetric key, and encrypt a column with the symmetric key.
- Implement Asymmetric Encryption: Create an asymmetric key in SSMS and encrypt data using the created asymmetric key.
- Apply Hashing for Data Integrity: Create a computed column using a hashing function (e.g., HASHBYTES) to ensure data integrity.
- Monitor and Maintain Encryption: Backup encryption keys and certificates, review encryption configurations, and rotate keys and certificates regularly.



Application of learning 4.3

As the Database Administrator for "BrightFuture Academy," you need to ensure the confidentiality and integrity of sensitive data stored in the **SchoolDB** database, such as student personal information, grades, and staff details. 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: 6hrs

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. Differentiate backup form restore

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

Description of database backup and restore

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

I.Types of Backups:

1) **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.

2) **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.

3) **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.

II. Backup Strategies

1) **Hot Backup (Online Backup):** Performed while the database is still running and accessible by users.

2) **Cold Backup (Offline Backup):** Performed when the database is shut down and not accessible by users.

3) **Warm Backup:** A combination where parts of the database may be online, but most operations are paused or limited.

4) **Frequency of Backups:** Regular backups are essential, and organizations determine frequency based on factors like database size, transaction volume, and criticality of data.

III. **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**

I. **Types of Restores:**

1. **Full Restore:**

— Restores the entire database from a full backup.

— Suitable when the entire database is damaged or lost.

2. **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.

3. **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.

4. **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).

II. **Restore Process Steps:**

1. **Identify the Backup:** Choose the correct backup file(s) based on the recovery point and type of failure.

2. **Restore Data:** Load the backup file into the database system.
3. **Apply Transaction Logs (Optional):** If necessary, apply logs to roll the database forward to a specific point in time.
4. **Verify Data Integrity:** Ensure the restore was successful and that the database is functioning as expected.
5. **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:

Configuring Database Backup and Restore in SQL Server Using SSMS

The objective of this is to understand and demonstrate how to configure various types of database backups and perform restores in SQL Server using SQL Server Management Studio (SSMS), ensuring effective data recovery and minimizing data loss. First, you will provide a brief introduction to data backup and restore within SQL Server, highlighting the importance of backups for data protection and recovery.

You will then explain different backup methods: full backups that capture the entire database, differential backups that record changes since the last full backup, and transaction log (incremental) backups that capture changes since the last incremental or full backup. Next, you will propose a backup schedule, recommending weekly full backups, daily differential backups, and hourly transaction log backups for databases using the Full recovery model. Using SSMS, you will perform a full backup, a differential backup, and a transaction log backup, walking through each process step-by-step.

For recovery, you will demonstrate how to perform a full database recovery, rollback recovery using transaction logs, and point-in-time recovery to undo changes made after a certain point. To ensure the reliability of your backup and recovery strategy, you will regularly test restores in a test environment, verify the integrity of backup files using the "Verify Backup Integrity" option, and maintain detailed documentation of the backup schedules, types, and recovery procedures for consistency and accuracy during emergency recovery situations

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

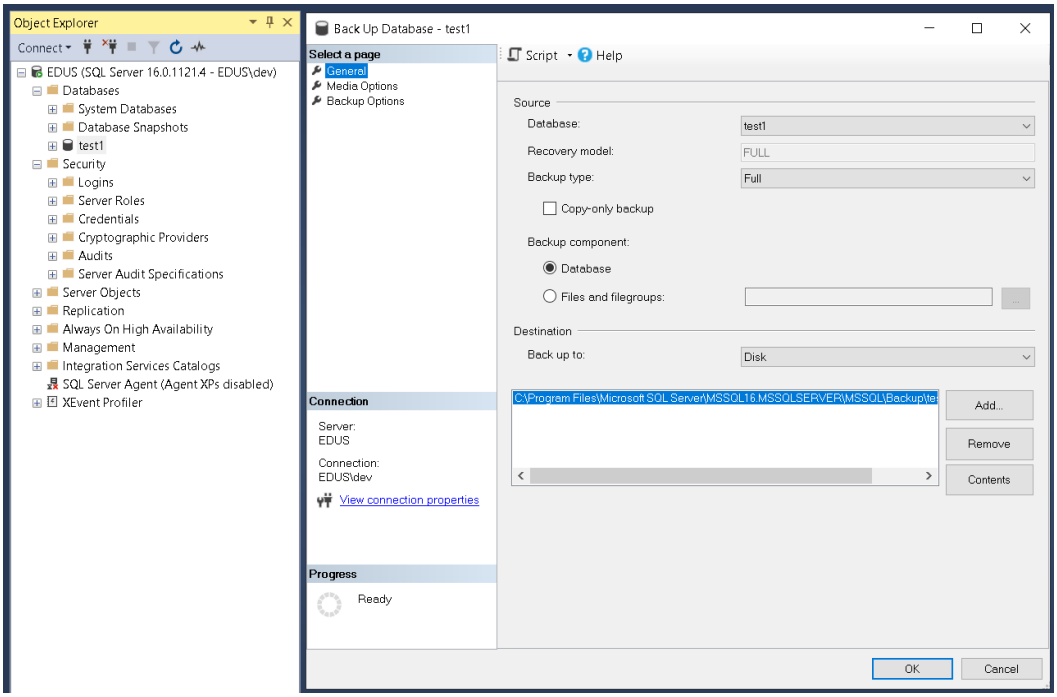
Configuration of Database Backup and Restore in SQL Server Using SSMS

1. Create Backup Using SSMS

To create backups in SSMS, follow these steps:

A. 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.



2. Click **OK** to start the full backup.

B. 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.

C. Transaction Log Backup (Incremental Backup)

1. Right-click on the database, hover over **Tasks**, and select **Back Up...**.
2. In the **Back Up Database** window:
   - Set the **Backup Type** to **Transaction Log**.
   - Ensure the **Backup Destination** is set.
3. Click **OK** to start the transaction log backup.

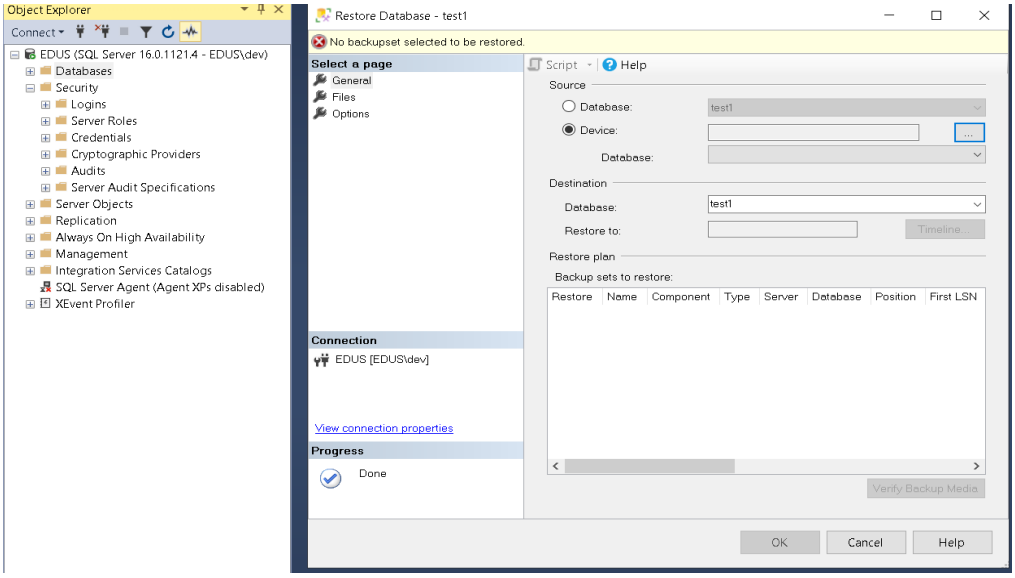
5. Perform Recovery Methods

SQL Server supports several recovery methods, depending on the type of backup used and the recovery scenario:

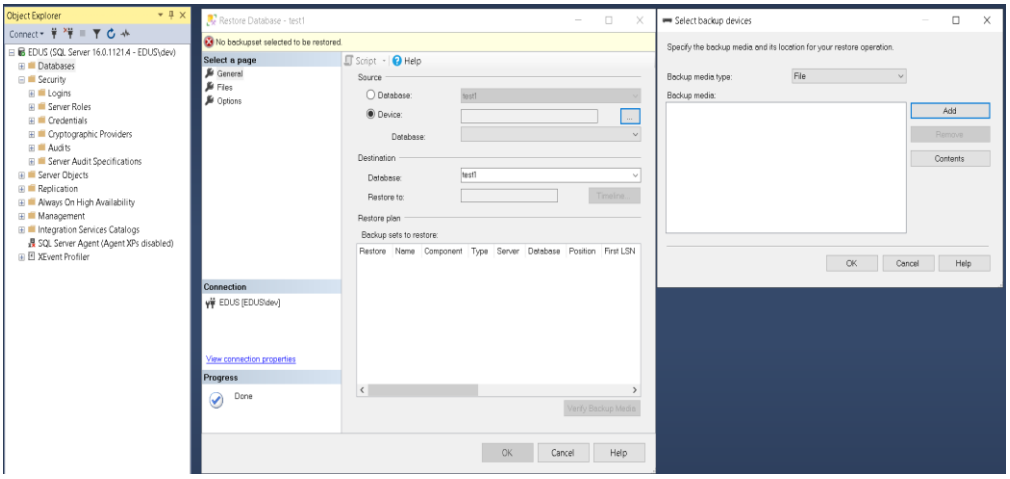
A. Full Database Recovery

To restore a full database backup:

1. Right-click on the **Databases** folder and select **Restore Database...**.



2. 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**.

3. Click **OK** to start the restoration process.

B. Rollback Recovery

Rollback recovery involves reverting the database to a specific point before unwanted transactions or errors.

1. Use the **Transaction Log Restore** options in the **Restore Database** window.
2. Select **Transaction Log Backups** to apply logs up to a specific point.

C. 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.

1. In the **Restore Database** window, select the **Timeline** option.
2. Choose **Point-in-time restore** and specify the exact date and time.
3. Click **OK** to restore the database to that specific point.

6. Test Your Backup and Recovery Plan

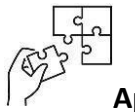
Regularly testing your backup and recovery plan is crucial to ensure it works correctly when needed.

1. **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.
2. **Validate Backups**:
   - Use the **Verify Backup Integrity** option in the **Back Up Database** window to check the validity of the backup files.
3. **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

- A **backup** is the process of creating a copy of the database, its data, and its configuration. While a **restore** is the process of retrieving data from a backup and applying it to a database, making it operational again.
- **Backup Importance:** Backups are crucial for protecting data and enabling recovery from accidental deletions, hardware failures, or other disasters.
- **Backup Methods:** Full backups capture the entire database, differential backups capture changes since the last full backup, and incremental backups capture changes since the last incremental or full backup.

- **Backup Schedule:** A recommended schedule includes weekly full backups, daily differential backups, and hourly transaction log backups for databases with the Full recovery model.
- **Backup Using SSMS:** Use SSMS to create full, differential, and transaction log backups.
- **Recovery Methods:** Restore databases from full backups, rollback changes using transaction log backups, and perform point-in-time recovery to restore to a specific point in time.
- **Testing and Documentation:** Regularly test backups and recovery plans, validate backup integrity, and document procedures for consistent and accurate recovery.



Application of learning 4.4

As 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 such as full backups, which capture the entire database including tables, data, and schema, differential backups that record changes since the last full backup, and transaction log backups that capture incremental changes since the last backup. Additionally, you will propose and configure a backup schedule that includes weekly full backups, daily differential backups, and hourly transaction log backups using SQL Server Management Studio (SSMS). In terms of recovery, you will 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. To ensure the integrity and availability of the SchoolDB database, you will regularly test the backup and recovery plan by restoring backups in a test environment, verifying the integrity of backup files using the "Verify Backup Integrity" option in SSMS, and maintaining detailed documentation of the backup schedules, methods, and recovery procedures for consistent and reliable recovery during emergencies.

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. Symmetric Encryption	A. Uses different keys for encryption and decryption
……	2. Full Backup	B. Checks data integrity
……	3. Role-Based Access Control (RBAC)	C. Complete snapshot of the database
……	4. Auditing	D. Same key for encryption and decryption
……	5. Point-in-Time Recovery	E. Tracks and logs events
……	6. Incremental Backup	F. Restores to a specific point in time
……	7. Hashing	G. Tracks only changes since the last full or incremental backup
……	8. Authentication	H. Verifying user identities
……	9. Logging	I. Capturing and storing system events and actions
……	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 such as SQL injection, unauthorized access, and data breaches, while ensuring access is restricted to authorized users and actions are logged. You will define roles such as FinanceManager and SupportStaff, assigning FinanceManager permissions to SELECT, INSERT, UPDATE, and DELETE on all tables, and SupportStaff permissions to SELECT on the Customers and Transactions tables. Next, you will establish access control policies based on these roles and classify data as Public, Confidential, or Highly Confidential.

In configuring authentication, you will create user accounts for the FinanceTeam and SupportTeam, 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. You will also configure logging settings to capture events like login attempts, data modifications, and permission changes, monitor and analyze logs for suspicious activity, and archive log data regularly. Further, you will configure auditing by setting up server and database audit specifications to track key activities. Data encryption techniques, including symmetric encryption for sensitive columns (e.g., CustomerSSN), asymmetric encryption for secure data transfer, and hashing for data integrity checks, will be applied.

Finally, the configuration of database backups will include performing full, differential, and transaction log backups, and you will test recovery methods such as full database recovery, rollback recovery, and point-in-time recovery to ensure data integrity and availability.

END

References

Books

Bhushan, B. (2024, 10 30). *Infosecurity Magazine*.

Raza, M. (2024, 10 30). *Audit Logging: A Comprehensive Guide*.

Satori. (2024, 10 30). *Access Control Policies: Definitions & Types*.

Shivanshu. (2024, 10 30). *Database Security*

Web links

geeksforgeeks.org. (2024, 10 30). *Database Recovery Techniques in DBMS*. Retrieved from www.geeksforgeeks.org: https://www.geeksforgeeks.org/database-recovery-techniques-in-dbms/

https://satoricyber.com/access-control/access-control-policies-definitions-types/#:~:text=Access%20control%20policies%20help%20define,roles%2C%20policies%2C%20or%20rules.

https://www.intellipaat.com/blog/database-security

www.infosecurity-magazine.com: https://www.infosecurity-magazine.com/blogs/how-to-backup-and-restore-database/

www.splunk.com: https://www.splunk.com/en_us/blog/learn/audit-logs.html

RTB | RWANDA TVET BOARD

October, 2024