



Database Management Systems Training

June 12 - 16, 2023

One Central Hotel
Cebu City, Philippines

Training Staff

- Worgie V. Flores
ETRACS Developer

Purpose of this Training

- ⦿ To provide a comprehensive understanding of database concepts, including data models and relational databases
- ⦿ Learn how to query and manipulate data using SQL
- ⦿ Database management and administrative tasks
 - Security measures and access control
 - Backup and recovery
 - Replication

Training Timeline

DATE	ACTIVITY	WHO SHOULD ATTEND
Day 1 - 6/12/23	<ul style="list-style-type: none">• Installation of MySQL Server• Configuration and Initial Setup of MySQL Server• Introduction to Databases• Introduction to Relational Databases• Introduction to SQL (Structured Query Language)	IT Personnel / DB Administrator
Day 2 - 6/13/23	<ul style="list-style-type: none">• Understanding Basic SQL Statements• Understanding Database Design Principles• Database Schema Design• Data Modeling	IT Personnel / DB Administrator
Day 3 - 6/14/23	<ul style="list-style-type: none">• Working with multiple tables: Joins, subqueries, and unions• Advanced SQL functions: Aggregate, String, Date• Indexing and query optimization techniques• Performance tuning	IT Personnel / DB Administrator

Training Timeline

Day 4 – 6/15/23	<ul style="list-style-type: none">• Introduction to database administration• User management and access control• Backup and Restore Database• Database Replication	IT Personnel / DB Administrator
Day 5 – 6/16/23	<ul style="list-style-type: none">• Continuation of Day 4• Open Forum• Closing	IT Personnel / DB Administrator

Why do we need a **database** ?

Potential Problems

⦿ Size

- You may thousands or millions of rows of data information

⦿ Accuracy

- Typing incorrect data information

⦿ Security

- You need to restrict access to the data

Potential Problems

⦿ Redundancy

- Having multiple copies of the same data will lead to conflict

⦿ Importance

- Disconnected, crash or lose important data

⦿ Overwriting

- Having more than one person overwriting the same data at a time

Solution

- To keep the data reliable, secured and maintainable, you to have a database.

What is a **database** ?

What is a **database** ?

- ⦿ A database is a collection of related data organized in a way that data can be easily accessed, managed and updated
- ⦿ A Database can be a software based or hardware based, with one sole purpose, storing data

What is a **DBMS** ?

- ④ DBMS stands for Database Management System
- ④ It is a software that allows users to manage and interact with databases
- ④ It provides a set of tools and functions to create, store, retrieve, update, and delete data in a database

What is a **DBMS** ?

- ④ It acts as an intermediary between the users or applications and the underlying database, handling tasks such as data organization, data integrity, security, and data access
- ④ DBMSs facilitate efficient and structured management of data, ensuring that it is stored, organized, and retrieved in a reliable and controlled manner

Examples of Popular DBMS

- ⦿ MySQL
- ⦿ Microsoft SQL Server
- ⦿ Oracle Database
- ⦿ IBM DB2
- ⦿ PostgreSQL

Characteristics of DBMS

Characteristics of DBMS

⦿ Data Independence

- DBMS provides data independence, separating the physical storage details from the way data is logically perceived and accessed
- This allows for modifications to the database structure without affecting the applications or user views that interact with the data

Characteristics of DBMS

⦿ Data Integrity

- DBMS ensures data integrity by enforcing constraints, rules, and validations to maintain the accuracy, consistency, and reliability of data
- It helps prevent data inconsistencies and errors through mechanisms like referential integrity, data type validation, and unique key enforcement

Characteristics of DBMS

● Data Security

- DBMS offers robust security features to protect sensitive data from unauthorized access, modification, or disclosure
- It includes mechanisms for user authentication, access controls, encryption, and auditing to ensure data privacy and comply with regulatory requirements

Characteristics of DBMS

⦿ Concurrent Access and Concurrency Control

- DBMS allows multiple users or applications to concurrently access and modify the data in the database
- Concurrency control mechanisms, such as locking, to ensure data consistency and prevent conflicts when multiple users attempt to modify the same data simultaneously

Characteristics of DBMS

⦿ Data Recovery and Backup

- DBMS provides mechanisms for data backup and recovery to protect against system failures, data corruption, or human errors
- It allows for regular backups and enables recovery to a previous consistent state in case of failures or disasters

Characteristics of DBMS

◉ Querying and Ad Hoc Retrieval

- DBMS supports structured query languages (e.g., SQL) that enable users to easily retrieve, manipulate, and analyze data in a database
- Users can perform ad hoc queries to retrieve specific subsets of data based on various criteria

Characteristics of DBMS

⦿ Scalability and Performance

- DBMS is designed to handle large volumes of data and support scalability as data grows
- It provides mechanisms for optimizing performance, such as indexing, query optimization, and caching, to ensure efficient data retrieval and processing

Characteristics of DBMS

⦿ Data Consistency and ACID Properties

- DBMS ensures data consistency and adherence to ACID (Atomicity, Consistency, Isolation, Durability) properties for reliable transaction processing
- ACID properties guarantee that database transactions are executed reliably, and the database remains in a consistent state even in the presence of failures.

Characteristics of DBMS

⦿ Data Sharing and Collaboration

- DBMS enables data sharing and collaboration among multiple users or applications
- It allows concurrent access to data while maintaining data integrity and providing mechanisms for managing concurrent updates and ensuring data consistency

Characteristics of DBMS

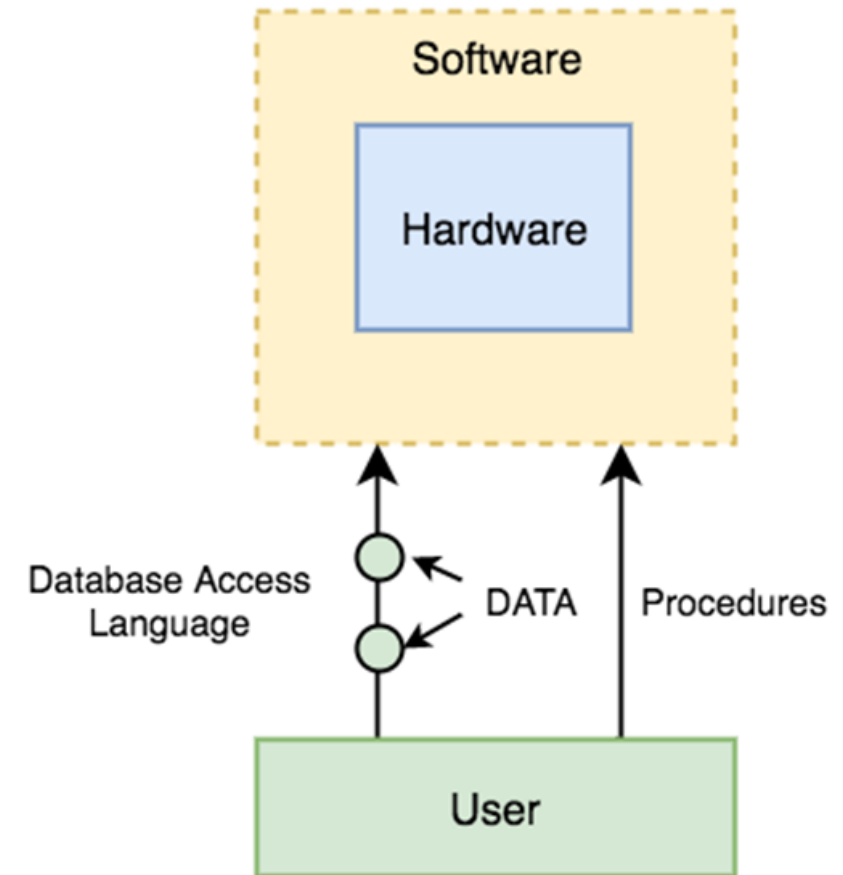
⦿ Centralized Control and Administration

- DBMS provides centralized control and administration capabilities
- Database administrators can manage user access, security permissions, backups, performance tuning, and other administrative tasks to ensure efficient and secure database operations

Components of DBMS

Components of DBMS

- Hardware
- Software
- Data
- Procedures
- Data Access Language
- Security



Components of DBMS

⦿ Hardware

- When we say Hardware, we mean computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory.

Components of DBMS

● Software

- This is the main component, as this is the program which controls everything
- The DBMS software is more like a wrapper around the physical database, which provides an easy-to-use interface to store, access and update data
- Capable of understanding the Database Access Language and interpret it into actual database commands to execute them on the DB

Components of DBMS

● Data

- Data is the resource, for which DBMS was designed.
- The motive behind the creation of DBMS was to store and utilize data

Components of DBMS

⦿ Procedures

- Refers to the general instructions to use a DBMS which includes setup and installation, to login and logout, to manage databases, to take backups, generating reports, etc...

Components of DBMS

⦿ Database Access Language

- Database Access Language is a simple language designed to write commands to access, insert, update and delete data stored in any database.

Components of DBMS

⦿ Users

- Database Administrators
 - Takes care of security and availability
 - Managing licensed keys
 - Managing user accounts and access, etc...
- Software Developer
 - Designing the parts of DBMS
- End User
 - The one who store, retrieve, update and delete data.

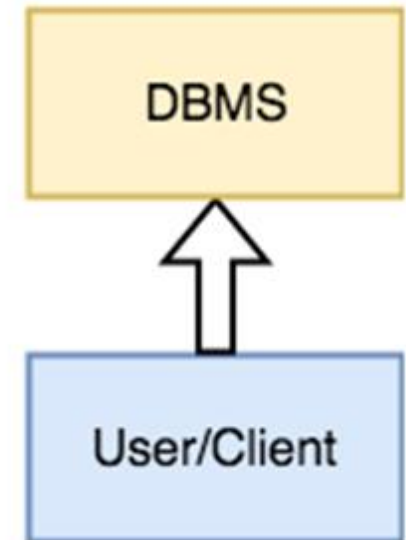
Architecture of DBMS

Architecture of DBMS

- ⦿ A Database Management System is not always directly available for users and applications to access and store data in it
- ⦿ It can be **centralised** (all the data stored at one location)
- ⦿ It can be **decentralised** (multiple copies of database at different locations)
- ⦿ It can be hierarchical, depending upon its architecture.

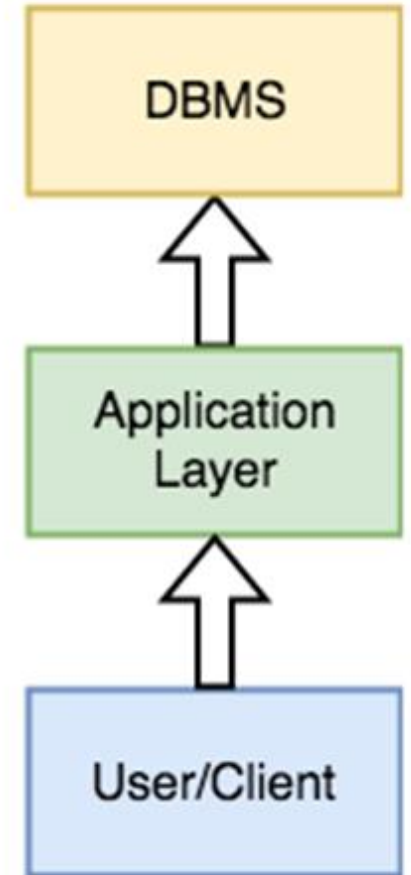
1-tier DBMS Architecture

- ⦿ This is when the database is directly available to the user for using it to store data.



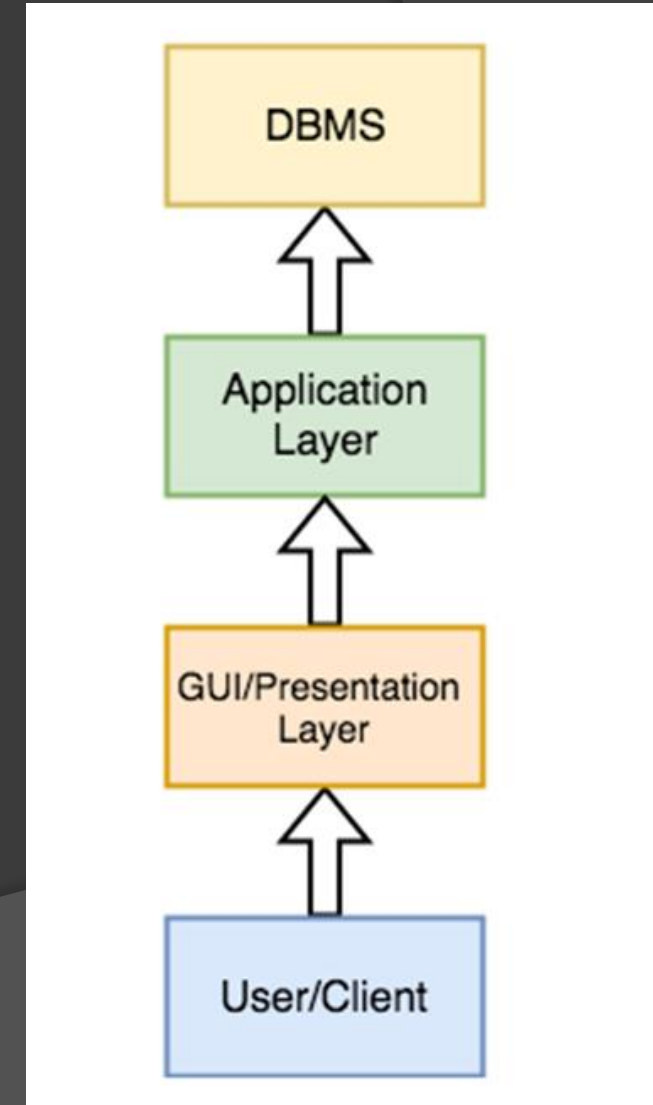
2-tier DBMS Architecture

- Includes an **Application** layer between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.



3-tier DBMS Architecture

- This architecture divides a database system into presentation, application, and data layers, enabling separate handling of user interface, logic processing, and data storage and management
- This separation promotes scalability, modularity, and efficient resource utilization



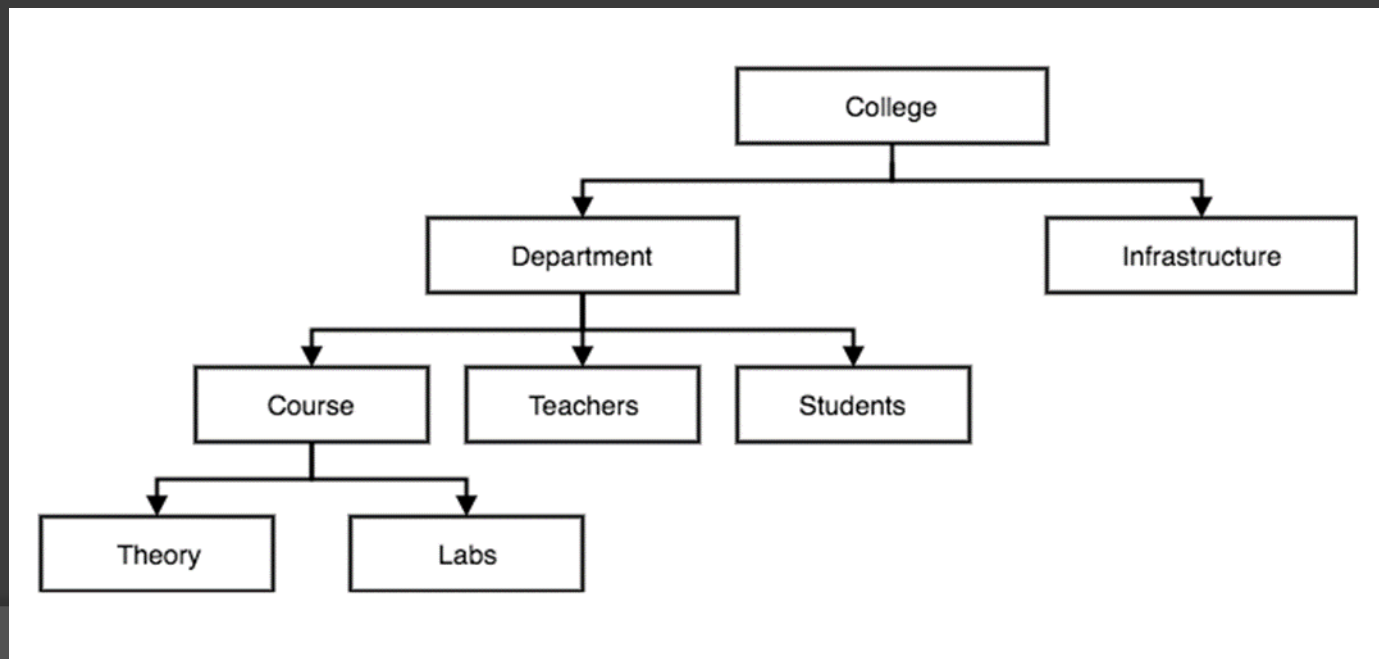
DBMS Database Model

Database Model

- ⦿ A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. **The Relational Model** is the most widely used database model
- ⦿ Other models are:
 - Hierarchical
 - Network
 - Entity-Relationship
 - Relational

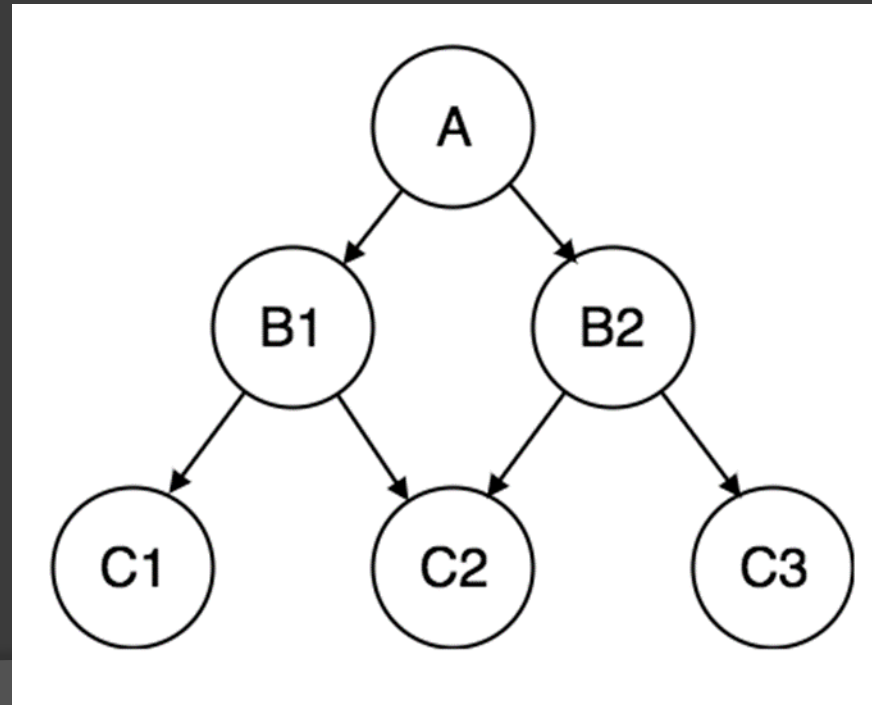
Hierarchical Model

- ⦿ This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked.
- ⦿ The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.



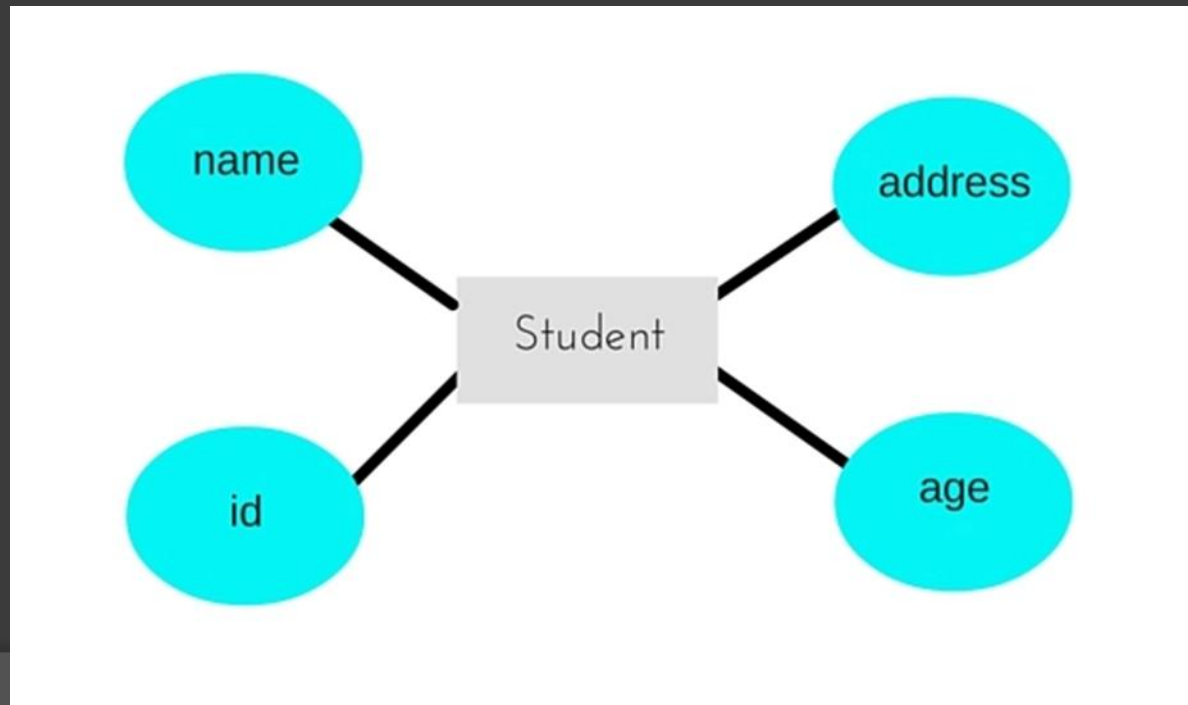
Network Model

- ⦿ This is an extension of the Hierarchical model. In this model data is organized more like a graph, and are allowed to have more than one parent node.



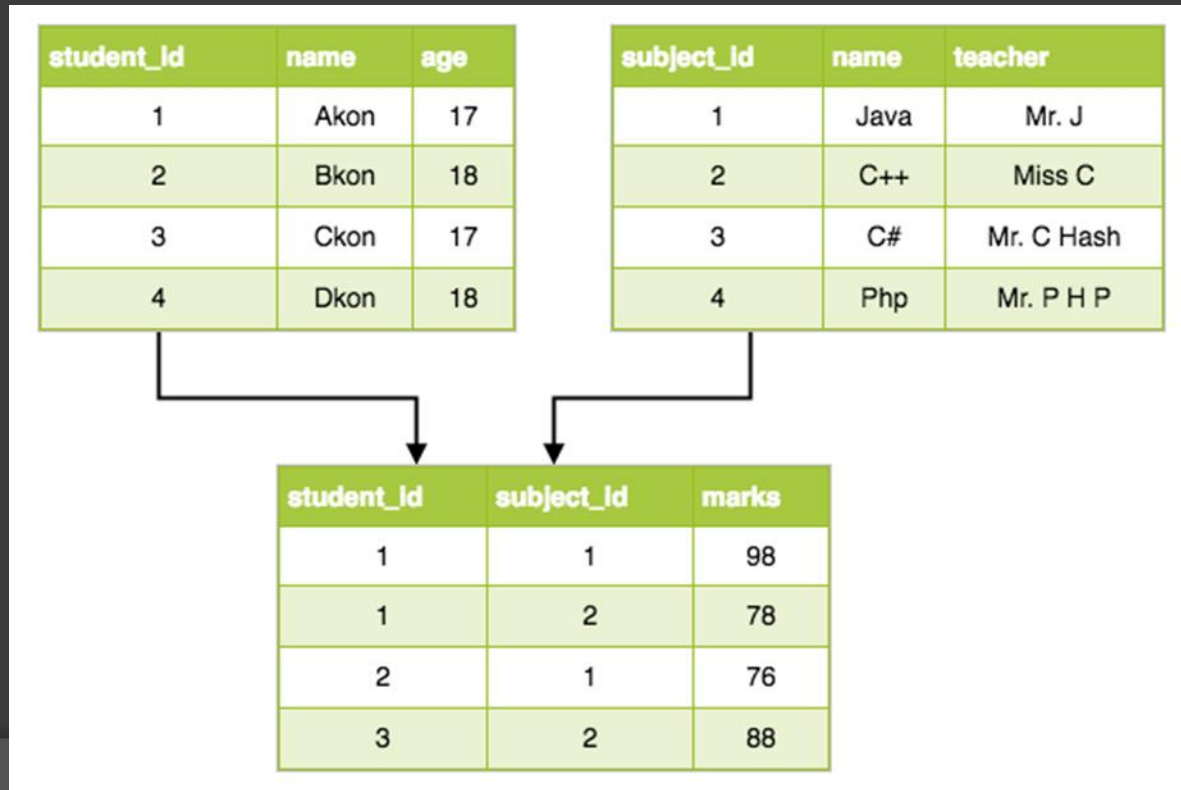
Entity-Relationship Model

- In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes
- Different entities are related using relationships



Relational Model

- In this model, data is organized in two-dimensional tables and the relationship is maintained by storing a common field



Introduction to Relational DBMS

Basic Concepts of RDBMS

- ⦿ Data is stored in relations (tables) and is represented in form of tuples (rows)
- ⦿ It is a collection of organized set of tables related to each other, and from which data can be accessed easily
- ⦿ It is the most commonly used DBMS these days

Fundamentals of RDBMS

- ⦿ Tables
- ⦿ Primary Key
- ⦿ Foreign Key
- ⦿ Relationships
- ⦿ Normalization
- ⦿ Structured Query Language (SQL)
- ⦿ ACID Properties

RDBMS: Tables

- ⦿ Data is organized into tables, which are composed of rows and columns
- ⦿ Each table represents an entity or a relation in the real-world domain
- ⦿ Tables have a defined structure with columns representing attributes or fields, and rows representing individual records or instances

RDBMS: Table Rows

- Also known as records or tuples, rows represent individual instances or entries within a table
- Each row contains a collection of related data that corresponds to a specific entity or object
- For example, in a table for "Employees," each row would represent a unique employee record, containing information such as employee ID, name, department, and salary

RDBMS: Table Columns

- ⦿ Columns, also referred to as fields or attributes, define the specific data elements or properties associated with each row in a table
- ⦿ Each column has a name and a data type that determines the kind of data it can store
- ⦿ Example, in the "Employees" table, columns may include attributes such as "EmployeeID" (numeric data type), "Name" (text data type), "Department" (text data type), and "Salary" (numeric data type)

RDBMS: Primary Key

- ⦿ A primary key is a unique identifier for each row in a table
- ⦿ It ensures the uniqueness and integrity of data within the table
- ⦿ It can be a single column or a combination of columns that uniquely identifies each record

RDBMS: Foreign Key

- ⦿ A foreign key establishes a relationship between two tables
- ⦿ It is a column or set of columns in one table that refers to the primary key in another table
- ⦿ Foreign keys help maintain referential integrity and enforce relationships between tables

RDBMS: Relationships

- Relationships define how tables are related to each other
- The common types of relationships in RDBMS are one-to-one, one-to-many, and many-to-many
- Relationships are established through primary and foreign keys, allowing data to be connected and linked across multiple tables

RDBMS: Relation Schema

- A relation schema describes the structure of the relation, with the name of the relation (name of table), its attributes and their names and type

RDBMS: Relation Key

- A relation key is an attribute which can uniquely identify a particular tuple (row) in a relation (table)

RDBMS: Normalization

- ⦿ Normalization is the process of organizing data in a database to eliminate redundancy and improve data integrity
- ⦿ It involves dividing large tables into smaller, related tables and applying normalization rules to ensure efficient data storage and minimize data anomalies

RDBMS: Structured Query Language

- SQL is the standard language used to interact with RDBMS
- It provides a set of commands for creating, querying, modifying, and managing relational databases
- SQL allows users to perform operations like inserting, updating, deleting, and retrieving data from tables

RDBMS: ACID Properties

- ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability) properties ensure reliable and consistent transaction processing in an RDBMS
- **Atomicity** guarantees that a transaction is treated as a single unit of work
- **Consistency** ensures that a transaction brings the database from one consistent state to another
- **Isolation** ensures that concurrent transactions do not interfere with each other
- **Durability** guarantees that once a transaction is committed, its changes are permanently saved and not lost due to failures

Relational Integrity Constraints

Relational Integrity Constraints

- ⦿ Key Constraints
- ⦿ Domain Constraints
- ⦿ Referential integrity Constraints

Key Constraints

- ⦿ Specifies that there should be such an attribute (column) in a relation (table), which can be used to fetch data for any tuple (row)
- ⦿ The Key attribute should never be **NULL** or same for two different row of data

Domain Constraints

- ⦿ Refers to the rules defined for the values that can be stored for a certain attribute
- ⦿ Example:
 - We cannot store Address of employee in the column for Name
 - A mobile number cannot exceed 11 digits

Referential integrity Constraints

- Referential integrity constraints are rules in a database that ensure the consistency and accuracy of relationships between tables by enforcing valid primary key and foreign key relationships
- They prevent the creation of orphan records and maintain data integrity during insertions, updates, and deletions

Database Keys

Database Keys

- Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table
- A key can be a single attribute or a group of attributes, where the combination may act as a key

Why need a Key ?

- To easily identify any row of data in a table

student_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Candidate Key

- ⦿ Defined as the minimal set of fields which can uniquely identify each record in a table
- ⦿ It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table
- ⦿ A candidate key can never be NULL or empty. And its value should be unique
- ⦿ A candidate key can be a combination of more than one columns(attributes)
- ⦿ There can be more than one candidate keys for a table

Primary Key

- Primary key is a candidate key that is most appropriate to become the main key for any table.
- It is a key that can uniquely identify each record in a table

Primary Key for this table



student_id	name	age	phone


Super Key

- ⦿ Defined as a set of attributes within a table that can uniquely identify each record within a table
- ⦿ It is a superset of Candidate key
- ⦿ Example:
 - (student_id, name)

Composite Key

- Key that consists of two or more attributes that uniquely identify any record in a table is called **Composite Key**. But the attributes which together form the **Composite key** are not a key independently or individually

Composite Key



student_id	subject_id	marks	exam_name

Secondary or Alternative Key

The candidate key which are not selected as primary key are known as secondary keys or alternative keys.

Non-Key Attributes

Attributes or fields of a table, other than candidate key attributes/fields in a table.

Non-prime Attributes

Attributes other than Primary Key attribute(s).

Introduction to SQL

What is SQL ?

- SQL stands for **Structure Query Language**
- It is a database query language used for storing and managing data in Relational DBMS
- Almost all RDBMS (*MySQL, Oracle, Sybase, MS Access, MS SQL Server, etc...*) use SQL as the standard database query language
- SQL is used to perform all types of data operations in RDBMS

SQL Command

- It defines following ways to manipulate data stored in RDBMS

DDL: Data Definition Language

- ⦿ This includes changes to the structure of the table like creation of table, altering table, deleting a table, etc...
- ⦿ All DDL commands are auto-committed. That means it saves all the changes permanently in the database

Command	Description
create	to create new table or database
alter	for alteration
truncate	delete data from table
drop	to drop a table
rename	to rename a table

DML: Data Manipulation Language

- ⦿ DML commands are used for manipulating the data stored in the table and not the table itself.
- ⦿ DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Command	Description
insert	to insert a new row
update	to update existing row
delete	to delete a row
merge	merging two rows or two tables

TCL: Transaction Control Language

- ⦿ These commands are to keep a check on other commands and their affect on the database.
- ⦿ These commands can annul changes made by other commands by rolling the data back to its original state.
- ⦿ It can also make any temporary change permanent.

Command	Description
commit	to permanently save
rollback	to undo change
savepoint	to save temporarily

DCL: Data Control Language

- Data control language are the commands to grant and take back authority from any database user

Command	Description
grant	grant permission of right
revoke	take back permission.

DQL: Data Query Language

- Data query language is used to fetch data from tables based on conditions that we can easily apply

Command	Description
<code>select</code>	retrieve records from one or more table

Database Design

Database Design

- ⦿ Database design refers to the process of creating a well-structured and efficient database system that meets the requirements of an organization or application
- ⦿ It involves identifying data entities, defining relationships between them, determining attributes and data types, and establishing data integrity constraints
- ⦿ The goal of database design is to optimize data storage, retrieval, and manipulation while ensuring data consistency and usability

Database Design

Key Steps

1. Requirement Analysis

- ④ Understand the requirements of the organization or application that will use the database.
- ④ Identify the key entities (objects), their attributes (properties), and the relationships between them.

2. Conceptual Design

- ④ Create a high-level conceptual model that represents the entities, attributes, and relationships using techniques like entity-relationship (ER) diagrams.
- ④ This step focuses on the overall structure and logical organization of the database.

3. Logical Design

- ④ Translate the conceptual model into a logical design by mapping entities and relationships into tables and defining the attributes and their data types.
- ④ Normalize the tables to eliminate redundancy and ensure data integrity.

4. Physical Design

- ⦿ Decide on the physical implementation details of the database, such as selecting the database management system (DBMS), defining table structures, indexing, partitioning, and storage considerations.
- ⦿ This step takes into account performance, scalability, and specific features of the chosen DBMS.

5. Schema Refinement

- ④ Refine the database schema by adding constraints, such as primary key, foreign key, and other data validation rules.
- ④ Define integrity constraints to maintain data consistency and ensure data quality.

6. Database normalization

- Apply normalization techniques to eliminate data redundancy, improve efficiency, and reduce anomalies in data storage.

7. Indexing and Performance Optimization

- ④ Identify the key access patterns and optimize database performance by creating indexes on frequently queried columns.
- ④ Consider strategies like query optimization, caching, and partitioning to enhance performance.

8. Security and Access Control

- ⦿ Define security measures, including authentication, authorization, and access control mechanisms, to ensure that only authorized users can access and modify the data.

9. Data Migration and Integration

- ④ Plan and execute the migration of existing data into the new database system.
- ④ Integrate data from different sources and ensure data consistency during the migration process.

10. Testing and Evaluation

- ⦿ Thoroughly test the database design and functionality to ensure it meets the specified requirements.
- ⦿ Perform data validation, perform stress testing, and verify the accuracy and reliability of the system.

11. Documentation

- ⦿ Document the database design, including the schema, relationships, constraints, and any other relevant information.
- ⦿ This documentation helps in understanding and maintaining the database system in the future.

Identifying The Tables and Relationships

Identifying Tables and Relationships

- To identify tables and relationships in database design, first, determine the key entities and their attributes.
- Analyze the relationships between entities and define their cardinality (e.g., one-to-one, one-to-many).
- Assign primary keys for each table and foreign keys to establish relationships, ensuring data integrity and efficient data organization.

How to Approach

Identify Entities

- Begin by identifying the main entities or objects relevant to the application or organization.
- These entities represent the key components or concepts that you want to store data about.
- For example, in a school management system, entities could include students, teachers, courses, and classrooms.

Determine Attributes

- For each entity, determine the attributes or properties that describe it.
- Attributes are the specific data elements or characteristics associated with an entity.
- For example, for the "students" entity, attributes could include student ID, name, date of birth, and contact information.

Define Relationships

- ⦿ Analyze the relationships between entities.
- ⦿ Relationships represent how entities are connected or associated with each other.
- ⦿ There are different types of relationships, such as one-to-one, one-to-many, and many-to-many.
- ⦿ Determine which entities have relationships and how they are related.
- ⦿ For example, a student entity may have a one-to-many relationship with a course entity, as a student can be enrolled in multiple courses.

Determine Cardinality

- Cardinality defines the number of instances or occurrences of one entity that can be associated with another entity in a relationship.
- Determine the cardinality of each relationship, such as one-to-one, one-to-many, or many-to-many.
- For example, in the student-course relationship, a student can be associated with multiple courses, indicating a one-to-many cardinality.

Assign Primary Keys

- ④ Identify the primary key for each table.
- ④ A primary key is a unique identifier for each row in a table.
- ④ It ensures that each record is uniquely identifiable.
- ④ Usually, an attribute with a unique value, such as student ID or course ID, is chosen as the primary key.

Assign Foreign Keys

- ⦿ Determine the foreign keys that establish relationships between tables.
- ⦿ A foreign key is a column in one table that references the primary key of another table.
- ⦿ It enables the establishment of relationships and ensures data integrity.
- ⦿ For example, in the student-course relationship, the student ID in the course table would be a foreign key referencing the primary key in the student table.

By identifying the tables and their relationships, you establish the foundation for designing the database schema.

The step helps determining the structure, attributes, and connectivity between tables, enabling efficient data organization and retrieval in the database.

Commonly Used Data Types

String Datatypes

The following are the **String Datatypes** in MySQL:

Data Type Syntax	Maximum Size	Explanation
CHAR(<i>size</i>)	Maximum size of 255 characters.	Where size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARCHAR(<i>size</i>)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.
TINYTEXT(<i>size</i>)	Maximum size of 255 characters.	Where size is the number of characters to store.
TEXT(<i>size</i>)	Maximum size of 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(<i>size</i>)	Maximum size of 16,777,215 characters.	Where size is the number of characters to store.
LONGTEXT(<i>size</i>)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characters to store.
BINARY(<i>size</i>)	Maximum size of 255 characters.	Where size is the number of binary characters to store. Fixed-length strings. Space padded on right to equal size characters. (Introduced in MySQL 4.1.2)
VARBINARY(<i>size</i>)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string. (Introduced in MySQL 4.1.2)

Numeric Datatypes

The following are the **Numeric Datatypes** in MySQL:

Data Type Syntax	Maximum Size	Explanation
BIT	Very small integer value that is equivalent to TINYINT(1). Signed values range from -128 to 127. Unsigned values range from 0 to 255.	
TINYINT(<i>m</i>)	Very small integer value. Signed values range from -128 to 127. Unsigned values range from 0 to 255.	
SMALLINT(<i>m</i>)	Small integer value. Signed values range from -32768 to 32767. Unsigned values range from 0 to 65535.	
MEDIUMINT(<i>m</i>)	Medium integer value. Signed values range from -8388608 to 8388607. Unsigned values range from 0 to 16777215.	
INT(<i>m</i>)	Standard integer value. Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295.	

INTEGER(<i>m</i>)	Standard integer value. Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295.	This is a synonym for the INT datatype.
BIGINT(<i>m</i>)	Big integer value. Signed values range from -9223372036854775808 to 9223372036854775807. Unsigned values range from 0 to 18446744073709551615.	
DECIMAL(<i>m</i> , <i>d</i>)	Unpacked fixed point number. <i>m</i> defaults to 10, if not specified. <i>d</i> defaults to 0, if not specified.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal.
DEC(<i>m</i> , <i>d</i>)	Unpacked fixed point number. <i>m</i> defaults to 10, if not specified. <i>d</i> defaults to 0, if not specified.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal. This is a synonym for the DECIMAL datatype.
NUMERIC(<i>m</i> , <i>d</i>)	Unpacked fixed-point number. <i>m</i> defaults to 10, if not specified. <i>d</i> defaults to 0, if not specified.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal. This is a synonym for the DECIMAL datatype.

FIXED(<i>m</i> , <i>d</i>)	Unpacked fixed-point number. <i>m</i> defaults to 10, if not specified. <i>d</i> defaults to 0, if not specified.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal. (Introduced in MySQL 4.1) This is a synonym for the DECIMAL datatype.
FLOAT(<i>m</i> , <i>d</i>)	Single precision floating point number.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal.
DOUBLE(<i>m</i> , <i>d</i>)	Double precision floating point number.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal.
DOUBLE PRECISION(<i>m</i> , <i>d</i>)	Double precision floating point number.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal. This is a synonym for the DOUBLE datatype.
REAL(<i>m</i> , <i>d</i>)	Double precision floating point number.	Where <i>m</i> is the total digits and <i>d</i> is the number of digits after the decimal. This is a synonym for the DOUBLE datatype.

FLOAT(<i>p</i>)	Floating point number.	Where <i>p</i> is the precision.
BOOL	Synonym for TINYINT(1)	Treated as a boolean data type where a value of 0 is considered to be FALSE and any other value is considered to be TRUE.
BOOLEAN	Synonym for TINYINT(1)	Treated as a boolean data type where a value of 0 is considered to be FALSE and any other value is considered to be TRUE.

Date/Time Datatypes

The following are the **Date/Time Datatypes** in MySQL:

Data Type Syntax	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'YYYY-MM-DD'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIMESTAMP(<i>m</i>)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2 4)]	Year value as 2 digits or 4 digits.	Default is 4 digits.

Large Object (LOB) Datatypes

The following are the **LOB Datatypes** in MySQL:

Data Type Syntax	Maximum Size	Explanation
TINYBLOB	Maximum size of 255 bytes.	
BLOB(<i>size</i>)	Maximum size of 65,535 bytes.	Where size is the number of characters to store (size is optional and was introduced in MySQL 4.1)
MEDIUMBLOB	Maximum size of 16,777,215 bytes.	
LONGTEXT	Maximum size of 4GB or 4,294,967,295 characters.	

What can SQL do ?

- ⦿ SQL can execute queries against a database
- ⦿ SQL can retrieve data from a database
- ⦿ SQL can insert records in a database
- ⦿ SQL can update records in a database
- ⦿ SQL can delete records from a database
- ⦿ SQL can create new databases
- ⦿ SQL can create new tables in a database
- ⦿ SQL can create stored procedures in a database
- ⦿ SQL can create views in a database
- ⦿ SQL can set permissions on tables, procedures, and views

SQL is a Standard - BUT....

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the major commands (such as **SELECT, UPDATE, DELETE, INSERT, WHERE, CREATE, DROP**) in a similar manner.

Some of The Most Important SQL Commands

- ◉ **SELECT** - extracts data from a database
- ◉ **UPDATE** - updates data in a database
- ◉ **DELETE** - deletes data from a database
- ◉ **INSERT INTO** - inserts new data into a database
- ◉ **CREATE DATABASE** - creates a new database
- ◉ **ALTER DATABASE** - modifies a database
- ◉ **CREATE TABLE** - creates a new table
- ◉ **ALTER TABLE** - modifies a table
- ◉ **DROP TABLE** - deletes a table
- ◉ **CREATE INDEX** - creates an index (search key)
- ◉ **DROP INDEX** - deletes an index