INTRODUCTION TO DATABASE SYSTEMS

DATA

- Raw, unprocessed facts, figures, or symbols without context or meaning

CHARACTERISTICS

- unorganized
- can be numbers, text, images, or sounds (input devices are needed to record data)
- No interpretation

INFORMATION

- Processed or organized data
- Has structure
- Has meaning,relevance, and context

CHARACTERISTICS

- Arranged or structured
- Answers, "who, what, where, and when"
- Useful for decision making of a basic level

KNOWLEDGE

- Information that has been understood, internalized, and applied based on experience, rules, or insight

CHARACTERISTICS

- Answers, "how and why"
- Enables prediction and decision making
- Often tied to human expertise

DATABASE

- Structured collection of data stored electronically
- Manage by Database Management System (DBMS)

IMPORTANCE IN ORGANIZATION

- Efficiency - has an element of time (faster locating of data)
- Reliability - no redundancy, reliable, accurate, consistency
- Scalability - storage grows as the business grows

EXAMPLE:

- Banking system
- Inventory management system (needs to measure critical level)
- Hospital records
- School systems

Social media platform

- Uses XML structure
- Posts are unstructured form of database

DATABASE MODELS

DATABASE MODEL

- Determines how data is stored, organized, manipulated
- Structured
- Data model is a framework

WHY DOES IT MATTERS?

- Organize complex data
- Improve efficiency and reliability
- Provide structure for queries and updates
- Accurate and consistent records

TYPES OF DATABASE MODELS

- Choose depends on what you need

HIERARCHICAL MODEL

- parent-child relationship
- Can adapt organizational structure
- Tree structure
- One to many relationships

PROS: simple, efficient
CONS: rigid, hard to update

NETWORK MODEL

- Support many to many
- Multiple records
- Uses graph structure

PROS: handles complex relationships
CONS: hard to maintain

RELATIONAL MODEL

- Data in tables with rows and columns
- Relationships defined by common values in columns
- Scalable

PROS: easy to use, SQL support, scalable
CONS: performance issues at very large scale

OBJECT-ORIENTED MODEL

- Stores data in objects
- Includes data and the method that operates on that data

PROS: great for multimedia and complex applications
CONS: less widespread compared to relational

DATABASE MANAGEMENT SYSTEM (DBMS) AND ITS FUNCTION

DATABASE MANAGEMENT SYSTEM

- Software that allows users to create, access, manage, and control databases efficiently.
- Acts as interface for users

DBMA

- grant and revoke access, authorized

- Database Management Administrator

DBMS FUNCTION

DATA DEFINITION

- Allows defining the database structure

DATA STORAGE, RETRIEVAL, AND UPDATE

- Stores large volumes of data systematically
- Provides efficient retrieval using queries
- Updates records without affecting other data

DATA SECURITY AND INTEGRITY

- Ensures only authorized users can access or modify data
- Maintains accuracy and consistency

DATA ADMINISTRATION

- Provides tools for monitoring, backup, and recovery

MULTI-USER ACCESS & CONCURRENCY CONTROL

- Supports multiple users working simultaneously
- Prevents conflicts when two users try to update the same record

DATA INDEPENDENCE

- Logical design is separate from physical storage
- Users don't need to know or where data is stored

BACKUP AND RECOVERY

- Ensures data is recoverable after system failures

REAL-WORLD EXAMPLES

- Banking systems
- E-commerce
- University enrollment

RELATIONAL MODEL

RELATIONAL MODEL

- Represents how data is stored and managed in Relational Database
- Data is organized into tables.
- Introduced by Edward F. Codd of IBM in 1970
- Based on mathematical set theory.
- Each table consists of rows (tuples) and columns (attributes).
- Relationships between tables are well-defined.

RELATION (TABLE)

- A two-dimensional table used to store a collection of data.

TUPLE (ROW)

- A single row in a table that represents one complete record.

ATTRIBUTE (COLUMN)

- A column in a table that contains values for a specific data point.

RELATION SCHEMA

- The metadata or blueprint that defines the structure of table
- (Customer, Customer_ID, Name, Contact_Number, Address)

DOMAIN

- The set of all possible, valid, and atomic (indivisible) values for an attribute.

DEGREE

- The number of attributes (columns) in a relation.

CARDINALITY

- The number of tuples(rows) in a relation.

PRIMARY KEY (PK)

- A column or set of columns that uniquely identifies each record (row) in a table.

FOREIGN KEY (FK)

- A column or set of columns in one table that links to the primary key of another table, establishing a relationship.

RELATIONAL INTEGRITY CONSTRAINTS

These are rules that maintain the accuracy, consistency, and reliability of data.

1. **Domain Constraints:** Restrict the values an attribute can take to its defined domain. (e.g., employee_age must be an INTEGER between 18 and 65).
2. **Key Constraints:** Dictate that the Primary Key cannot have a null value. This ensures every row has a unique identifier.
3. **Referential Integrity Constraints:** Ensure that a Foreign key value must have a matching Primary key value in the table it references (or be NULL). THis prevents orphaned records.

OPERATIONS IN THE RELATIONAL MODEL

DATA MANIPULATION LANGUAGE (DML)

Used for managing existing data within tables.

- **INSERT:** adds a new tuple (row)
- **UPDATE:** Modifies attributes values in existing tuples.
- **DELETE:** removes one or more tuples.
- **SELECT:** Retrieves a specific range of data based on criteria.

RELATIONAL ALGEBRA

A procedural query language based on mathematical operations.

- **Selection:** Filters rows from a table based on a condition.
- **Projection:** Selects specific columns from a table.
- **Union:** Combines the tuples of two relation (with the same attributes), removing duplicates
- **Set Difference:** returns tuples from the first relation that are not in the second.
- **Cartesian Product:** Combines every tuple of the first relation with every tuple of the second.
- **Join:** Combines related tuples from multiple relation based on a common attribute.

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Simple: Easy to use and understand.
- **Flexible:** More flexible than other models.
- **Secure:** Offers good security mechanisms.
- **Data Accrucacy**: Clear structure leads to high accuracy.
- **Data Integrity:** Maintained via constraints.

DISADVANTAGES

- **Performance:** Can have issues with very large databases.
- **Complexity for Complex Data:** Struggles with hierarchical or graph-like data.
- **Normalization Overhead:** Extensive normalization (splitting tables) can lead to complex queries.

BEST PRACTICES FOR DESIGN

1. **Understand Business Requirements:** Clarify what data is needed and how it will be used.
2. **Follow Normalization Principles:** Minimize data redundancy (aim for at least 3NF).

3. **Define Clear Naming Conventions:** Use consistent, descriptive names for tables and columns.
4. **Use Appropriate Data Types:** Select the correct type to ensure accuracy and optimize storage.
5. **Utilize Constraints Effectively:** USE PKs, FKs, and other constraints to enforce data integrity.
6. **Optimize with Indexing:** add indexes to frequently queried columns to speed up retrieval.
7. **Document the Design:** keep clear documentation of the schema, relationships, and rules.
8. **Plan for Scalability:** Consider future growth.

ENTITY-RELATIONSHIP MODELING

ENTITY-RELATIONSHIP MODEL

- A diagram that shows how data is related in the database. It uses attributes, and relationships.
- Serves as a blueprint before creating the database.
- Helps plan and organize the database structure.
- shows how data is connected.
- Prevents errors and reduces data redundancy.

ENTITIES

ENTITY

- A real-world object or concept that can be clearly identified and stored (e.g., a specific "Student", "Product", or "Place").

ENTITY TYPE

- A collection of entities that share the same properties or attributes (e.g., the general concept of a Student or Department).

ENTITY SET

- The collection of all entities of a particular type stored in the database (e.g., all the student records in the Student table).

KINDS OF ENTITIES

Independent (Strong) Entity:

- The "backbone" of the database; it can exist on its own.
- Has a primary key that is not a foreign key.

Dependent (Weak) Entity:

- Cannot exist without being associated with a strong entity.

- Often, its primary key will include a foreign key from the strong entity it depends on.
- **Quick Test:** Ask, "Can I add a row to this table without needing data from another table first?"
  - **Yes:** It's an Independent (Strong) entity.
  - **No:** It's a Dependent (Weak) entity.

ATTRIBUTES

- A property or characteristic of an entity (e.g., Name, Age, Price). This corresponds to a column in a table.

TYPES OF ATTRIBUTES

Type	**Description**	**ERD Symbol**	**Example**
Simple	An "atomic" value that cannot be broken down.	Single Oval	Age, Gender
Composite	Can be broken down into smaller, simpler attributes.	Oval with branching ovals	Address (can be broken into Street, City, Region)
Derived	A value that is **calculated** from other attributes. It is not stored directly.	Dashed Oval	Age (calculated from Birthdate)
Multivalued	Can hold more than one value for a single entity.	Double Oval	PhoneNumber, Email (a student might have multiple)
Key	An attribute that uniquely identifies	Oval with underline	StudentID

	each entity instance.	d text	
Stored	An attribute whose value is stored directly in the database.	Single Oval	Birthdate (used to calculate Age)
Null	An attribute that may not have a value (it is optional or not applicable).	(No special symbol)	MiddleName, Nickname

NULL VALUE

- **NULL** represents a **missing, unknown, or inapplicable** value.
- It is **NOT** the same as:
  - 0 (Zero)
  - "" (An empty string)
  - false (Falsy value)
- It simply means "we don't know" or "it doesn't apply."

KEYS

Keys are fields used to identify records and link table together.

- **Primary Key (PK):** The main unique identifier for a record in a table.
  - Must be **UNIQUE**.
  - **CANNOT** be NULL.
- **Candidate Key:** Any attribute (or set of attributes) that *could* be used as the Primary Key.
  - Must be **UNIQUE**.
  - **CANNOT** be NULL.
- **Alternate Key:** A Candidate Key that was **not** chosen to be the Primary Key.
- **Foreign Key (FK):** A column in one table that references the **Primary Key** of another table, creating a link.
  - **CAN** be NULL.
- **Composite Key:** A key that is made up of **two or more columns** combined to create a unique identifier.
- **Super Key:** *Any* set of attributes that can uniquely identify a record (it is not necessarily minimal; e.g., (StudentID, Name) is a Super Key, but (StudentID) is the minimal Candidate/Primary Key).

- **Unique Key:** A constraint that ensures all values in a column are unique.
    - It is like a Primary Key, but it **CAN** accept NULL values (usually just one).

RELATIONSHIPS

Relationships are the "glue" that connects entities.

- **Cardinality:** The **maximum** number of entities that can be involved in the relationship.
- **Modality (or Participation):** The **minimum** number of entities required (i.e., is the relationship mandatory or optional?).

TYPES OF RELATIONSHIPS

- **One-to-One (1:1):** One instance of Entity A is linked to exactly one instance of Entity B.
    - *Example*: Student is issued one StudentNumber.
- **One-to-Many (1:N):** One instance of Entity A can be linked to many instances of Entity B.
    - *Example*: One Customer can place many Orders.
- **Many-to-Many (M:N):** Many instances of Entity A can be linked to many instances of Entity B. This requires a **Junction Table** (or "associative entity") to work.
    - *Example*: Many Students can enroll in many Projects. The Enrollment table links them.
- **Recursive (Unary) Relationship:** An entity has a relationship with itself.
    - *Example*: An Employee (who is a manager) Supervises other Employees.
- **Ternary (N-ary) Relationship:** A single relationship that involves **three or more** entities.
    - *Example*: A Doctor prescribes a Medication to a Patient. All three are linked by the Prescription entity.

ER DIAGRAMS (ERD)

- An ERD is the **graphical visualization** of the ER Model.
- **Conceptual ERD (Chen Notation):**
    - Entities: Rectangles
    - Relationships: Diamonds

    - Attributes: Ovals
- **Physical ERD (MySQL Workbench Notation):**
    - **Attributes:**
        - Solid Diamond: NOT NULL
        - Unfilled Diamond: NULL (optional)
    - **Keys:**
        - Blue: Simple Key
        - Red: Foreign Key

SQL BASICS

SQL

- Structured Query Language
- A standard language used to manage and manipulate relational databases.
- Allows users to store, retrieve, update, and delete data.

SQL COMMAND CATEGORIES

- **DDL (Data Definition Language):** Deals with the *structure* of the database.
- **DML (Data Manipulation Language):** Deals with the *data inside* the database.

DATA DEFINITION LANGUAGE (DDL)

DDL commands are used to **define, modify, and delete the structure** of database objects like tables, views, and indexes.

DDL COMMANDS

- **CREATE**
    - **What it does:** Creates new database objects.
    - **Examples:** CREATE DATABASE, CREATE TABLE, CREATE VIEW, CREATE INDEX.
- **ALTER**
    - **What it does:** Modifies the structure of an existing table.
    - **Common Uses:**
        - ADD: Adds a new column.
        - DROP: Removes a column.
        - MODIFY: Changes a column's data type or size.
        - RENAME COLUMN: Renames an existing column.
- **RENAME**
    - **What it does:** Changes the name of an existing table.
    - **Example:** RENAME TABLE Customers to VIP;

- **COMMENT**
  - **What it does:** Adds notes to the SQL code that are not executed.
  - **Types:**
    - Single-Line: -- This is a comment
    - Multi-Line: /* This is a multi-line comment */
- **DROP**
  - **What it does:** **Permanently removes** an entire database object (like a table), including its structure and all its data.
  - **Foreign Key (FK) Issue:** You cannot DROP a table if another table's Foreign Key references it. To fix this, you must first drop the "child" table (the one with the FK) or ALTER the child table to remove the FK constraint.
- **TRUNCATE**
  - **What it does:** Removes **all rows (data)** from a table but leaves the table's structure intact.
  - It is much faster than DELETE for clearing an entire table.

DATA MANIPULATION LANGUAGE (DML)

DML commands are used to **manage and manipulate the actual data** stored within the database tables.

DML COMMANDS

- **SELECT**
  - **What it does:** Retrieves data from one or more tables.
  - **Examples:**
    - SELECT * FROM Customers; (Retrieves all columns)
    - SELECT customer_name, address FROM Customers; (Retrieves specific columns)
- **INSERT**
  - **What it does:** Adds new rows (records) to a table.
  - **Example:** INSERT INTO Customers (customer_name, phone) VALUES ('Jose Rizal', '09918203345');
- **UPDATE**
  - **What it does:** Modifies existing records in a table.

    - **Example:** UPDATE Suppliers SET email = 'new_email@gmail.com' WHERE supplier_name = 'TheHague Ph';
    - **Note:** The presentation mentions **Error 1175 (Safe Update Mode)**, which prevents UPDATE or DELETE commands without a WHERE clause that uses a key. This can be disabled with SET SQL_SAFE_UPDATES = 0;.
- **DELETE**
    - **What it does:** Removes specific rows from a table.
    - **Example:** DELETE FROM Suppliers WHERE supplier_name = 'DavaoTech Ph';

DIFFERENCES DDL VS. DML

Feature	DDL (Data Definition Language)	DML (Data Manipulation Language)
Purpose	Defines database objects	Manipulates data within objects
What it Affects	The **structure** of the database	The **data** stored in the database
Transactional	No (auto-committed, cannot be rolled back)	Yes (can be rolled back)
When Used	Typically during design and setup	During normal database operation