

Unit 1, Topic 5 – Data and Databases

European Institute of Innovation, Entrepreneurship and Technology
Lessingstraße 79, 13158 Berlin, Germany

Presenter Name| Irikefe Hope Oniovosa



**Lessingstraße 79,
13158 Berlin, Germany**
Phone: +49 30 233618555
Fax: +49 30 233618556
Email: info@eiiet.com

Discover the world of eiIET

Data vs. Information

- **Data:**

- Definition: Raw facts and figures without context.
- Characteristics:
 - Unprocessed, unorganized.
 - May include numbers, text, images, etc.
 - Example: A list of temperatures (e.g., 23, 19, 21, 22).

- **Information:**

- Definition: Processed data that has meaning and value.
- Characteristics:
 - Data that has been organized, interpreted, or structured.
 - Provides context and relevance.
 - Example: "The average temperature for today is 21°C," which gives meaning to the raw data values.

Source and Types of Data

- **Structured Data**

- **Definition:** Data that is organized in a pre-defined manner.

- **Characteristics:**

- Easily searchable and often stored in spreadsheets or relational databases.
- Organized in tables with rows and columns.

- **Examples:**

- A customer database with fields for name, address, and phone number.
- A spreadsheet of sales figures organized by date and product.

- **Unstructured Data**

- **Definition:** Data that is not organized in a pre-defined manner.

- **Characteristics:**

- More challenging to search and analyze.
- Often includes formats like text, images, videos, and social media posts.

- **Examples:**

- Emails and free-form text messages.
- Videos, audio recordings, and social media posts.
- Web pages and blogs with varied content.

Activity: Classifying Data vs. Information

- **Examples for Discussion:**

- **Example 1:**

- Data: "78, 82, 85, 80"
- Information: "The exam scores of the class in Mathematics."

- **Example 2:**

- Data: A collection of social media posts.
- Information: "Analysis of sentiment trends from social media posts about a new product launch."

- **Example 3:**

- Data: A raw log file from a website.
- Information: "A report summarizing peak user activity hours on the website."

Structured Data

- **1. Why Databases?**

- **Efficient Retrieval:**

- Databases allow for rapid searching and filtering of data using structured query languages (e.g., SQL).

- **Data Integrity:**

- Rules and constraints (like primary keys) ensure that the data is accurate and consistent.

- **Scalability:**

- As organizations grow, databases can handle increasing volumes of data without a complete system overhaul.

- **Centralized Management:**

- Databases consolidate data storage, making maintenance, updates, and security management more efficient.

Structure of a Relational Database

- **Tables:**

- **Definition:** Core structures that hold records.
- **Example:** A "Customers" table with each row representing a single customer.

- **Records and Fields:**

- **Records:** Rows in a table, each containing a complete set of information.
- **Fields:** Columns in a table, representing individual attributes (e.g., CustomerID, Name, Email).

- **Primary Keys:**

- **Definition:** A unique identifier for each record in a table.
- **Example:** CustomerID in a "Customers" table.

- **Foreign Keys:**

- **Definition:** Fields that create a link between tables, referencing primary keys in other tables.
- **Example:** The CustomerID in an "Orders" table, linking an order to a specific customer.

- **Composite Keys:**

- **Definition:** When more than one field is used together as a unique identifier for a record.
- **Example:** In a table that records course enrollments, a combination of StudentID and CourseID may serve as a composite key.

Entities and Relationships

- **Entities:**

- Represent real-world objects or concepts (e.g., Customer, Order, Product).

- **Types of Relationships:**

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

- **Definition:** One instance of an entity is related to one instance of another entity.
 - **Example:** Each person has one unique passport.

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

- **Definition:** One instance of an entity relates to multiple instances of another.
 - **Example:** A single customer can place multiple orders.

- **Many-to-Many (M:N):**

- **Definition:** Multiple instances of one entity relate to multiple instances of another.
 - **Example:** In an online retail store, one order may include several products, and one product may appear in many orders.

Entity Relationship Diagrams (ERDs)

- **Purpose:**

- ERDs visually represent entities, attributes, and the relationships among them.

- **Symbols (referencing Appendix 7):**

- **Entities:** Represented as rectangles with the entity name inside.
- **Attributes:** Often listed within the rectangle or as ovals connected to the entity.
- **Relationships:** Lines connecting entities, often using "crow's foot" notation to indicate one-to-many or many-to-many relationships.

- **Example Demonstration:**

- Walk through an example ERD for a retail inventory system:
 - **Entities:** Product, Category, Supplier, Order.
 - **Relationships:**
 - Each product belongs to one category (One-to-Many).
 - Each product is supplied by a supplier (One-to-Many).
 - An order can include multiple products, and each product can appear in multiple orders (Many-to-Many)

1. ERD example involving Products, Categories, and Suppliers:

-  **Diagram Explanation:**
-  **Primary Keys and Entities:**
- **Product Entity:**
 - **Primary Key:** ProductID
 - Other attributes: ProductName, Price, CategoryID, SupplierID
- **Category Entity:**
 - **Primary Key:** CategoryID
 - Other attributes: CategoryName, Description
- **Supplier Entity:**
 - **Primary Key:** SupplierID
 - Other attributes: SupplierName, ContactInfo
- Each entity has a **primary key** that uniquely identifies every record in its table:
- ProductID uniquely identifies each product.
- CategoryID uniquely identifies each category.
- SupplierID uniquely identifies each supplier.

2. ERD example involving Products, Categories, and Suppliers:

- The **Products table** contains:
 - CategoryID → A **foreign key** referencing CategoryID in the **Categories** table.
 - SupplierID → A **foreign key** referencing SupplierID in the **Suppliers** table.
- These foreign keys **establish relationships** between tables:

1. Products → Categories (Many-to-One)

1. Each product **belongs to one category**.
2. But a single category can include **many products**.
3. This is a **One-to-Many relationship** from **Category to Product**.

2. Products → Suppliers (Many-to-One)

1. Each product is **supplied by one supplier**, but a supplier can provide **many products**.
2. This is a **One-to-Many relationship** from **Supplier to Product**

- The **Products table** contains:

- CategoryID → A **foreign key** referencing CategoryID in the **Categories** table.
- SupplierID → A **foreign key** referencing SupplierID in the **Suppliers** table.

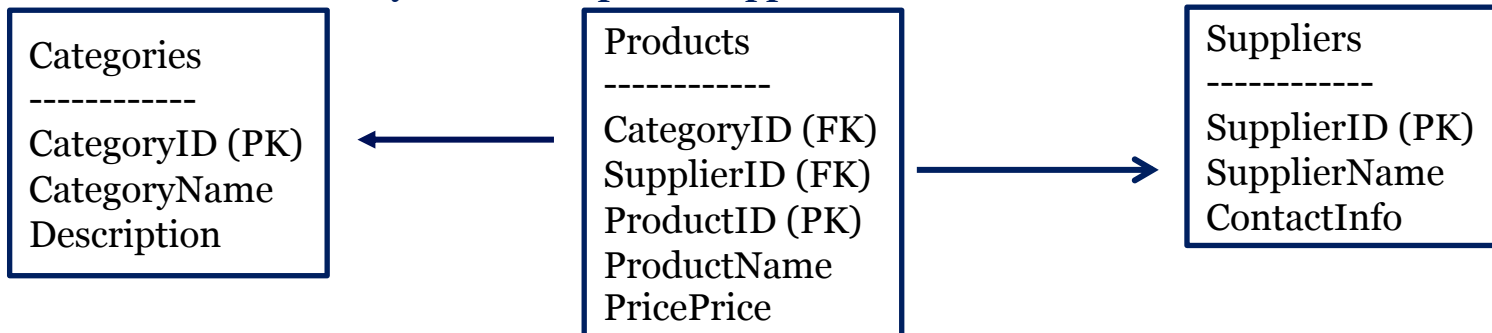
- These foreign keys **establish relationships** between tables:

1. Products → Categories (Many-to-One)

1. Each product **belongs to one category**.
2. But a single category can include **many products**.
3. This is a **One-to-Many relationship** from **Category to Product**.

2. Products → Suppliers (Many-to-One)

1. Each product is **supplied by one supplier**, but a supplier can provide **many products**.
2. This is a **One-to-Many relationship** from **Supplier to Product**



Structured Query Language (SQL)

- **Concept Overview**

- **What is SQL?**

SQL (Structured Query Language) is a domain-specific language used to manage and manipulate relational databases. It allows users to create, read, update, and delete data (commonly known as CRUD operations) using standardized commands.

- **Why SQL?**

SQL is crucial for interacting with databases because it enables:

- Efficient data retrieval.
- Data organization and integrity.
- Complex queries involving multiple tables.
- Automated data manipulation and reporting.

Key SQL Operations

- **1. Querying Data**

- **SELECT Statements:**

Used to retrieve data from one or more tables

```
SELECT * FROM Customers;
```

- **Filtering with WHERE:**

Allows you to narrow down the results based on specific criteria

```
SELECT * FROM Orders WHERE OrderDate > '2023-01-01';
```

- **Subqueries:**

A query nested within another query to perform more advanced filtering.

```
SELECT * FROM Orders  
WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE Country = 'UK');
```

Key SQL Operations

2. Creating and Modifying Tables

- **CREATE TABLE Statements:**

Defines a new table with specified columns and data types.

```
CREATE TABLE Employees (  
    EmployeeID INT PRIMARY KEY,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    HireDate DATE  
);
```

- **Altering Tables:**

Modifies an existing table (e.g., adding new columns).

```
ALTER TABLE Employees ADD Email VARCHAR(100);
```

Key SQL Operations

- **3. Manipulating Data**

- **INSERT:**

Adds new records to a table.

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, HireDate)
VALUES (1, 'John', 'Doe', '2020-05-01');
```

- **UPDATE:**

Modifies existing records.

```
UPDATE Employees
SET LastName = 'Smith'
WHERE EmployeeID = 1;
```

- **DELETE:**

Removes records from a table.

```
DELETE FROM Employees
WHERE EmployeeID = 1;
```

Key SQL Operations

- **4. Linking Tables**

- **JOIN Operations:**

Combine rows from two or more tables based on a related column.

```
SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

- **LEFT JOIN:** Returns all rows from the left table and matched rows from the right.

```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

- **UNION:**

Combines the results of two SELECT statements into a single result.

```
SELECT City FROM Customers  
UNION  
SELECT City FROM Suppliers;
```

Key SQL Operations

- **5. Using Wildcards and Aggregates**

- **Wildcards in LIKE Clauses:**

- **% (Percent):** Represents zero, one, or multiple characters.
- **INNER JOIN:** Returns rows with matching values in both tables.

```
SELECT * FROM Customers WHERE CustomerName LIKE 'A%';
```

- **_ (Underscore):** Represents a single character.

```
SELECT * FROM Customers WHERE CustomerName LIKE '_ohn';
```

- **Aggregate Functions:**

Functions such as COUNT, SUM, AVG, MAX, MIN, often used with: **GROUP BY:** Groups rows sharing a property.

- **ORDER BY:** Sorts the results.

```
SELECT Country, COUNT(*) AS NumberOfCustomers  
FROM Customers  
GROUP BY Country;
```

Hands-On Exercise

- **Hands-On Exercise**

- **Task:** Provide a simple dataset and assign the following exercises:

1.Create Tables:

1. Create a Products table and an Orders table with appropriate columns and relationships.

2.Insert Records:

1. Insert a few sample records into each table.

3.Query Data:

1. Write a SELECT query that uses a JOIN to display product details alongside corresponding orders.

4.Aggregation:

1. Write a query to count the number of orders per product or per customer.

5.Extra Challenge (Optional):

1. Use a WHERE clause and wildcards to filter results (e.g., find all customers whose names start with "J").

- **Goal:**

Ensure students practice writing SQL queries, understand how to structure their queries, and see how SQL manipulates and retrieves data from relational databases.

Summary and Review (Recap)

- **Data vs. Information:**

- **Data:** Raw facts and figures without context (e.g., individual numbers or text entries).
- **Information:** Processed data that has meaning and context (e.g., the average temperature calculated from a series of readings).

- **Importance of Databases:**

- Databases organize data efficiently, ensuring fast retrieval, maintaining data integrity, and supporting scalable operations.
- They are essential for structured data storage, allowing businesses and organizations to manage large volumes of data reliably.

- **Relational Database Design and ERDs:**

- **Tables, Records, and Fields:** The fundamental building blocks of relational databases.
- **Primary and Foreign Keys:**
 - Primary keys uniquely identify records.
 - Foreign keys establish relationships between tables, linking related data.
- **Entity Relationship Diagrams (ERDs):** Visual tools that help design and understand these relationships and database structures.

- **SQL Operations:**

- **Querying:** SELECT statements and filtering data with WHERE clauses.
- **Manipulating Data:** INSERT, UPDATE, DELETE operations for managing records.
- **Linking Tables:** JOIN operations (INNER JOIN, LEFT JOIN, etc.) combine data from multiple tables.
- **Additional Tools:** Using wildcards with LIKE, and aggregate functions with GROUP BY, ORDER BY, and COUNT.

Q&A Session

- **“What distinguishes structured data from unstructured data?”**

Expected Answer: Structured data is organized in a predefined format (e.g., spreadsheets, relational databases) making it easily searchable, while unstructured data lacks a specific format (e.g., emails, videos, social media posts).

- **“Why are primary keys important in a database?”**

Expected Answer: Primary keys provide a unique identifier for each record, ensuring data integrity and enabling accurate linking between tables via foreign keys.

- **“How does a JOIN operation work in SQL?”**

Expected Answer: A JOIN operation combines rows from two or more tables based on a related column (often a foreign key), allowing data from different tables to be viewed together as a single result set.

Additional Resources

- Videos

1.Introduction to Databases

1. [Introduction to Databases – freeCodeCamp](https://youtu.be/HXV3zeQKqGY) “https://youtu.be/HXV3zeQKqGY”

This video provides an overview of database concepts, explaining why databases are crucial for storing and managing data efficiently.

2.SQL Tutorial for Beginners

1. [SQL Tutorial - Full Database Course for Beginners – Programming with Mosh](https://youtu.be/7S_tz1z_5bA) “https://youtu.be/7S_tz1z_5bA”

This beginner-friendly SQL tutorial covers the fundamentals of SQL operations, including creating tables, querying data, and joining tables.

Thank you



Entrepreneurship and Technology

Lessingstraße 79,

13158 Berlin, Germany

Phone: +49 176 55524449

Email: info@eiiet.com