UNIDAD 6: PROGRAMMING, DATABASE, ENTITY RELATIONSHIPS AND DOCUMENTATION

Index

1. PROGRAMMING BASICS

In this section we will study the basic syntax of programming and other concepts like classes and objects. We will also know object-oriented programming.

1.1 Basic Syntax

;	Semicolon
:	Colon
{}	Brace
[]	Brackets
()	Parentheses
/	Slash
"	Double quotes
'	Single quotes
	Whitespace

1. Variables.

Function: Containers for storing data values.

Types:

- String: stores text.
- Integer (int): stores whole numbers.
- Float: stores numbers with decimals.
- Character (char): stores single characters.
- Boolean: stores true or false.

```
String name = "John";
System.out.println(name);
```

Verbs related to variables.

- Declare a variable.
- Assign value to a variable.
- Change the value of a variable

```
int myNum;
myNum = 15;
System.out.println(myNum);
```

2. Operators.

Function: Perform operations on variables and values.

Types:

- Arithmetic: +, -, *, /
- Assignment: =, +=, etc.
- Comparison: ==, !=, >, <, etc.
- Logical: &&, ||, !

```
String name = "John";
System.out.println(name);
```

Verbs related to arithmetic operators.

Addition (4 + 6 = 10)

- Four plus six equals ten.
- Add (number) to (number).

Subtraction (8 – 3 = 5)

- Eight minus three equals five.
- Subtract (number) to (number).

Multiplication (6 x 4 = 24)

- Six multiplied by four equals 24.
- If you multiply six by four, you get twenty four.

Division (15 / 3 = 5)

- Fifteen divided by three equals five.
- If you divide fifteen by three, you get five.

3. If… Else.

Function: Executes a block of code if a condition is true or false.

Conditions:

- a < b: less than
- a <= b: less than or equal to
- a > b: greater than
- a >= b: greater than or equal to
- a == b: equal to
- a != b: not equal to

```
int time = 20;
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
// Outputs "Good evening."
```

Verbs related to conditionals.

- If a condition evaluates to true/false.
- Execute a block of code.
- Jump over the statement.

4. Loops

Function: Loops can execute a block of code as long as a specified condition is reached.

Types:

- While: loops through a block of code as long as the condition is true.
- Do-while: executes the code block first and then checks the condition.
- For: if we know how many times we want to loop through a block of code.
- For-each: to loop through elements in an array.

```java
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

Verbs related to loops.

- Loop through a block of code/array/etc.
- Increase/decrease the variable.

5. Array.

Function: Used to store multiple values in a single variable.

```java
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

Verbs related to arrays.

- Access the elements of an array.
- Add an element to an array.
- Change an array element.
- Loop through an array.

6. Methods.

Function: Block of code which only runs when it is called.
We use them to perform actions and to reuse code. We can use it as many times as we want.

```java
public class MyClass {
  static void myMethod() {
    System.out.println("I just got executed!");
  }

  public static void main(String[] args) {
    myMethod();
  }
}

// Outputs "I just got executed!"
```

Verbs related to methods.

- Pass parameters to a method.
- Call a method.
- Return a value.

1.2 Classes and Objects

1. Class or Object?

What is it?: A class is a template for objects.

An Object is an instance of a class.

```java
public class MyClass {
  int x = 5;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

Verbs related to Classes and Objects.

- Create a class/an object of x class.
- Create an instance of x class.
- Instantiate a class.

```
int myNum;
myNum = 15;
System.out.println(myNum);
```

2. Class attributes.

What is it? Attributes are qualities/properties of a class.

```
public class Person {
  String fname = "John";
  String lname = "Doe";
  int age = 24;

  public static void main(String[] args) {
    Person myObj = new Person();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Age: " + myObj.age);
  }
}
```

Verbs related to attributes.

- Specify attributes.
- Access the attributes.
- Modify/Override the attributes.
- Initialize the attributes.

3. Constructor.

Function: It is a special method to initialize objects.

- It must match the class name.
- It can take parameters.

```java
public class Car {
  int modelYear;
  String modelName;

  public Car(int year, String name) {
    modelYear = year;
    modelName = name;
  }

  public static void main(String[] args) {
    Car myCar = new Car(1969, "Mustang");
    System.out.println(myCar.modelYear + " " + myCar.modelName);
  }
}
```

Verbs related to Constructors.

- Create a constructor.
- Pass a parameter to the constructor.

4. Access Modifiers.

Modifier	Description
Public	The code is accesible by any other class.
Private	The code is only accesible withing the declared class.
Protected	The code is accesible in the same package and subclasses.
Default	The code is only accesible in the same package.

1.3 OOP concepts

1. Encapsulation.

Function: Make sure sensitive data is hidden from users.

Examples:

- Declare class variables/attributes as private.
- Getters and setters

```
public class Person {
  private String name; // private = restricted access

  // Getter
  public String getName() {
    return name;
  }

  // Setter
  public void setName(String newName) {
    this.name = newName;
  }
}
```

2. Inheritance.

Function: Inherit attributes and methods from one class to another.

Two categories:

- Subclass (child) – the class that inherits from another.
- Superclass (parent) – the class being inherited from.

```
class Vehicle {
  protected String brand = "Ford";         // Vehicle attribute
  public void honk() {                     // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle class) and the
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```

3. Polymorphism.

Function: When a class inherits from another, it can use those methods to perform different tasks.

```java
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}

class Dog extends Animal {
  public void animalSound() {
    System.out.println("The dog says: bow wow");
  }
}
```

4. Abstraction.

Function: Show only "relevant" data and "hide" unnecessary details of an object from the user.

To access the abstract class, it must be inherited from another class.

```java
// Abstract class
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}
```

2. DATABASES BASICS

In this section we will know basic concepts of database their relationships and the datatypes.

2.1 Basic concepts

1. Database and DBMS.

Function: A **database (DB)** is an organized collection of structured information, or data, typically stored electronically in a computer system.

A **Database Management System (DBMS)** is software for creating, manipulating, and administering a database.

Translation: Base de datos y Sistema Gestor de Base de Datos

2. Table.

Function: A collection of rows and columns.

Also known as **entities** or **relations**.

Translation: Tabla, entidad o relación.

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

3. Row.

Function: A **row** contains data of a single item or a record in a table.

Also known as **record** or **tuples**.

Translation: Fila, registro o tupla.

Bat	Man	321 Cavern Ave	Gotham	54
Donald	Duck	555 Quack Street	Mallard	65
Wonder	Woman	987 Truth Way	Paradise	39

4. Column.

Function: A **column** contains data representing a specific characteristics of the records in the table.

Also know as **fields** or **attributes**.

Translation: Columna, campo o atributo.

Address
123 Fantasy Way
321 Cavern Ave
987 Truth Way
555 Quack Street
567 Carrot Street
999 Acme Way
234 Purrfect Street
543

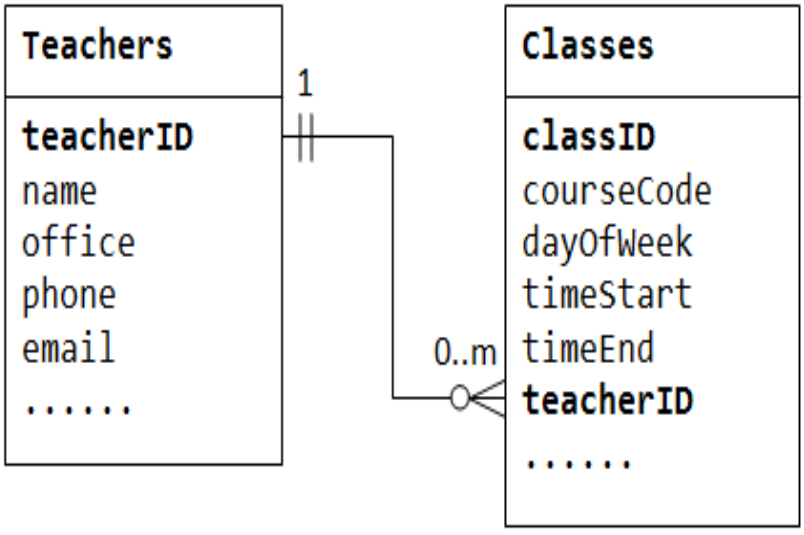
2.2 Relationships

1. Relationship.

Function: A **relationship** is a link between two tables.

They make it possible to find data in one table that pertains to a specific record in another table.

Translation: Relación.



2. Primary Key (PK).

Function: The **primary key** constraint uniquely identifies each record in a table.

Rules:

- Contain UNIQUE values
- Cannot contain NULL values.
- A table can have only ONE PK (consisting of one or multiple colums).

Translation: Clave primaria

"Persons" table:

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

3. Foreign Key (FK).

Function: The **foreign key** is used to link two tables together.

It is a field (or collection of fields) in one table that refers to the PK in another table.

Child table → table with FK.

Parent/Referenced table → table with PK

Translation: Clave ajena o foránea

"Persons" table:

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

2.3 Datatypes

Character Datatype.

Types:

- **CHAR**: stores a fixed length string. e.g. *car_plate CHAR(8)*
- **VARCHAR2**: stores a variable length string. e.g. *full_name VARCHAR2(15)*

Numeric Datatype.

NUMBER: stores numeric values that can be negative or positive.

The NUMBER datatype also has precision and scale:

- Precision: the number of digits in a number.
  e.g. *price NUMBER(3)*

- Scale: the number of digits to the right of the decimal point.
  e.g. *price NUMBER(6, 2)*

Date Datatype.

- **DATE**: stores dates and times in a table.

The standard Oracle date format is DD-MON-YY:
e.g. '14-NOV-98'

3. SQL STATEMENTS

Now we will study the main commands of SQL statements such as DML (database manipulation language), DDL (database definition language) and DCL (database control language).

3.1 Database Manipulation Language (DML)

1. Select.

Function: Extract or **retrieve** data from a database.

We can retrieve records from a single or multiple tables with record grouping and sorting.

We can use keywords like "ORDER" and "WHERE" to sort and filter the results.

```
SELECT column1, column2
FROM table_name;

SELECT * FROM CUSTOMERS
WHERE CUSTOMERID = 1;
```

2. Insert Into.

Function: Add new rows to a table.

To take into account:

- **Order** of values.
- Value **type.**
- **Number** of values.

```
INSERT INTO table_name (column1, column2)
VALUES (value1, value2);

INSERT INTO CUSTOMERS (NAME, CITY)
VALUES ('CHRIS', 'OSLO');
```

3. Update.

Function: This statement is used to **modify** existing rows.

It is possible to update several records at the same time.

We need to use the "WHERE" clause if we do not want to update all the records.

```
UPDATE table_name_1;
SET column1 = value1, column2 = value2
WHERE conditions;

UPDATE CUSTOMERS
SET NAME = 'ALFRED', CITY = 'SIDNEY'
WHERE CUSTOMERID = 1;
```

4. Delete.

Function: To **delete** existing records in a table.

We cannot delete a single field. We can delete rows.

If we omit "WHERE" in our statement, we will delete all the rows in the table.

```
DELETE FROM table_name_1
WHERE conditions;

DELETE FROM CUSTOMERS
WHERE NAME = 'ALFRED';
```

3.2 Database Definition Language (DDL)

1. Create table.

Function: This statement is used to **create a table** a new a table in a database.

We need to specify the **names** and **datatypes** of the columns of the table.

```
CREATE TABLE table_name_1 (
  colum1 datatype,
  colum2 datatype,
  colum3 datatype
);

CREATE TABLE PERSONS (
  PersonID NUMBER,
  LastName VARCHAR2,
  FirstName VARCHAR2,
  Address VARCHAR2
);
```

2. Alter.

Function: With the alter statement we can **modify** the columns of a table.

We can perform 3 different operations:

- **Add**: adding a new column.
- **Modify**: changing a column's datatype.
- **Drop**: removing a column.

```
ALTER TABLE table_name_1
ADD column_name datatype;

ALTER TABLE table_name_1
MODIFY column_name datatype;

ALTER TABLE table_name_1
DROP COLUMN column_name;
```

3. Drop table.

Function: To move a table to the recycle bin or remove it entirely from the database, we use the **drop table** statement.

We need to indicate the name of the table, but also the constraints if we want to remove the foreign key constraints from other tables.

```
DROP TABLE table_name_1;

DROP TABLE PERSONS;
```

3.3 Database Control Language (DCL)

1. Grant.

Function: We use **grant** to provide any user access privileges or other privileges for the database.

Some of these privileges are:

- Create **session**.
- **Create/Drop** table.
- Provide user with **space** on tablespace to store tables.

```
GRANT CREATE TABLE
ON USERS TO MIKE;

GRANT ALL
ON COMPANY TO JEFF;
```

2. Revoke.

Function: We use it to **revoke** privileges already granted to these users.

Some of these privileges are:

- Create **session**.
- **Create/Drop** table.
- Provide user with **space** on tablespace to store tables.

```
REVOKE SELECT
ON COMPANY FROM KEITH;

REVOKE UPDATE
ON COMPANY FROM KEITH;
```

4. ENTITY-RELATIONSHIP (ER) MODEL

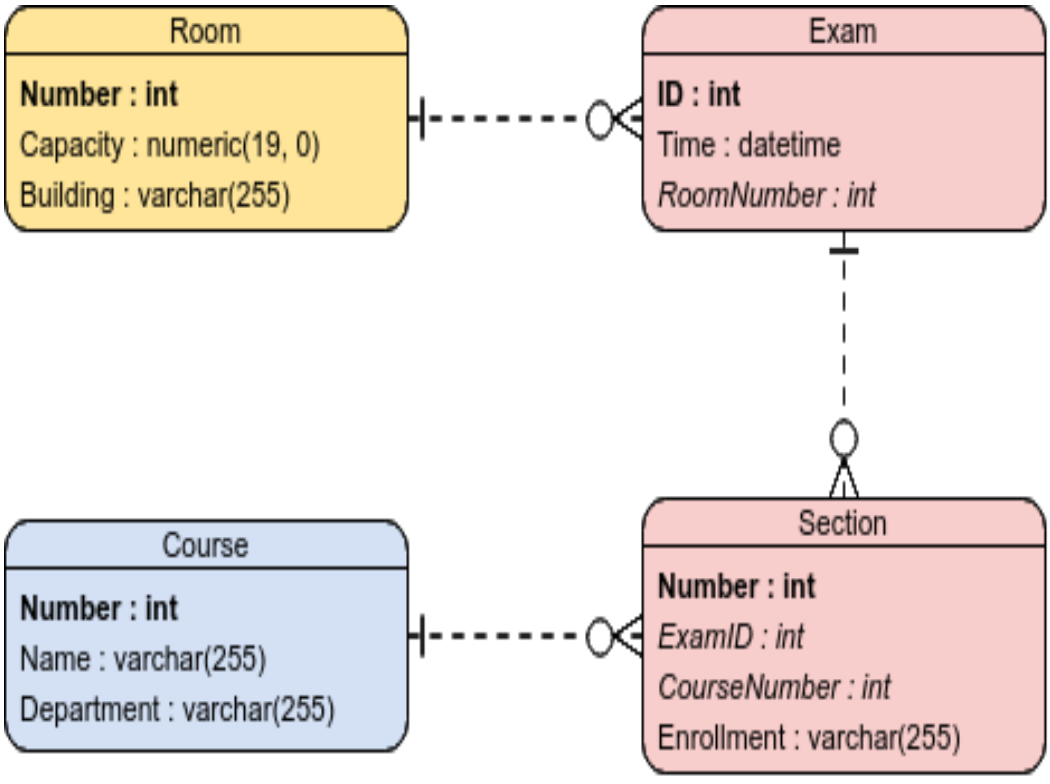
In the following sections we will study the basic concepts, components and types of relationships of the ER Model.

4.1 Basic concepts

1. What is it?

The **Entity-Relationship (ER) model** represents real-world entities and the relationship between them.

ER modeling helps you to analyze data requirements systematically to produce a well-designed database. Therefore, it is considered a best practice to **complete ER modeling before implementing your database**.

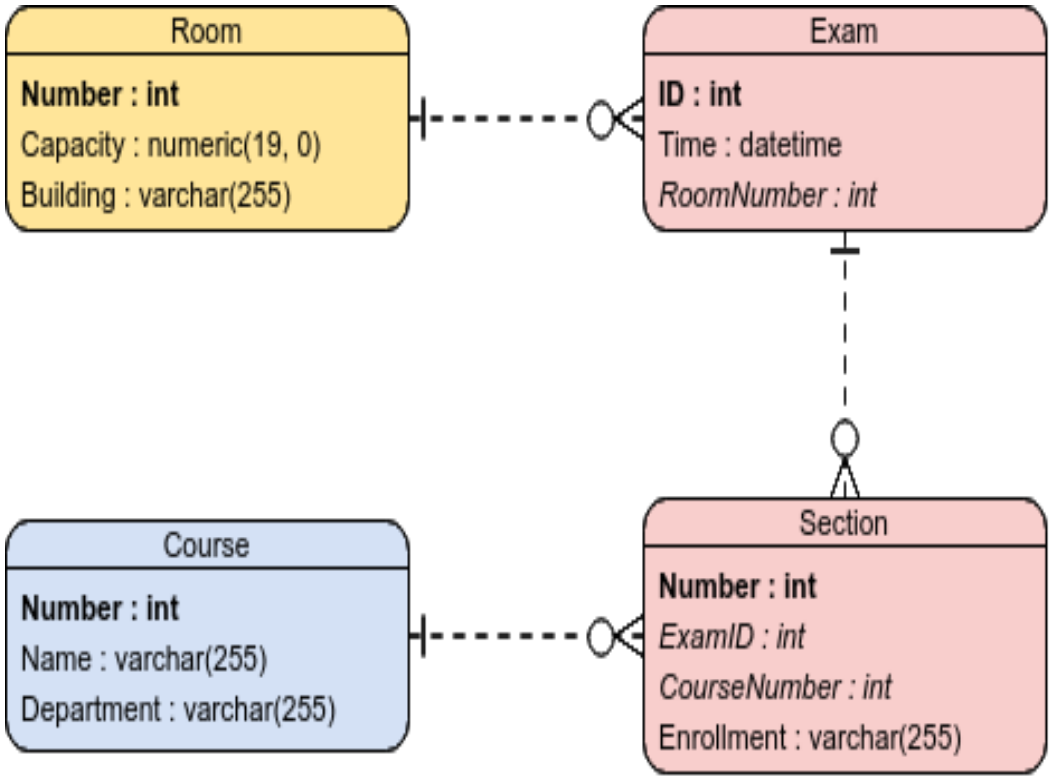


2. How is it represented?

In order to display the ER model, **ER Diagrams (ERD)** are used.

The ERD helps to explain the logical structure of the database.

We can identify the entities existing in the system and the relationships between them.



4.2 Components of the ERD

1. Entity.

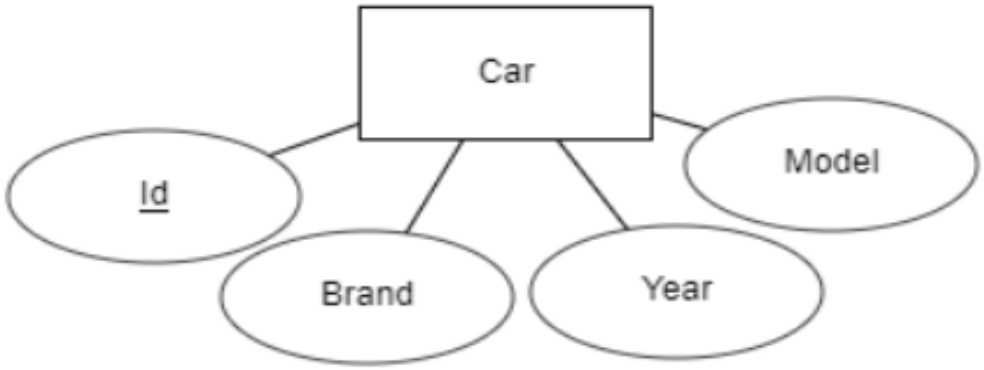
Definition: An **entity** is a thing that exists either physically or logically.

Features:

- Entities are **nouns**.
- Must have **an attribute**.
- Must have **a unique key**.

Examples:
E.g. An object with physical existence (**a lecturer, a student, a car**)
E.g. An object with conceptual existence (**a course, a job, a position**)

Examples:



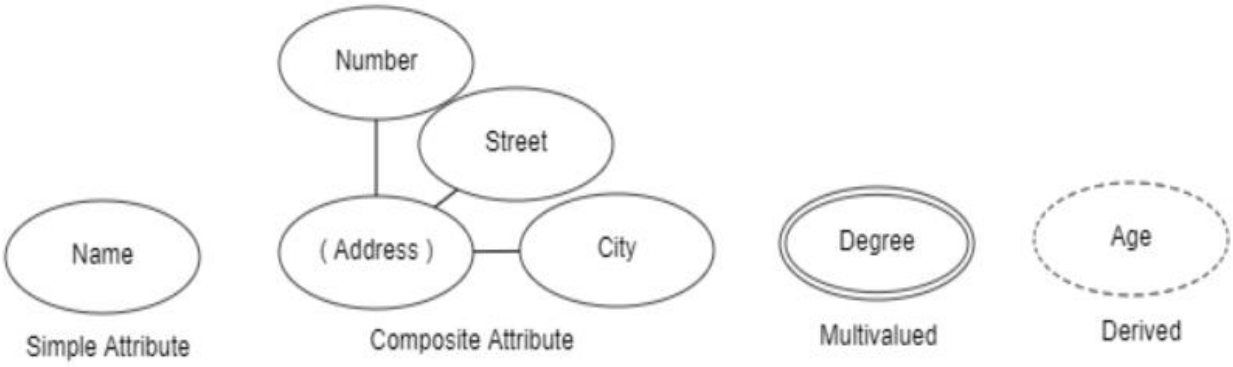
2. Attributes.

Definition: An **attribute** is a property. Each entity is described by a set of attributes.

Types:

- **Simple attributes**. Also called single-valued attributes.
  E.g. Name = {John}
- **Composite attributes**. Attributes that consist of other attributes.
  E.g. Address: Number = {42}, Street = {'Baker St'}, City = {'Cork'}
- **Multivalued attributes**. These have a set of values for each entity.
  E.g. Degrees = BA, BSc, PhD
- **Derived attributes**. Contain values calculated from other attributes.
  E.g. Age can be derived from the attribute Birthdate.

Examples:



3. Relationship

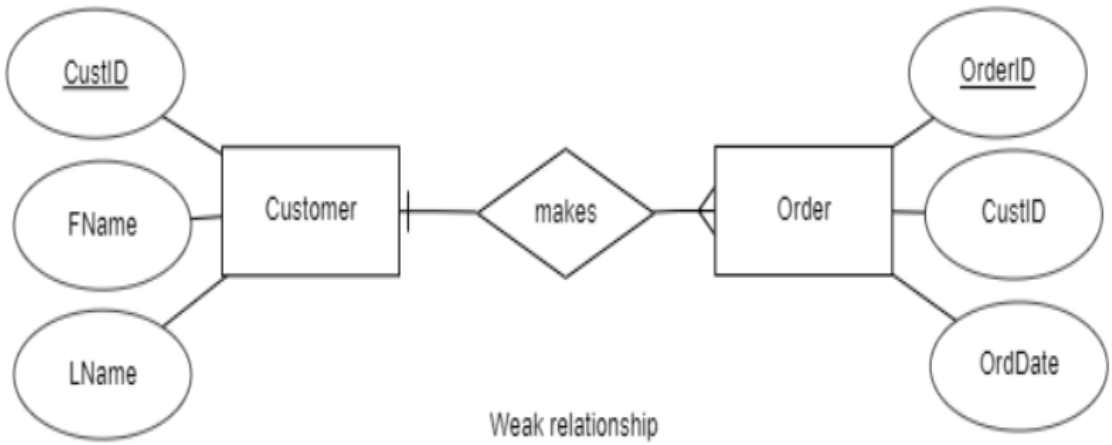
Definition: A **relationship** is an association among two or more entities. They are used to connect related information between tables. Relationships are **verbs**.

Strength:

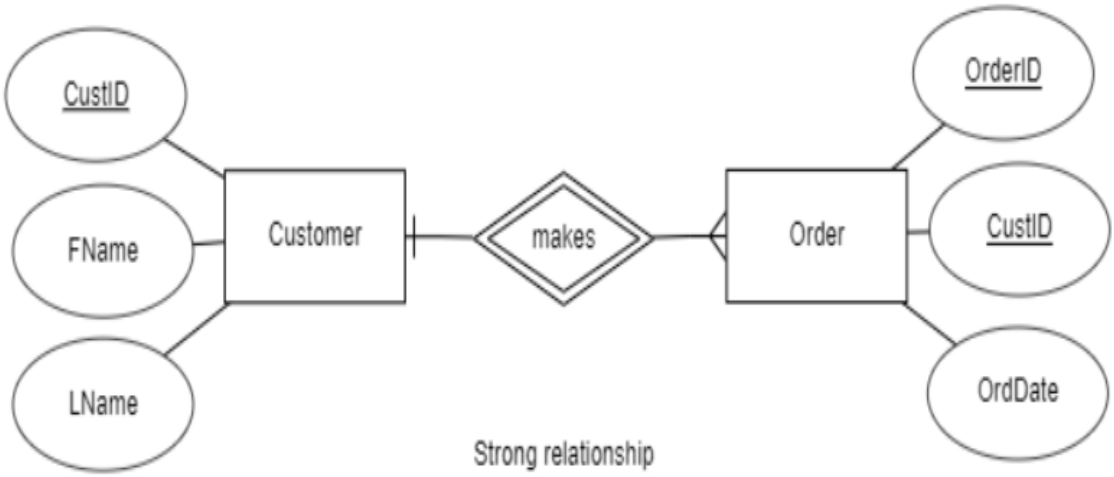
- **Weak relationship**: When the PK of the related entity does NOT contain a PK component of the parent entity.

- **Strong relationship**. When PK of the related entity contains the PK component of the parent entity.

Examples:



Weak relationship



Strong relationship

4.3 Types of Relationships

One to One (1:1)

Definition: A **one to one (1:1)** relationship is the relationship of one entity to only one other entity, and vice versa. It should be **rare** in any relational database design. In fact, it could indicate that two entities actually belong in the **same table**.

E.g. A person has a passport.

One to Many (1:M)

Definition: A **one to many (1:M)** relationship should be the norm in any relational database design and is found in all relational database environments.

E.g. A doctor has many patients.

Many to Many (M:N)

Definition: A **many-to-many (M:N)** relationship exists between two entities if for one entity instance there may be multiple records in the other table, and vice versa.

E.g. Many student study many subjects, and many subjects are studied by many students.

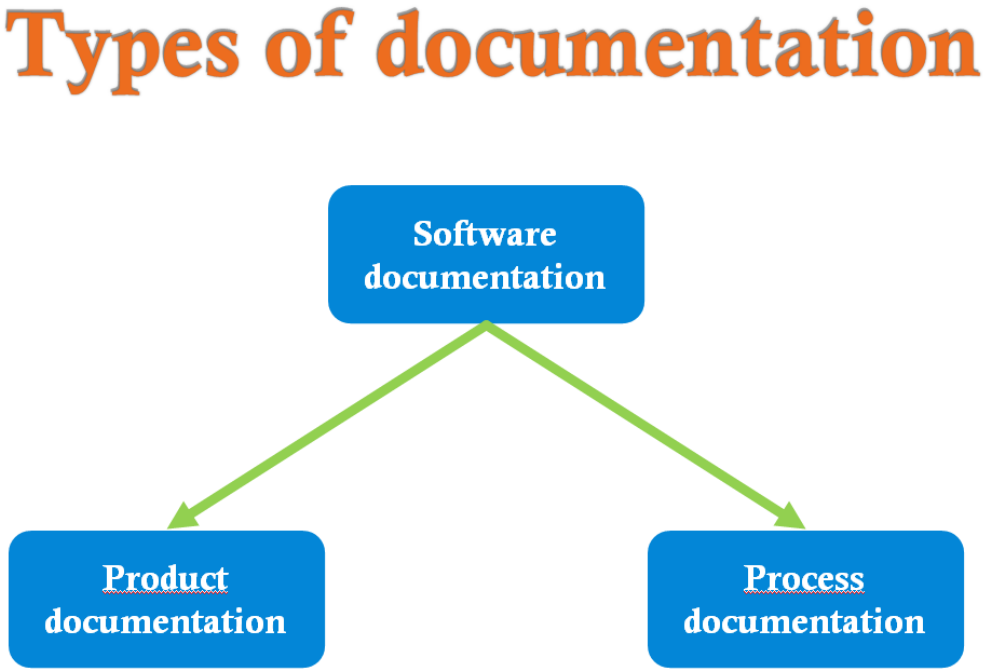
5. DOCUMENTATION BASICS

Documentation is as important as any other part of the process of developing an application.

Why is this important? There are several reasons:

- Maintenance
- Collaboration
- Clarity
- Assessment
- Improvement

The types of documentation will be explained next.



5.1 Product documentation

What is it?
Description of the actual product being developed. It contains instructions on how to perform tasks with it.

Documents:

- **System** documentation
- **User** documentation

Translation: Documentación del producto

a. System documentation

What is it?
An overview of the system and helps engineers to understand the technology used in the application.

Documents:
- Requirement document
- Architecture design
- Source code
- QA docs
- Maintenance and help guide

Translation: Documentación del sistema

Requirement document
- Function: what the system should do.
- Includes: purpose, features, functionalities and behaviour.

Architecture design
- Function: how the system is designed.
- Includes: design, diagrams, modules, components.

Source code
- Function: explains how the code works
- Includes: brief descriptions of elements and the code.

QA docs
- Function: show all the testing performed.
- Includes: test strategy, plans, case specifications, and checklists.

Maintenance and help guide
- Function: describe known problems with the system and solutions.
- Includes: maintenance guide.

b. User documentation

What is it?
Manuals prepared for end-users of the product and **system administrators**.

What does it include?
- End-users documents
- System administrators documents

Translation: Documentación del usuario

End-users

- Function: explain how to operate the software
- Includes: FAQs, video tutorials, assistance, support portals.

System Administrators

- Function: explain how to install it and help with product maintenance
- Includes: functional description and sys admin guide.

5.2 Process documentation

What is it?
Documents produced during development that describe the whole process.

What does it include?

- Project plans
- Test schedules
- Meeting notes
- Reports: how time and human resources were used.

Translation: Documentación del proceso