# *DBMS-Refresher-Notes*

## *1. Data Attribute*

A **data attribute** is a property or characteristic of an entity. Think of it as a column in a table.

**Example:**
Imagine a **student** database. The entity is **Student**, and its attributes could be:

- **Name** (John Doe)
- **Age** (20)
- **Email** (john@example.com)

Each row in the database stores specific values for these attributes.

---

## *2. Entity*

An **entity** is an object or thing in the real world that can be stored in a database.

**Example:**

- A **student** in a school database
- A **car** in a vehicle database
- A **product** in an e-commerce store

Each entity has attributes (like a student has a name and roll number).

---

## *3. Cardinality*

**Cardinality** defines the number of relationships between entities.

**Types:**

1. **One-to-One (1:1):** A person has **one passport**.
2. **One-to-Many (1:N):** A teacher can teach **many students**.
3. **Many-to-Many (N:N):** A student can enroll in **many courses**, and a course can have **many students**.

---

## *4. DB Design (Database Design)*

Database design is about organizing data efficiently. It ensures that:
✅ Data is stored properly
✅ There are no duplicate records
✅ Queries run fast

**Example:**
If you're making a library database, you should have tables like:

1. **Books (ISBN, Title, Author## 1. Data Attribute****

A **data attribute** is a property or characteristic of an entity. Think of it as a column in a table.

**Example:**
Imagine a **student** database. The entity is **Student**, and its attributes could be:

- **Name** (John Doe)
- **Age** (20)
- **Email** (john@example.com)

Each row in the database stores specific values for these attributes.

---

---

## *5. Relationships*

Relationships connect different tables in a database.

- **One-to-One (1:1)** → Each person has **one** unique ID card.

- **One-to-Many (1:N)** → A customer can place **many** orders.
- **Many-to-Many (N:N)** → A student enrolls in **many** courses, and a course has **many** students.

**Example:**
A **Users** table and an **Orders** table are connected because one user can place many orders.

## One-to-one

A one-to-one relationship in a database means that each record in one table is linked to exactly one record in another table, and vice versa.

Imagine a system where each **employee** has **one unique company car**.

#### Table: Employees

| EmployeeID | Name |
|---|---|
| 1 | Alice |
| 2 | Bob |

#### Table: CompanyCars

| CarID | EmployeeID | CarModel |
|---|---|---|
| 101 | 1 | Toyota Camry |
| 102 | 2 | Honda Civic |

Here, each employee is associated with **one car**, and each car is assigned to **one employee**.

## One-to-many

A one-to-many relationship in a database means that one record in a table can be associated with multiple records in

#### Example Scenario:

Imagine a system where **one teacher** can teach **many students**, but each student has **only one teacher**.

#### Table: Teachers

| TeacherID | Name |
|---|---|
| 1 | Mr. Shah |
| 2 | Ms. Rao |

#### Table: Students

| StudentID | Name | TeacherID |
|---|---|---|
| 101 | Aditi | 1 |
| 102 | Rohan | 1 |
| 103 | Meera | 2 |

Here:

- Mr. Shah teaches Aditi and Rohan.
- Ms. Rao teaches Meera.

## Many-to-many

A many-to-many relationship in a database means that multiple records in one table can be associated with multiple records in another table.

#### Example Scenario:

Imagine a system where **students can enroll in multiple courses**, and **each course can have multiple students**.

#### Table: Students

| StudentID | Name |
|---|---|
| 1 | Aditi |
| 2 | Rohan |

#### Table: Courses

| CourseID | Title |
|---|---|
| 101 | Math |

| CourseID | Title |
| --- | --- |
| 102 | Science |

#### Linking Table: Enrollments

| StudentID | CourseID |
| --- | --- |
| 1 | 101 |
| 1 | 102 |
| 2 | 101 |

Here:

- Aditi is enrolled in Math and Science.
- Rohan is enrolled in Math.
- Math has both Aditi and Rohan.

---

## *6. Data Types*

Each column in a database has a **data type**, which defines what kind of values it can store.

**Common Data Types:**

- **INT** → Numbers (e.g., age, price)
- **VARCHAR** → Text (e.g., names, addresses)
- **DATE** → Dates (e.g., 2025-03-07)
- **BOOLEAN** → True or False

**Example:**
A **Student** table may have:

- **Name** → `VARCHAR(100)`
- **Age** → `INT`
- **Enrollment Date** → `DATE`

---

## *7. Difference Between Primary and Unique Key*

| Feature | Primary Key | Unique Key |
|---|---|---|
| Purpose | Uniquely identifies each record | Ensures uniqueness but allows NULL values |
| NULL Allowed? | No | Yes |
| Number per Table | Only **one** primary key | Multiple unique keys allowed |

**Example:**
A **Student** table may have:

- **StudentID (Primary Key)** → Always unique
- **Email (Unique Key)** → Ensures no two students have the same email

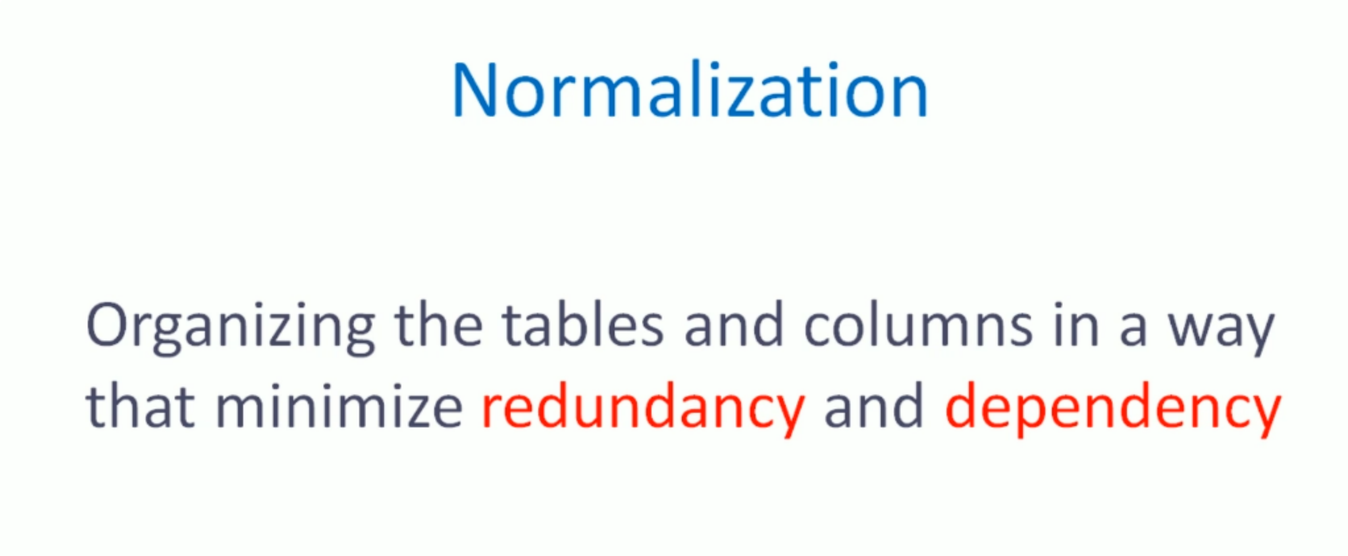
## *8. 1:1, 1:N, N:N Relationships*

**One-to-One (1:1):** Each employee has **one** company car.
**One-to-Many (1:N):** A teacher teaches **many** students.
**Many-to-Many (N:N):** Students enroll in **many** courses, and a course has **many** students.

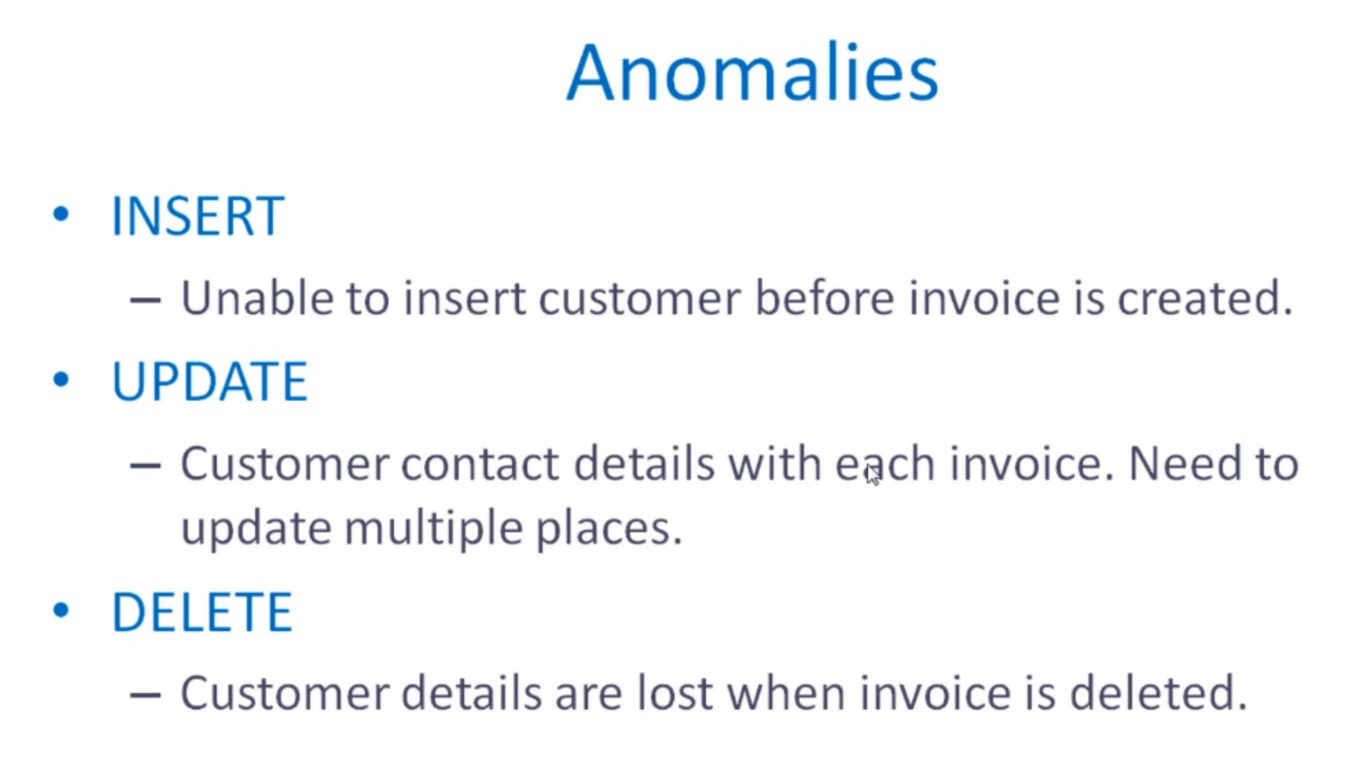
**Example:**
A **Library Database** may have:

- **Books** (BookID, Title)
- **Students** (StudentID, Name)
- **BorrowedBooks** (StudentID, BookID, BorrowDate) → (N:N relationship)

## 9. *Normalization* and its Types




## Anomalies




## 1NF

# First normal form(1NF)

- Attributes/Columns are uniquely named.
- The order of columns and rows is insignificant.
- Every row and column intersection contains exactly one value of the applicable domain, and nothing else.
- No duplicate rows

**2NF**

# Second Normal Form (2NF)

*Candidate Key* *Non-prime attributes*

| <u>a</u> | <u>b</u> | c | d | e |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

*Note: A non-prime attribute of a table is an attribute that is not part of any candidate key of the table.*

Every non-prime attribute of the table is dependent on the complete key (whole of a candidate key)

**Example**

PARTS SUPPLY

| Supplier Id | Part Id | Part Name | Quantity |
|---|---|---|---|
| supplier1 | BA | Base Assembly | 10 |
| supplier1 | CS | Crank Shaft | 5 |
| supplier2 | BA | Base Assembly | 40 |
| supplier2 | BU | Battery Unit | 35 |
| supplier3 | BA | Base Assembly | 8 |
| supplier3 | CS | Crank Shaft | 2 |
| supplier3 | BU | Battery Unit | 12 |

- Here there are 2 candidate keys
  - supplier id
  - part id
- The non prime attributes here are
  - Part name
  - Quantity
- 2nf says that every non prime attribute is dependent on the complete key (both the candidate keys)
- Lets check each non prime attribute one by one
  - **Part name**
    - Part name is determined by Part ID
    - But its not determined by supplier ID as well
    - So its partially dependent on the complete key
  - **Quantity**
    - Quantity is determined by the combination of supplier ID and quantity
- Transforming it into 2NF
  - We need to split into 2 tables where 2nf is satisfied

PARTS SUPPLY

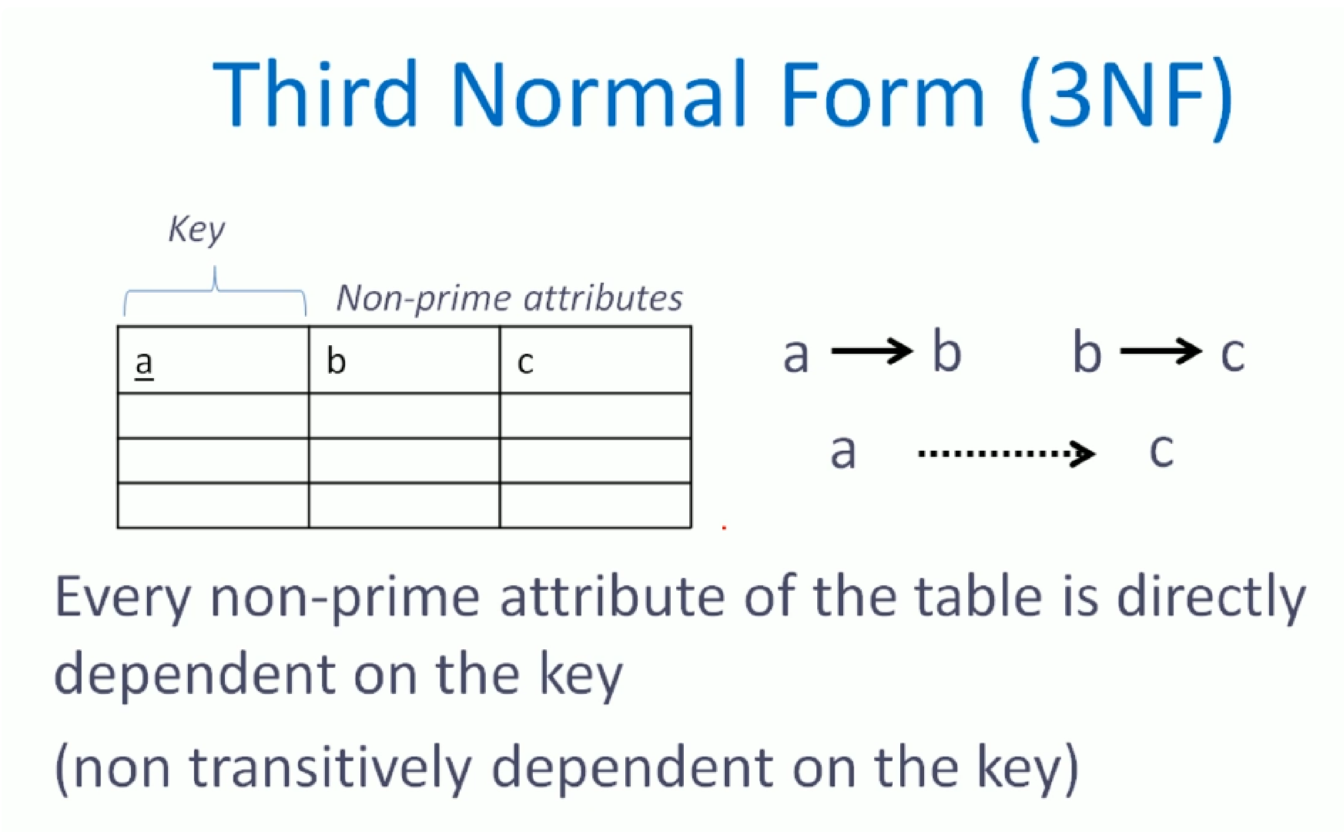
| Supplier Id | Part Id | Quantity |
|---|---|---|
| supplier1 | BA | 10 |
| supplier1 | CS | 5 |
| supplier2 | BA | 40 |
| supplier2 | BU | 35 |
| supplier3 | BA | 8 |
| supplier3 | CS | 2 |
| supplier3 | BU | 12 |

PARTS

| Part Id | Part Name |
|---|---|
| BA | Base Assembly |
| CS | Crank Shaft |
| BU | Battery Unit |

- Now quantity is determined by combination of supplier ID and part ID
- Part name is determined by Part ID
- Both tables follow 2nf

## 3NF




**Example**

SALES

| Order No | Customer | Product | Coupon Code | Discount(%) | Net Amount |
|---|---|---|---|---|---|
| ORD1001 | Alice Cooper | Video Camera | NEWYEAR40 | 40 | 600.00 |
| ORD1002 | Barbara Park | Keyboard | NEWYEAR40 | 40 | 30.00 |
| ORD1003 | Cynthia Nixon | LCD Monitor | SUMMER30 | 30 | 70.00 |
| ORD1004 | Mohan Lal | Mobile phone | SUMMER30 | 30 | 350.00 |
| ORD1005 | Zoya Afrose | Hard Disk | SPRING25 | 25 | 450 |
| ORD1006 | Steve Smith | Printer | SPRING25 | 25 | 150.00 |
| ORD1007 | Barbara Park | Network Router | NOCOUPON | 00 | 45.00 |

- Here our key is Order No, other columns are non prime attributes
- Here Discount depends on Coupon Code
- And coupon code depends on Order number
- So this is Transitively dependent
    - Discount -> Coupon Code -> Order Number
    - This is not allowed in 3NF
- We need to split the tables such that the transitive dependencies are removed
- We need a separate table for the columns in the transitive dependency
    - The columns are Coupon code and discount

COUPON

| Coupon Code | Discount(%) |
|---|---|
| NEWYEAR40 | 40 |
| SUMMER30 | 30 |
| SPRING25 | 25 |
| NOCOUPON | 00 |

SALES

| Order No | Customer | Product | Coupon Code | Net Amount |
|---|---|---|---|---|
| ORD1001 | Alice Cooper | Video Camera | NEWYEAR40 | 600.00 |
| ORD1002 | Barbara Park | Keyboard | NEWYEAR40 | 30.00 |
| ORD1003 | Cynthia Nixon | LCD Monitor | SUMMER30 | 70.00 |
| ORD1004 | Mohan Lal | Mobile phone | SUMMER30 | 350.00 |
| ORD1005 | Zoya Afrose | Hard Disk | SPRING25 | 450 |
| ORD1006 | Steve Smith | Printer | SPRING25 | 150.00 |
| ORD1007 | Barbara Park | Network Router | NOCOUPON | 45.00 |

- Now the Tables follow 3NF

## BCNF

# Boyce-Codd Normal Form(BCNF)

- Developed by Raymond F. Boyce and E.F. Codd
- A table is in BCNF if every determinant is a candidate key

- Determinant is the value which can determine other attributes

PRODUCT SPECIALIST ASSIGNMENT

| Customer Id | Product Id | Specialist Id | Date | Status |
|---|---|---|---|---|
| C10001 | LED Television | Peter | 01-Dec-2014 | Active |
| C10002 | Personal Computer | Reshmi | 03-Jan-2014 | Active |
| C10002 | Network Router | Mallik | 11-Feb-2014 | Active |
| C10003 | Mobile Phone | Roonie | 07-Nov-2014 | Inactive |
| C10005 | LED Television | Peter | 17-Oct-2014 | Inactive |
| C10006 | Network Router | Mallik | 19-Mar-2014 | Active |

- Here the determinants are
- CustomerID, SpecialistID,ProductId are deteminants
- Combination of customerid, productid and customerid,specialist ID can be keys
- We need to split the tables

- PRODUCT SPECIALIST

| Specialist Id | Product Id | Specialization Level |
|---|---|---|
| Peter | LED Television | L1 |
| Reshmi | Personal Computer | L1 |
| Mallik | Network Router | L2 |
| Roonie | Mobile Phone | L1 |

PRODUCT SPECIALIST ASSIGNMENT

| Customer Id | Specialist Id | Date | Status |
|---|---|---|---|
| C10001 | Peter | 01-Dec-2014 | Active |
| C10002 | Reshmi | 03-Jan-2014 | Active |
| C10002 | Mallik | 11-Feb-2014 | Active |
| C10003 | Roonie | 07-Nov-2014 | Inactive |
| C10005 | Peter | 17-Oct-2014 | Inactive |
| C10006 | Mallik | 19-Mar-2014 | Active |

, Category)**

1. **Students (ID, Name, Email)**
2. **BorrowedBooks (StudentID, ISBN, BorrowDate, ReturnDate)**

This way, the **BorrowedBooks** table links students and books without storing duplicate data.

---

## *10. Types of databases*

### 1. Relational Databases (RDBMS)

- Structure: Tables with rows and columns.
- Examples: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.
- Use Case: Structured data with clear relationships (e.g., banking, inventory systems).

### 2. NoSQL Databases

- These are non-relational and are designed for flexibility and scalability.

#### a. Document Stores

- Structure: JSON-like documents.
- Examples: MongoDB, CouchDB.

- Use Case: Content management, catalogs.

**b. Key-Value Stores**

- Structure: Key-value pairs.
- Examples: Redis, DynamoDB.
- Use Case: Caching, session management.

**c. Column-Family Stores**

- Structure: Columns grouped into families.
- Examples: Apache Cassandra, HBase.
- Use Case: Big data, real-time analytics.

**d. Graph Databases**

- Structure: Nodes and edges.
- Examples: Neo4j, Amazon Neptune.
- Use Case: Social networks, recommendation engines.

## *11. Benefits of relational databases*

- Data Integrity and Accuracy
  - Enforced through constraints (e.g., primary keys, foreign keys, unique constraints).
  - Ensures that data is consistent and accurate across related tables.
- Data Relationships
  - Ideal for representing relationships between entities (e.g., customers and orders).
  - Foreign keys help maintain referential integrity.
- Scalability
  - Can handle large volumes of data with proper indexing and optimization.
  - Suitable for enterprise-level applications.