**Normalization**

* Normalization is the process of removing duplicate data from a table.
* It divides a larger table into smaller tables and links them using relationships.

**Why Do We Need Normalization?**

* The main reason for normalization is removing anomalies.

**What is an Anomaly?**

* An anomaly is a side effect caused by duplicate data or large tables.

**Example Table**

| **S\_id** | **s\_name** | **s\_contact** | **c\_id** | **c\_name** | **f\_name** | **f\_contact** |
| --- | --- | --- | --- | --- | --- | --- |
| 101 | John | 1234567890 | 201 | Math | Smith | 9876543210 |
| 102 | Alice | 1234567891 | 202 | Science | Brown | 9876543211 |
| 101 | John | 1234567890 | 203 | History | Smith | 9876543210 |

**Types of Anomalies**

* Insertion Anomaly:  
  Occurs when you cannot insert a new record because some required data is missing.  
  For example, if s\_id is the primary key and cannot be null, we cannot insert a new course without valid student data.
* Update Anomaly:  
  Occurs when updating a single piece of data requires multiple changes.  
  For example, if faculty data is repeated several times, updating the faculty contact requires multiple updates.
* Deletion Anomaly:  
  Occurs when deleting data unintentionally removes other important data.  
  For example, deleting a student entry might also delete related course and faculty information.

To avoid these anomalies, we divide one big table into smaller related tables.

**Smaller Normalized Tables Example**

| **S\_id** | **s\_name** | **s\_contact** |
| --- | --- | --- |
| 101 | John | 1234567890 |
| 102 | Alice | 1234567891 |

| **C\_id** | **c\_name** |
| --- | --- |
| 201 | Math |
| 202 | Science |
| 203 | History |

| **f\_name** | **f\_contact** |
| --- | --- |
| Smith | 9876543210 |
| Brown | 9876543211 |

**Types of Functional Dependencies**

1. Total Functional Dependency:  
   All non-key attributes depend on the entire primary key attribute.  
   Example: R={empno,ename,job,sal,hiredate}*R*={*empno*,*ename*,*job*,*sal*,*hiredate*}
2. Partial Functional Dependency:  
   Non-key attributes depend on part of a composite primary key.  
   Example: R={empno,empname,job,sal,deptno,dname,loc}*R*={*empno*,*empname*,*job*,*sal*,*deptno*,*dname*,*loc*}
3. Transitive Functional Dependency:  
   A non-key attribute depends on another non-key attribute, which depends on the key.  
   Example: R={empno,ename,job,sal,deptpincode}*R*={*empno*,*ename*,*job*,*sal*,*deptpincode*}

**Definition:**  
A transitive functional dependency occurs in a relation when a non-key attribute depends on another non-key attribute, which in turn depends on the primary key. Formally:  
If A→B and B→C, then A→Cis called a transitive dependency, where:

* A is the primary key,
* B and C are non-key attributes,
* The dependency is indirect through B.

**Example Relation R={empno,ename,job,sal,deptpincode} *R*={*empno*,*ename*,*job*,*sal*,*deptpincode*}**

* Attributes:
  + empno: Employee number (Primary Key)
  + ename: Employee name
  + job: Job title
  + sal: Salary
  + dept\_pincode: Department pincode (address)

**Functional Dependencies:**

1. empno → dept\_pincode — Employee number determines department pincode.
2. empno → ename, job, sal — Employee details depend on empno.
3. dept\_pincode → (some other info about department, e.g., location) (implicit to show dependency between non-key attributes)

**How Transitive Dependency Occurs:**

* empno determines dept\_pincode (since each employee belongs to one department).
* dept\_pincode determines something else non-key about that department (like location).
* Therefore, empno indirectly determines that information through dept\_pincode.

Simplified, the key point is:

* empno → dept\_pincode
* dept\_pincode → location (example)
* So, empno → location (transitive dependency)

Because location depends on dept\_pincode, which itself depends on employee number, this is an indirect or transitive dependency.

**Normalization Forms**

**First Normal Form (1NF)**

* A table is in 1NF if there is no duplicate data.
* It has a primary key.
* All values are atomic (indivisible).

**Example:**

| **Roll\_no** | **Name** | **Course** |
| --- | --- | --- |
| 1 | James | Laptop |
| 2 | Rodes | Tablet |
| 3 | Williams | Shirts |
| 4 | Smith | Watches |

**Second Normal Form (2NF)**

* Table is in 1NF.
* No partial dependency of any column on part of the primary key.

**Example:**

Student Table

| **student\_id** | **student\_name** |
| --- | --- |
| 121 | James |
| 122 | Rodes |

Course Table

| **course\_id** | **course\_name** |
| --- | --- |
| 32 | Laptop |
| 33 | Tablet |

**Third Normal Form (3NF)**

* Table is in 2NF.
* There is no transitive dependency between non-key attributes.

**Example:**

| **emp\_no** | **emp\_name** | **job** | **salary** | **dept\_pincode** |
| --- | --- | --- | --- | --- |
| 101 | Alice | Dev | 50000 | 12345 |

**Summary**

* Once tables are normalized, relations can be built using foreign keys.
* Normalization helps reduce redundancy and anomalies in the database.