**Normalization**

Normalization is the process of organizing data in database to reduce redundancy and improve data integrity. it helps make the database more efficient and prevents issues like data anomalies.

**Needed of normalization?**

* Remove duplicate date
* Ensures data consistency
* Optimizes storage and retrieval

**Normalization Forms**

**1. First Normal Form (1NF)**

* Each column contains atomic values
* Each column contains values of the same type.
* Each row is uniquely identifiable using a primary key.0

**Quary**

Create table students(id int primary key, Name varchar(20), phone number varchar(50) );

Example

|  |  |  |
| --- | --- | --- |
| ID | NAME | Phone Number |
| 1 | Neha | 1212121212 |
| 2 | Hema | 1234567890 |
| 3 | Radha | 1987654321 |

**2. Second Normal form (2NF)**

* It is in 1NF
* No partial dependency (every non-key column depends on the whole primary key)

**Query**

Create table student\_subjects( student id int primary key, subject varchar);

Insert into student\_subjects values (1, ‘english’) (2,’math’);

Example

|  |  |
| --- | --- |
| Student id | subject |
| 1 | English |
| 2 | Math |

Create table subject\_teacher (subject varchar primary key, Teacher varchar);

Insert into subject\_teacher values (‘math’, ‘mr.A’), (‘science’, ‘mr.B’);

|  |  |
| --- | --- |
| Subject | Teacher |
| English | Mr.A |
| Math | Mr.B |

**3. Third Normal from**

* It is in 2NF
* No transitive dependency (Non-key attributes must depend only on the primary key, not on other non-key attributes)

**Query**

Create table students(student id int primary key, name varchar, zipcode varchar);

Insert into student values (1, ‘Neha’, 1002);

|  |  |  |
| --- | --- | --- |
| Student id | Name | Zip code |
| 1 | Neha | 1002 |

Create table zipcode(zipcode varchar primary key, city varchar);

Insert into student values (1002,’Nagpur’);

|  |  |
| --- | --- |
| Zip code | city |
| 1002 | Nagpur |

**Advantages**

* No duplicate data – saves space b storing data efficiently.
* Easy updates-change data in one place, and it updates everywhere.
* Better data Accuracy-no risk of wrong or mismatches information.

**Disadvantages**

* Complex queries-Data is in multiple tables, so queries need more JOINS.
* Slower performance- Too many tables can mark searches slow.
* Hard for beginners-understanding relationship between tables is different.
* **Denormalization**

Denormalization is the process of combination multiple tables into a single table to improve query performance by reducing the need for JOINS.

**Need of Denormalization?**

* Faster data retrieval:- avoids complex joins, making queries run quickly.
* Reduce joins:- stores related data together, reducing the need to combine multiple tables.
* Useful in reporting and analytics:- helps fetch data faster for dashboards and reports.

**Types of Denormalization**

**1. Adding redundant data**

Duplicate data is stored in multiple tables to avoid joins

**Query:-**

Create table student(student id int primary key, name varchar, class id int);

**Advantages:**

* Reduces the need for frequent JOIN operations.
* Speeds up query execution for commonly accessed foreign key data.
* Improves performance in distributed databases where joins are expensive.

**Disadvantages:**

* Data redundancy leads to increased storage consumption.
* Requires careful synchronization to prevent inconsistencies.

**2. pre-computed Aggregations**

Store aggregated values instead of calculating on each query.

**Combination tables**

**Query**

select \* from Sales\_Summary where Month = '2024-03';

**3.storing computed columns**

Instead of calculating values at query time, store pre-computed results in a table.

**Query;-**

Alter table orders add column total price decimal(10,2);

Update orders set total price = quantity\*unit price;

* Speeds up queries by avoiding on-the-fly calculations.
* Reduces CPU workload for frequent calculations.
* Simplifies queries by using a direct column instead of computed expressions.

Disadvantages:

* Requires additional storage for computed values.
* Needs regular updates if source data changes (risk of outdated values).
* Write operations become slower due to the extra update step.

**4.Duplicating Data from Parent Tables**

Instead of performing joins, store frequently accessed parent table data in the child table.

**Queary**

Alter table orders add column customer name varchar(255);

Update orders 0 join customers c on o.customer id = c.customer\_id set o.customer\_name = c.name;

**Advantages:**

* Reduces the need for costly JOIN operations.
* Faster retrieval for frequently accessed parent table data.

**Disadvantages:**

* Increased storage due to redundant data.
* Risk of data inconsistency if the parent table changes.
* Requires additional update mechanisms to keep data in sync.

**5.Pre-joining table**

Combine multiple tables into a single table for faster queries.

Create table Sales\_Summary as Select region, month(order\_date) as month, sum(total\_price) AS total\_sales from Orders group by region, month;

**Advantages:**

* Boosts performance for analytics and reporting queries.
* Reduces the need for complex aggregation queries at runtime.
* Optimized for read-heavy workloads.

**Disadvantages:**

* Additional storage is required for summary data.
* Data may become outdated if not refreshed properly.