CHA

NORMALISATION

Introduction:

Normalization:

A process of organizing the data in the database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

A process of organizing data into tables in such a way that the results of using the database are always unambiguous and as intended. Such normalization is intrinsic to relational database theory. It may have the effect of duplicating data within the database and often results in the creation of additional tables.

Advantages of Normalization

- Elimination of data redundancy makes the database to be compact reducing the overall amount of space a database consumes.
- Enforcement of referential integrity on data ensuring data to be consistent across all tables.
- Maintenance becomes easier and faster since the data are organized logically in a normalized database in a flexible way.

- Searching and sorting of records is easier and faster because data will appear in a separate, smaller table when a database is normalized allowing us to easily find them.

Difference between Normalization and Denormalization

- Normalization and denormalization are two processes that are completely opposite.
- Normalization is the process of dividing larger tables into smaller ones reducing the redundant data, while denormalization is the process of adding redundant data to optimize performance.
- Normalization is carried out to prevent database anomalies.
- Denormalization is usually carried out to improve the read performance of the database, but due to the additional constraints used for denormalization, writes (i.e., insert, update and delete operations) can become slower. Therefore, a denormalized database can offer worse write performance than a normalized database.
- It is often recommended that you should “normalize until it hurts, denormalize until it works”.
- Normalizing data means eliminating redundant information from a table and organizing the data so that future changes to the table are easier.

TYPES OF NORMALISATION

Types of Normal Form

- 1NF
- 2NF
- 3NF
- BCNF
- 4NF

1st Normal Form (1NF)

The values in each column of a table are atomic (No multi-value attributes allowed). Each table has a primary key: minimal set of attributes which can uniquely identify a record

There are no repeating groups: two columns do not store similar information in the same table.

Example of a table not in 1NF :

Genres	Name	Movie

Marvel	Kevin Feige	The Avengers
		Captain America
DCEU	Zack Snyder	Batman Vs Super Man
		Suicide Squad

This table contains Attribute values which are not single. This is not in Normalised form.

To make it into 1NF we have to decompose the table into atomic elements.

Table in 1NF after eliminating:

Genres	**Name**	**Movie**
Marvel	Kevin Feige	The Avengers
Marvel	Kevin Feige	Captain America
DCEU	Zack Snyder	Batman Vs Superman
DCEU	Zack Snyder	Suicide Squad

Second Normal Form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of the table.

Prime attribute :an attribute, which is a part of the prime-key, is known as a prime attribute.

Non-prime attribute : an attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

Example of a table not in 2NF:

Studio	Movie	Budget	city
Marvel	Avengers	100	New York
Marvel	Captain America	120	New York
DCEU	Batman Vs Superman	150	Gotham
DCEU	Suicide Squad	75	Gotham

Here the Primary key is (studio, movie) and the city depends only on the studio and not on the whole key.
So, this is not in 2NF form.

Solution of 2 NF

Old Scheme ->{Studio, Movie, Budget, City}
New Scheme -> {Movie, Studio, Budget}
New Scheme ->{Studio, City}

Movie	Studio	Budget
The Avengers	Marvel	100
Captain America	Marvel	120
Batman Vs Superman	DCEU	150
Suicide Squad	DCEU	75

Studio	City
Marvel	New York
DCEU	Gotham

Now the 2 tables are in 2NF form.

Third normal form 3 NF

This form dictates that all non-key attributes of a table must be functionally dependent on a candidate key i.e., there can be no interdependencies among non-key attributes.

For a table to be in 3NF, there are two requirements

- The table should be second normal form
- No attribute is transitively dependent on the primary key

Example of a table not in 3nf

Studio	City Temp	Studio City
Marvel	96	New York
DCEU	99	Gotham
Fox	96	New York
Paramount	95	Hollywood

Here Studio is the primary key and both studio temp and city depends entirely on the Studio.

1. **Primary Key** ->**{Studio}**
2. **{Studio}** -> **{Studio City}**
3. **{Studio City}** ->**{City Temp}**
4. **{Studio}** -> **{City Temp}**
5. **CityTemp transitively depends on Studio hence violates 3NF**

It is called **transitive dependency**.

Solution of 3NF

Old Scheme -> {Studio, Studio City, City Temp}

New Scheme-> {Studio, Studio City}

New Scheme ->{Studio City, City Temp}

Studio	Studio City
Marvel	New York
DCEU	Gotham

FOx	New York
Paramount	Hollywood

Studio City	**City Temp**
New York	96
Gotham	95
Hollywood	99

Boyce Codd Normal Form (BCNF) – 3.5NF

BCNF does not allow dependencies between attributes that belong to candidate keys.

BCNF is a refinement of the third normal form in which it drops the restriction of a non-key attribute from the 3rd normal form.

Third normal form and BCNF are not same if the following conditions are true:

- The table has two or more candidate keys
- At least two of the candidate keys are composed of more than one attribute
- The keys are not disjoint i.e. The composite candidate keys share some attributes.

Example of BCNF

Scheme -> {Movie Title, Movie ID, Person Name, Role, Payment }

Key1 -> {Movie Title, Person Name}

Key2 ->{Movie ID, Person Name}

Movie Title	Movie ID	Person Name	Role	Payment
The Avengers	M101	Robert Downey Jr.	Tony Stark	200m
The Avengers	M101	Chris Evans	Chris Rogers	120m
Batman Vs Superman	D101	Ben Affleck	Bruce Wayne	180m
Batman Vs Superman	D101	Henry Cavill	Clarke Cent	125m
A walk to remember	P101	Mandy Moore	Jamie Sullivan	50m

Dependency between Movie ID & Movie Title Violates BCNF

Solution of BCNF

Place the two candidate primary keys in separate entities

Place each of the remaining data items in one of the resulting entities according to its

dependency on the primary key.

New Scheme ->{Movie ID, Person Name, Role, Payment}

New Scheme ->{Movie ID, Movie Title}

Movie ID	Person Name	Role	Payment
M101	Robert Downey Jr.	Tony Stark	200m

M101	Chris Evans	Chris Rogers	125m
D101	Ben Affleck	Bruce Wayne	175m
D101	Henry Cavill	Clarke Cent	120m
P101	Mandy Moore	Jamie Sullivan	50m

Movie ID	**Movie Title**
M101	The Avengers
D101	Batman VS Superman
P101	A walk to remember

4 NF

Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.

It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce- Codd Normal Form (BCNF).

It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

Example of 4NF

Scheme -> {MovieName, Screening City, Genre)

MovieName	Screening City	Genre
The Avengers	Los Angeles	Sci-Fi
The Avengers	New York	Sci-Fi
Batman vs Superman	Santa Cruz	Drama
Batman vs Superman	Durham	Drama
A Walk to remember	New York	Romance

Many Movies can have the same Genre and Many Cities can have the same movie. So, this table violates 4NF .

Solution of 4NF

Move the two multi-valued relations to separate tables Identify a primary key for each of the new entities.

New Scheme -> {MovieName, Screening City}

New Scheme -> {MovieName, Genre}

MovieName	Screening City
Batman vs Superman	Santa Cruz
The Avengers	Los Angeles

A Walk to remember	New York
Batman vs Superman	Durham
The Avengers	New York

MovieName	**Genre**
Batman vs Superman	Drama
The Avengers	Sci-Fi
A Walk to remember	Romance

We split the table into two tables with one multivalued value in each.