UNIT III Notes

Schema Refinement

Schema Refinement (also called as Normalization) is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies. Here Redundancy refers to repetition of same data or duplicate copies of same data stored in the Database.
Redundancy is at the root of several problems associated with relational schemas. Integrity constraints, in particular functional dependencies, can be used to identify schemas with such problems and to suggest refinements.
One of the main refinement technique is decomposition (replacing ABCD with, say, AB and BCD, or ACD and ABD).

Decomposition

Decomposition is a process of decomposing a larger relation into smaller relations. Each of smaller relations contain subset of attributes of original relation.
Anomalies refers to the problems occurred during database operations because of poorly planned and normalised databases.

Types of decomposition	

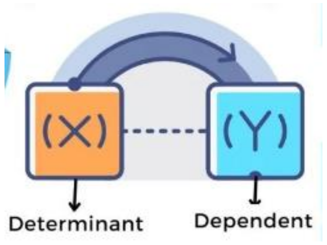
Loseless decomposition	It is a process of decomposing relation into two or more relations and ensures that
	▪ No information is lost during decomposition
	▪ When the sub relations are joined back, the same relation is obtained that was decomposed
	For Example,
	Consider a relation R is decomposed into R1 and R2
	R = (A, B, C) // R has three attributes namely A, B and C
	R1 = (A, B) // R1 has two attributes namely A and B
	R2 = (B, C) // R2 has two attributes namely B and C
	• common attribute B must be a super key of sub relations either R1 or R2
	• if it contains a duplicate value then the Lossless-join decomposition is not possible
	• R1U R2 is same as tuples of R, it is loseless decomposition

Lossy decomposition	• Consider there is a relation R which is decomposed into sub relations R1 , R2 , …. , Rn. • This decomposition is called lossy join decomposition when the join of the sub relations does not result in the same relation R that was decomposed • For lossy join decomposition, we always have $\mathbf{R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots \bowtie R_n \supset R}$ where ⋈ is a natural join operator

Functional Dependency

Functional dependency in DBMS refers to the relationship between attributes in a database table. Functional dependency is a form of integrity constraint that can identify schema with redundant storage problems and to suggest refinement. It typically exists between the primary key and non-key attribute within a table which would be denoted by

$X \rightarrow Y$



For example, Emp_Id → Emp_Name where employee name is dependent on employee id. Armstrong axioms defines the set of rules for reasoning about functional dependencies and also to infer all the functional dependencies on a relational database.

Rule -1: Augmentation	• PR -> QR, if P -> Q • If P holds Q and R is a set of attributes, then PR holds QP
Rule -2: Reflexivity	• P -> Q, if Q is a subset of P • If P is a set of attributes and Q is a subset of P, then P holds Q
Rule -3: Transitivity	• If P -> Q and Q -> R, then P -> R i.e. a transitive relation
Rule -4: Union	• If P→Q and P→ R then P → QR • If P holds Q and P holds R then P holds QR
Rule -5: Decomposition	• If P→ QR then P→Q and P → R • If P holds QR and P holds Q then P holds R

Types of functional dependencies

- ***Trivial functional dependency***:-If X􀃆Y is a functional dependency where Y subset X, these type of FD's called as trivial functional dependency.

- ***Non-trivial functional dependency***:-If X□Y and Y is not subset of X then it is called non-trivial functional dependency.
- ***Completely non-trivial functional dependency***:-If X□Y and X∩Y=Φ(null) then it is called completely non-trivial functional dependency.

Prime and non-prime attributes

Attributes which are parts of any candidate key of relation are called as prime attribute, others are non-prime attributes.

Candidate Key:

Candidate Key is minimal set of attributes of a relation which can be used to identify a tuple uniquely. Types of candidate keys:

1. simple(having only one attribute)
2. composite(having multiple attributes as candidate key)

Super Key:

Super Key is set of attributes of a relation which can be used to identify a tuple uniquely

Dependency preservation

It ensures

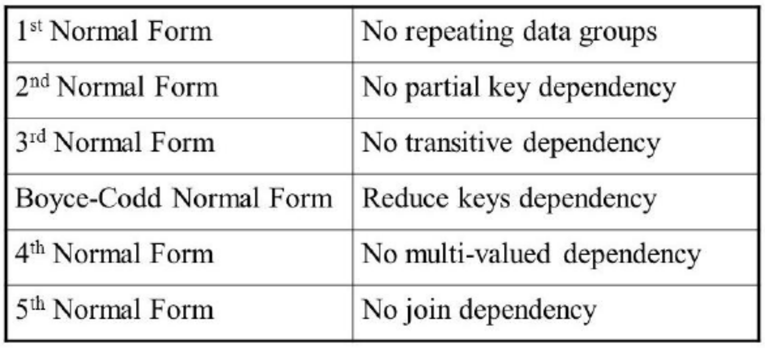
- None of the functional dependencies that holds on the original relation are lost.
- The sub relations still hold or satisfy the functional dependencies of the original relation.

Normalization:

Normalization is a process of designing a consistent database with minimum redundancy which support data integrity by grating or decomposing given relation into smaller relations preserving constraints on the relation.

Normalisation removes data redundancy and it will helps in designing a good data base which involves a set of normal forms as follows -

1. First normal form(1NF)
2. Second normal form(2NF)
3. Third normal form(3NF)
4. Boyce coded normal form(BCNF)
5. Forth normal form(4NF)
6. Fifth normal form(5NF)
7. Sixth normal form(6NF)
8. Domain key normal form.



1st Normal Form	No repeating data groups
2nd Normal Form	No partial key dependency
3rd Normal Form	No transitive dependency
Boyce-Codd Normal Form	Reduce keys dependency
4th Normal Form	No multi-valued dependency
5th Normal Form	No join dependency

Advantages of normalization

- Helps to minimize data redundancy.
- Provides greater overall database organization.
- Ensure data consistency within the database.
- Much more flexible database design.
- Allows to enforce the concept of relational integrity.

Disadvantages

- We cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF
- it is time-consuming and difficult to normalize relations of a higher degree
- Careless decomposition may lead to a bad database design

First normal form (1 NF):

A relation is said to be in first normal form if it contains all atomic values. An attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

For example, in the employee table, phone no has multiple values, against 1 NF

Emp_ID	**Emp_name**	**Phone_no**	**State**
A101	John	123456789 157856511	TN
A102	Hari	2548796 7598456	KA

Now 1 NF is ensured by organizing

Emp_ID	**Emp_name**	**Phone_no**	**State**
A101	John	123456789	TN
A101	John	157856511	TN
A102	Hari	7598456	KA
A102	Hari	2548796	KA

Second normal form

A relation is said to be in second normal form if it is in first normal form and all non-key attributes are fully functional dependent on the primary key.

For example

Example: **Student id**	**Student name**	**Project Id**	**Project name**

Here (student id, project id) are key attributes and (student name, project name) are non-prime attributes. It is decomposed as two table namely student and project

Student id	Student name	Project id

Project id	Project name

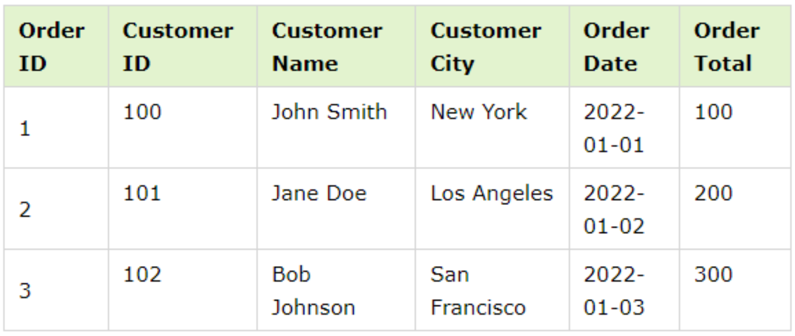
Third normal form (3 NF)

A relation is said to be in third normal form , if it is already in second normal form and no transitive dependencies exists. A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y

1. X is a super key.

2. Y is a prime attribute (each element of Y is part of some candidate key).

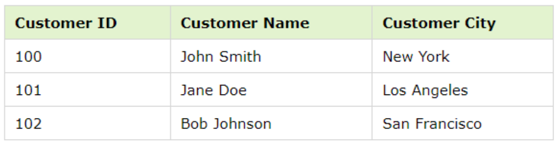
For example,



Order ID	Customer ID	Customer Name	Customer City	Order Date	Order Total
1	100	John Smith	New York	2022-01-01	100
2	101	Jane Doe	Los Angeles	2022-01-02	200
3	102	Bob Johnson	San Francisco	2022-01-03	300

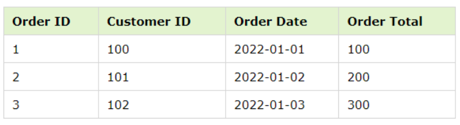
"Customer City" is transitively dependent on the primary key. That is, it depends on "Customer ID", which is not part of the primary key, instead of depending directly on the primary key "Order ID"

Table 1: Customers



Customer ID	Customer Name	Customer City
100	John Smith	New York
101	Jane Doe	Los Angeles
102	Bob Johnson	San Francisco

Table 2: Orders



Order ID	Customer ID	Order Date	Order Total
1	100	2022-01-01	100
2	101	2022-01-02	200
3	102	2022-01-03	300

Boyce normal form (BCNF)

It applies to tables with more than one candidate key. A relation is in BCNF if every determinant in the table is a candidate key. ie LHS is super key. If a table contains single candidate key, the 3NF and BCNF are equivalent.

A relation is in BCNF if in every non-trivial functional dependency X –> Y, X is a super key

For Example,

Emp_id	**Name**	**Dept_id**	**Qualification**	**Salary**
E102	Hari	SA01	BE	25000
E104	Kumar	SB01	BE	21400

The table has 3 determinates: Emp_id, Dept_id and qualification, and (Emp_id , Dept_id) is candidate keys. So it is not in BCNF.

It is divided into salary and employee table

Emp_id	**Dept_id**	**Salary**
E102	SA01	25000
E104	SB01	21400

Emp_id	**Name**	**Qualification**
E102	Hari	BE
E104	Kumar	BE

Fourth normal form (4NF)

When attributes in a relation have multi-valued dependency, further Normalization to 4NF and 5NF are required. Let us first find out what multi-valued dependency is

A ***multi-valued dependency*** is a typical kind of dependency in which each and every attribute within a relation depends upon the other, yet none of them is a unique primary key.

We illustrate this with an example. Consider a vendor supplying many items to many projects in an organization. The following are the assumptions:

1. A vendor is capable of supplying many items.
2. A project uses many items.
3. A vendor supplies to many projects.
4. An item may be supplied by many vendors

A multi valued dependency exists here because all the attributes depend upon the other and yet none of them is a primary key having unique value.

Vendor_id	**Item_id**	**Project_id**
V101	IX01	P01
V102	IX02	P01
V101	IX01	P02
V103	IX03	P02

Item IX01 is duplicated here and if item is not finalized for vendor, it would be empty
The problem is reduced by expressing this relation as two relations in the Fourth Normal Form (4NF).
A relation is in 4NF if it has no more than one independent multi valued dependency or one independent multi valued dependency with a functional dependency.
In the example, vendor relation is decomposed into vendor_item and vendor_project relations and ensures in 4 NF

Vendor_id	**Item_id**
V101	IX01
V102	IX02
V101	IX01
V103	IX03

Vendor_id	**Project_id**
V101	P01
V102	P01
V101	P02
V103	P02

Join Dependencies and 5th Normalization Form (5 NF)

Join Dependencies

A relation is said to have join dependency if it can be recreated by joining multiple sub relations and each of these sub relations has a subset of the attributes of the original relation.

If the join of R1 and R2 over Q is equal to relation R then we can say that a join dependency exists, where R1 and R2 are the decomposition R1 (P, Q) and R2 (Q, S) of a given relation R (P, Q, S). R1 and R2 are a lossless decomposition of R.

5th Normalization Form (5 NF)

It is a generalization of Multi Valued Dependency. A relation is said to have join dependency if it can be recreated by joining multiple sub relations and each of these sub relations has a subset of the attributes of the original relation.

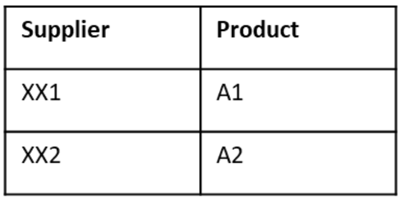
A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless. 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy. 5NF is also known as Project-join normal form (PJ/NF).

Suppose if the join of R1 and R2 over Q is equal to relation R then we can say that a join dependency exists, where R1 and R2 are the decomposition R1 (P, Q) and R2 (Q, S) of a given relation R (P, Q, S). R1 and R2 are a lossless decomposition of R

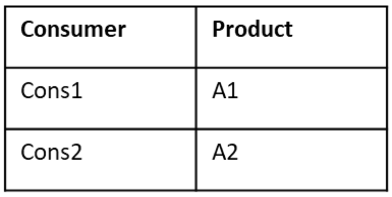
For example, a vendor has the following attributes

Supplier	**Product**	**Consumer**
XX1	A1	Cons1
XX2	A2	Cons2

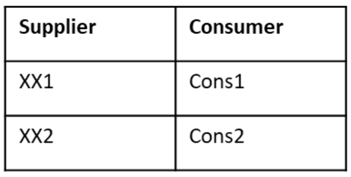
There is join dependency exists in this table, therefore it is not in 5NF. Now it is reduced into three relation namely supplier_product, consumer_product and supplier_cosnumer. So when these three relations are joined, original vendor table can be restored.



Supplier	Product
XX1	A1
XX2	A2



Consumer	Product
Cons1	A1
Cons2	A2



Supplier	Consumer
XX1	Cons1
XX2	Cons2

Let us finally summarize the normalization steps we have discussed so far.

1 NF	A relation is in 1NF if it contains an atomic value.
2 NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3 NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
Boyce-codd (BCNF)	A stronger definition of 3NF is known as Boyce Codd's normal form.
4 NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5 NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.