4. Relational Database design

Functional dependency:-
let R be the relation schema and let x and y are non empty sets of attribute in R we say that an instants r of R satisfies the Functional dependency $x \rightarrow y$,
$x - y$ (x Functionality of y)
If the following hold every pairs of touple t_1 and t_2 is are,

$t_1X = t_2X$

$t_1Y = t_2Y$

it means that if two touple agree on the value in attribute x, i.e, for both the rows values for attribute x is same then they ~~ha~~ must also have same values for the attribute.

Undesirable properties of RDB design:-
The goal of a relation database is to generate a set of relation schemas that allows us to store information without unnecessary duplication or redundancy, at the same time allowing us to retrive information easily.

i) Repetation of information (Redundancy)
ii) Inability to represent certain information (thereby loss of correct information)

Redundancy is storing the same information in more stored repeatedly in more than one place leading to need of larger storage space.

PAGE No.
DATE / /

i) Redundant storage:- Same information is stored repeatly in more than one places leading to need of larger storage space

ii) Update anamalies:- If one copy of such repeated data is updated, an inconsistancy is created unless all copies are similarly updated

iii) Insertation anamalies:- It may not be possible to store some information unless some other information is stored as will.

iv) Deletion anamalies:- It may not be possible to delete some information without losing some other information as well.

Armstrong Axioms:-

Rule:-	
1) Reflexivity rule 2) Augumentation rule 3) Transitivity rule	there is a set of Inference rules or a set of Axioms, also Called as Armstrong's axioms that is used to compute F^+

① Reflexivity rule:- if α is set of attributes and $\beta < \alpha$ then $\alpha \to \beta$ hold

② Augumentation rule:- $\alpha \to \beta$ & γ is the set of attributes, then $\gamma\alpha \to \gamma\beta$ hold $\to$ $A \to B$, $A \to C = A C \to BC$

③ Transitivity rule:- If $\alpha \to \beta$ holds then $\alpha \to \gamma$ holds
$A \to B$, $B \to C \Rightarrow A \to C$

PAGE No.
DATE / /

Ex 1) If $R = (A, B, C, G, H)$ is relational schema and $F_x = (A \to B, A \to C, B \to H)$ Then the functional dependancy $A \to H$ is logically implied if t_1, t_2 of two touples of R then by $A \to B$ mens $t_1[A] = t_2[A]$ and thus by $A \to B = t_1[B] = t_2[B]$ then by $B \to H$ that $t_1[H] = t_2[H]$

→ 1) Augumentation rule.
2) Transitivity rule.

Ex 2) Consider the relational schema $R = \{A, B, C, D, E\}$ and set of functional dependency defined on R. F as i.e. $F = \{A \to B, CD \to E, A \to C, B \to D, E \to A\}$ and the set of functional dependency use the Axioms rule we compute the F^+.

→ Given: $F = (A \to B, CD \to E, A \to C, B \to D, E \to A)$

F^+
- $A \to B, B \to D \Rightarrow A \to D$ --- (Transitivity rule)
- $CD \to E, A \to C \Rightarrow A \to E$ ------ (psudo-Transitivity rule)
- $CD \to E, E \to A \Rightarrow CD \to A$ --- (Transitivity rule)
- $B \to D, CD \to E \Rightarrow B \to E$ ----- (psudo-Transitivity rule)
- $A \to B, A \to C = AC \Rightarrow BC$ --- (Augumentation rule)

clousure of Attributes:-

Ex, To elustrate the following example compute the (AG^+) with the FD's then,

$FD = \{A \to B, A \to C, CG \to H, CG \to I, B \to H\}$

AG^+
- $A \to B = B$, $A \to C = C$ } $A \to BC$
- $CG \to H = CH$, $CG \to I = CI$ } $CH \to CI$
- $B \to H = BH$

$\therefore (AG)^+ = (BCHI)$
$(AG^+) = (AGBCHI)$

* Concept of decomposition:- Many problems arising from redundancy can be addressed by replacing the relation with a number of 'smaller' relations. Each of the smaller relation contain a strict subset of the attributes of the original relation. This process is called decomposition. larger → smaller

Concept of de-compotion:-

Ex, To remore the redundancy in library, Library
(c-no, name, adress, class, book issused)
class^{det}(class, b-allowed)

c-no	name	adress	class	book-issused
1-101	Mr Joshi	kantnude pune	Normal	1
1-103	Ms Deepa	karlc road	Normal	2
1-104	Mr. Bhide	ABC	silver	3
1-107	Mr Amit	Tilak road	gold	6
1-109	Ms sima	Laxmi rod	silver	5

Ass.

class	Book-allowed
Normal	2
silver	5
Gold	7

decomposition table:-

closure of an Attribute set of function Dependency; F^{+}

let R be a relational schema and F be set of functional dependencies defined on R.
The set F contains all functional dependencies that are semantically obvious.
But in addition, there are numerous other functional dependencies that hold all legal relational instances that satisfy dependencies in F. such functional dependencies are logically implied from F, and as defined as the clousure of F
Thus the set of all functional dependencies logically implied from F is called as the clousure of 'F' and is denoted by F^{+}.

* Trivial dependencies:-
Some functional dependencies are said to be trivial because they are statisfy by all relations.

PAGE No.
DATE / /

* clousre of Attributeset:-

A super key for a Relation R is set of Attributes can uniquely identify every tuple of R. This implies that the superkeys for a given Relation R are preciely those subsets K of the set of attributes of R such that the functional dependency K→A, holds true for every attribute A of R.

superkey, we must device an Algorithm for computing the set of attributes functionally determined by α.

Ex,

1) To illustrate how to Algorithm works, we shall use the algorithm to compute $(AG)^+$ with the FDs,

A→B,
A→C,
CG→H,
CG→I,
B→H

The input to the Algorithm will be,

F = { A→B, A→C, CG→H, CG→H, CG→I, B→H }

and α = {AG}

we start with result = AG

The First time we execute the while loop to test each FD, we find that,

A→B, causes us to include B in result.

∵ A→B is in F, A ⊆ result, so result = result ∪ B

A→C, causes result to become ABCG

CG→H, Causes result to become ABCGH

CG→I, Causes result to become ABCGHI

2) Consider the relation, R = {SSN, Ename, Pnumber, Pname, Plocation, Hours}

and the set

PAGE No.
DATE / /

$F = \{SSN \rightarrow Ename, Pnumber \rightarrow \{Pname, Plocation\}$
$\{SSN, Pnumber\} \rightarrow Hours\}$

Then,

$\{SSN\}^+ = \{SSN, Ename\}$

$\{Pnumber\}^+ = \{Pnumber, Pname, Plocation\}$

$\{SSN, Pnumber\}^+ = \{SSN, Pnumber, Ename, Pname, Plocation, Hours\}$

* Desirable properties of Decomposition:-

i) Loss less join Decomposition:-
when we decompose a relation into a number of smaller relations, it is crucial that the decomposition be lossless. lossless means no data is lost as a result of breaking a relation into a set of smaller relations. It also implies that any new unmeaningful or spurious data shoudn't get added as a result of decomposition.

* Algorithms:-
let R be a relation schema and let F be a set of Functional dependencies on R. Let R_1 & R_2 form a decomposition of R

$R_1 \cap R_2 \rightarrow R_1$

$R_1 \cap R_2 \rightarrow R_2$

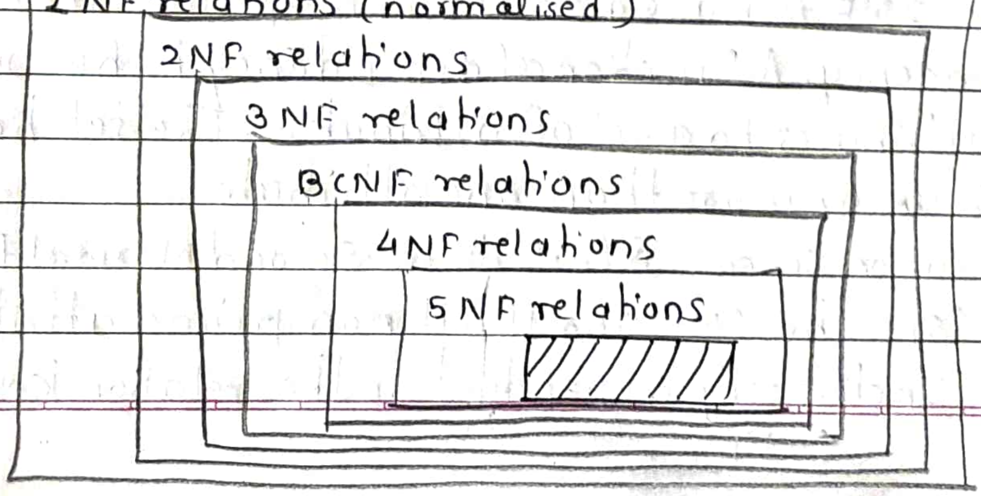
* Dependency Preservation:-
Another goal in relational-database design is dependency preservation. when an update is made to the database, the systeam should be able to cheak that update will not create an illegal relation - that is, one that dose not statisfy all the given functional dependencies.

Ex,

1) $R(A, B, C, D)$ and $F = \{A \to B, A \to C, C \to D\}$ is decomposed into
$R_1 = \{A, B, D\}$ with FDs
$F_1 = \{A \to B, A \to D\}$ and
$R_2 = \{B, C\}$ with Functional dependencies
$F_2 = \{\ \}$
This is not a dependency preserving and also not loss less join decomposition.

2) Let $R(A, B, C, D)$ and $F = \{A \to B, A \to C, C \to D\}$
R is decomposed into
$R_1 = (A, B, C)$ with the FD's
$F_1 = \{A \to B, A \to C\}$
$R_2 = (C, D)$ with the FD's
$F_2 = \{C \to D\}$
$F' = F_1 \cup F_2 = \{A \to B, A \to C, C \to D\}$
Hence $F'^+ = F^+$
Hence, the decomposition is dependency preserving and also lossless.

* Concept of Normalization:-
The Normalization process was first proposed by Dr. CODD, as a guide to obtaining a good relational database design, based on the functional dependencies that are exhibited by the data in the database.



→ Normal Form

PAGE No.
DATE / /

Types of Database:-
i) Updatable database
ii) Read only database

* Advantages:-
1) Eliminate modification anomalies.
2) Reduce duplicated data
3) Eliminate data integrity problems
4) Save file space

* Disadvantages:-
1. More complicated SQL required for multi-table sub-queries and joins.
2. Extra work for DBMS can mean slower applications.

1) First Normal Form (1NF):-
This is the lowest level of normal form. A relation schema is said to be in 1NF, if the values in the domain of each attribute of the relation are atomatic. every attribute has only atomic values in it, i.e, values which cannot be further decomposed or divided. Another requirment for first normal form is that all attributes in a relation should have only single value. i.e, multi-valued attributes are not allowed in a relation that conforms to 1NF.

2) Second Normal Form (2NF):-
The 2NF is based on the concept of full functional dependency. A functional dependency is between a set of attributes to a set of attributes. The set have single attribute or more than one attribute.
A relation schema R(S,F) is in Second Normal form (2NF) if it is in the 1NF and if all non prime attributes are fully functionally dependent on the relation key (S).
$X \rightarrow Y$

PAGE NO
DATE / /

* Partial Dependency:-

Given a relation schema R with the FD's F, defined on the attributes of R and k as a candidate key. if X is a proper subset of K and if,
$F \Rightarrow X \rightarrow A$, then A is said to be partially dependent on k.

3) Third Normal Form (3NF):-

Third Normal Form is baised on the concept of transitive dependency.

A relation is in Third Normal Form (3NF) if and only if it is in 2NF and there are no transitive dependencies.

A transitive dependency comes up when we have $X \rightarrow Y$, $Y \rightarrow Z$, thus implying $X \rightarrow Z$.

A relation schema is in 3NF with respect to a set F of FDs if, for all FD's in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ & $\beta \subseteq R$, at least one of the following holds:-

i) $\alpha \rightarrow \beta$, is trivial FD

ii) α is superkey for R

iii) Each attribute A in $\beta \rightarrow \alpha$ is contained in a candidate key for R

4) Boyce-Codd Normal Form (BCNF):-

Boyce-codd Normal Form is a simpler Form of 3NF but it is found to be stricter than 3NF, meaning that every relation in BCNF is also in 3NF, but, a relation in 3NF is not necessary in BCNF.

A relation schema is in BCNF with respect to a set F of Functional Dependencies, if for all FD's in F^+, of the form $\alpha \rightarrow \beta$ where $\alpha \subset R$, $\beta \subseteq R$, at least one of the following holds

i) $\alpha \rightarrow \beta$

ii) α is a super key for R.

PAGE No.
DATE / /

- A relation is in Boyce-codd normal form if every attribute or set of attributes on which some other attribute is fully functionally dependent is also candidate for primary key of the relation.

* key-Concepts:-

* Candidate key:-
The minimal set of attributes that can uniquely identify each tuple in a relation is called as a candidate key.
for ex,
each tuple of EMPLOYEE relation given can be uniquely identified by E-ID and it is minimal as well. So it will be a Candidate key of the relation. A relation can have more than one candidate keys. The candidate key that is selected as per the current business context, to unique indentify each touple, then becomes the primary key for the relation with respect to the current business context.

* Super key:-
A super key is a set of one or more attribute (columns) which can uniquely identify a tuple in a relation. Candidate keys are minimal subsets of super keys, and hence also termed as a minimal super key.
For ex, each tuple of Employee relation can be uniquely indentified by set of all attributes, i.e, (E_ID, E_NAME, E_CITY, E_STATE).

The minimal set of attributes whose attribute closure is set of all attributes of relation is called Candidate key of relation.

* Prime Attribute and Non-prime Attribute:-

An attribute A in a relation schema R is a prime attribute or simply prime if A is a part of any candidate key of the relation.
If A is not a part of any candidate key of R, A is called a non-prime attribute or simply non-prime

* An Algorithm for Finding the candidate key / primary key for a Relation R.

A candidate key for a relation R is defined as a minimal superkey. In the following algorithm, we start by setting the key denoted by K to the set of all attributes of R, and then we remove one attribute at a time and cheak whether the remaining attributes still form a superkey.

The algorithm given below determines only one key for R; the returned depends on the order in which attributes are removed from R in step 2 of the algorithm.
The algorithm as follows:-
i) set K := R;
ii) For each attribute A in K.