Functional Dependencies (FD)

- A functional dependency (FD) is a constraint between two sets of attributes. It is denoted by $x \rightarrow y$ (read as "Y is functionally dependent on X").

The left-hand side of the FD is some times called as the <u>determinants</u> and the right-hand side is called <u>dependent</u>.

eg.

SSN → Name

Pnumber → {Pname, Plocation}

<u>Functional Dependency set:</u> Functional dependency set or FD set of a relation is the set of all FDs present in the relation

<u>Attribute closure:</u> Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.

Trivial & Non trivial Functional dependency

if a functional dependency $x \to y$ holds true where Y is not a subset of X then this dependency is called <u>non-trivial</u> functional dependency.

eg: Following functional dependencies are <u>non trivial</u>.
employee (empid, emp_name, emp_adress)

- emp_id → emp_name
  emp_id → emb_addres

Following functional dependencies are <u>trivial</u>.

{emp_id, emp_name} → emp_name,
[empname is a subset of {emb_id, embname}

The following dependencies are also trivial

- $A \to A$ & $B \to B$,

Armstrong's Axioms.

1. **Reflexivity**: if $X \supseteq Y$ then $X \to Y$
(if X is a superset of Y or Y is a subset of X)

2. **Augmentation**: if $X \to Y$, then $XZ \to YZ$.

3. **Transitivity**: if $X \to Y$ and $Y \to Z$
then $X \to Z$.

(4) **Decomposition**: If $X \to YZ$, then $X \to Y$ and
$X \to Z$.

(5) **Union**: If $X \to Y$ and $X \to Z$ then $X \to YZ$

(6) **Pseudo-transitivity**: if $\alpha \to \beta$ holds and $\gamma\beta \to \delta$
holds then $\alpha\gamma \to \delta$ holds.

eg. Suppose a relation R(A,B,C,D,E,F) with a
set of FDs as shown below:

$A \to BC$, $B \to E$, $CD \to EF$

Show that the FD, $AD \to F$ holds for R. & is a
member of the closure.

(1) $A \to BC$ & $CD \to EF$ (given)
(2) $A \to B$ & $A \to C$ (Decomposition)
(3) $AD \to CD$ (Augmentation)
(4) $AD \to CD$ & $CD \to EF$
$AD \to EF$ (Transitivity)
(5) $AD \to E$ & $AD \to F$ (Decomposition).

Lecture [illegible]

Normalisation.

Normal forms —

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce - Codd Normal form)
- 4NF (Fourth Normal form)
- 5NF (Fifth Normal form).

Normalisation: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.

Normal form: condition using keys & FDs of a relation to certify whether a relation schema is in a particular normal form.

1NF

- Disallows composite attributes, multivalued attributes, and nested relations; attributes whose values for an individual tuple are non-atomic.

2NF

A relation schema R is in second normal form (2NF) if every non-prime attribute A in R is fully functionally dependent on the primary key.

3NF A relation is in Third Normal Form if it is in 2NF and non-primary key attributes must be non-transitively dependent upon primary key attributes.

In other words a relation is in 3NF if it is in 2NF and having no transitive dependency.

Boyce codd Normal Form (BCNF).

BCNF is a strict format of 3NF.
A relation is in BCNF if and only if all determinants are candidate keys.
BCNF deals with multiple candidate keys.

Fourth Normal form (4NF)

A relation is in 4NF if it is in BCNF and for all multi Valued Functional Dependencies (MVD) of the form $x \twoheadrightarrow y$

Project Join Normal Form (5NF) and Join Dependency:

Let R is a relation and D is a set of all dependencies. The relation R is in 5NF wrt. D if for every Join Dependency, join dependency is trivial.

5NF is the ultimate Normal form. A relation in 5NF is guaranteed to be free of anomalies.

Decomposition.

The basic idea in decomposition is to split a relation into smaller relation schemas.

we address the problem of redundancy to a large extent.

we must ensure that when a relation is decomposed the integrity constraints are maintained.

- **Lossless decomposition:–** A relation R is said to be a lossless decomposition into R1 & R2 iff. the natural join of these two relations gives back the original relation R.

Lossy decomposition:– The natural join of R1 & R2 does not provide the original relation R, then it is said to be a lossy decomposition.

- **Spurious tuples:** The effect of lossy decomposition is that when R1 & R2 are joined, some extra tuples will creep in. These extra tuples are called as spurious tuples.

Closure of a set of FD's

The set of all FD's that are implied by a given set F of FD's is called the closure of F and is denoted by F^+.

example

Let $F = \{ AB \rightarrow C,\ C \rightarrow B \}$ be a set of FD's satisfied by $R(A, B, C)$. Then

$$F^+ = \{ A \rightarrow A,\ B \rightarrow B,\ AB \rightarrow AC,\ AB \rightarrow BC,\ AB \rightarrow ABC,\ \text{etc.} \}.$$

Attribute closure F^+

Armstrong's Axioms do not produce any incorrect FD's that are added to F^+. However finding F^+ is too expensive; the complexity grows exponentially. The solution is to find the attribute closure of X denoted as X^+.

Algorithm

Attribute closure()

{

$X^+ = X$

Repeat {

for each FD $A \rightarrow B$ in F do

if $A \subseteq X^+$ then $X^+ \cup B$ // i.e. if A is in X^+,

then add B to X^+.

Until no change

// Until no more attributes are added to X^+.

}

}

MVD and JD.

Multivalued Dependencies : Multivalued dependency (MVD) $x \to\to y$ read as "there is a multivalued dependency of Y on X". or X multidetermines Y",

eg. course_no $\to\to$ time room
course_no $\to\to$ student

teacher is teaching a particular course irrespective of the time or room it is held.

Join dependency: Lossless join condition is one of the most important criteria for a good database design.

Let $S = \{R_1, R_2 \ldots R_k\}$ be a set of relation schemes over $R = R_1 \ldots R_k$. A relation r over R satisfies the join dependency called (JD), if r has a lossless join decomposition.

A multivalued dependency buy is therefore equivalent to a binary join dependency.