- A relational DB should be designed with min redundancy. Redundancy means storage of same data in multiple locations. It leads to wastage of storage space.

If we take an example Book_publisher relation

ISBN	Book_title	P_ID	Pname	phone
002-768-5	DBMS	P001	Bright publications	9876543210
003-978-6	C++	P001	"	9876543210

- Here, let us assume that 100 books are published by one publisher then, all the info related to that publisher will be repeated unnecessarily for 100 times.

- Insertion anomaly:
  - In the above relation, we can't insert a new publisher if he/she has not published any book.
  - we can't insert new book, if we don't know the publisher

- Deletion anomaly: It leads to a situation in which deletion of (2) data representing certain info resulting in losing data representing some other info that is associated with it.

- Modification anomaly:
  - Repeated data changed at one place results in inconsistency unless the same data is also changed at other places

Relational Database Design

* Undesirable properties in Relational Database Design:

- more redundant data
- more null values in tuples.

- Combined schema without repetetion:

Consider 2 relations,

① section_class (sec_id, building, room_number)

② section (course_id, sec_id, semester, year)

If we combine these two relations together,

section (course_id, sec_id, semester, year, building, room_number)

Here, we didn't get any repetetion of data.

- Decomposition:

- The only way to avoid repetetion of info is decomposition.
- But not all decompositions are good.

For Eg; If we take Employee (ID, name, street, city, salary)

If we decompose this into 2 relations:

Employee1 (ID, name)

Employee2 (name, street, city, salary)

Here, If 2 employees have the same name.

ID	name	street	city	salary
1100	ABC	main	Hyd	75000
1500	ABC	North	Secunderabad	85000

ID	Name
1100	ABC
1500	ABC

name	street	city	salary
ABC	main	Hyd	75000
ABC	North	Secunderabad	85000

↓ natural join ↙ (we will get 4 records instead of 2)

Example:

Emp_info (Emp_id, name, age, dept_id, dept_name)

- Emp details: emp_id, name, Age
- dept details: Dept_id, dept name

We can't join these two. Because we don't have one common attribute

we will not get the original α table — Lossy decomposition.

- lossless decomposition In previous eg; decompose into emp details (emp_id, name, age) dept details (dept_id, emp_id, dept_name)

- Let R be the relation schema & Let R_1, R_2 form a decomposition of R

  ie., $R = R_1 \cup R_2$

- We can say that decomposition is lossless decomposition. It there is no loss of info by replacing R with 2 relational schemas $R_1 \cup R_2$.

$$\Pi_{R_1}(\gamma) \bowtie \Pi_{R_2}(\gamma) = \gamma$$

Decomposition is lossy if, $\gamma \subset \Pi_{R_1}(\gamma) \bowtie \Pi_{R_2}(\gamma) = \gamma$

Eg: Decomposition of $R = (A, B, C)$ to $R_1 = (A, B)$, $R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

γ

A	B
α	1
β	2

$\Pi_{A,B}(\gamma)$

B	C
1	A
2	B

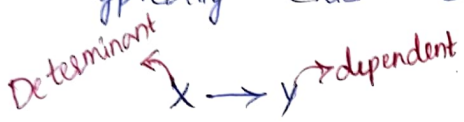
$\Pi_{B,C}(\gamma)$

→ $\Pi_A(\gamma) \bowtie \Pi_B(\gamma)$

A	B	C
α	1	A
β	2	B

* Functional dependency:

" " is a relationship that exist b/w 2 attributes. It typically exists b/w PK & non key attribute with in a table.



Left side of FD is determinant

Right side is dependent.

Eg: Assume that we have Employee table with attributes:

Emp_id, Emp_name, Emp_address

Emp_id → Emp_name

Here, Emp_name is functionally dependent on Emp_id.

* types of FD:

(i). Trivial Functional Dependency:

- A → B has trivial fund dependency if B is subset of A.

Eg: Consider Employee table with 2 columns Emp_id, Emp_name.

A → A, B → B also trivial.

{Emp_id, Emp_name} → {Emp_id} is trivial FD as

Emp_id is subset of {Emp_id, Emp_name}

Also, Emp_id → Emp_id

Emp_name → Emp_name } trivial FD.

(ii). Non trivial Functional Dependency:

- $A \rightarrow B$ has non trivial dual dependency if B is not a subset of A.
- When $A \cap B$ is null, then $A \rightarrow B$ is complete non trivial.

Eg: Emp_id → Emp_name

Emp_name is not subset of emp_id.

(iii) Multivalued FD:

- Entities of the dependent set are not dependent on each other

Eg: student table with rno, name, age columns.

rno → {name, age}

Here, name & age are not dependent on each other

(iv). Transitive FD:

Dependent is indirectly dependent on determinant

If $a \rightarrow b$ & $b \rightarrow c$ then $a \rightarrow c$

Eg: enroll_no, name, dept, building_no

Here, enroll_no → dept

dept → building_no then, enroll_no → building_no

* Closure of a set of FD:

A closure is a set of FDs is a set of all possible FDs that can be derived from a given set of FDs It is called as complete set of FDs

If F is used to donate set of FDs for relation R, then closure set is denoted by F^+.

Let's consider the set F of dual dependencies given below.

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

From F, It is possible to derive following dependencies. (3)

$A \to A$: Self determination

$A \to B$: Given in F

$A \to C$: Transitivity

$A \to D$: "

If we apply Union to all, $A^+ \to ABCD$ i.e., $A \to ABCD$.

→ The closure of FD is the complete set of all possible attributes that can be functionally derived from given FD using inference rules.

Steps to calculate closure of FD:

1. Add the attributes which are present on left hand side in the original FD.
2. " " " Right hand side of FD.
3. With the help of attributes present on RHS, check the other attributes that can be derived from the other given FD.

Repeat this process until all the attributes which can be derived are added in the closure.

Eg: R(A, B, C, D, E) Find closure of Set of FD for FD1 : $A \to BC$
FD2 : $C \to B$
FD3 : $D \to E$
FD4 : $E \to D$

$A^+ = \{A, B, C\}$, $C^+ = \{B, C\}$

$B^+ = \{B\}$, , $D^+ = \{D, E\}$, $\{E^+\} = \{E\}$

* Closure of Set of attributes:

- Closure of an attribute can be defined as a set of attributes that can be functionally determined from it.
- closure of a set of attributes X concerning F is the set X+ of all attributes that are functionally determined by X.

Eg: Relational schema R(P Q R S T U V) having following attribute

FD: $P \to Q$, $QR \to ST$, $PTV \to V$ Determine $(QR)^+$ & $(PR)^+$

Sol: $(QR)^+ = QR = QRST$; $(PR)^+ = PR = PQRST$

Eg: R(P Q R S T), FD: {$P \to QR$, $RS \to T$, $Q \to S$, $T \to P$} $\Rightarrow (T)^+$

Sol: $(T)^+ = TP = TPQR = TPQRS$

Q: $F: \{E \to A, E \to D, A \to C, A \to D, AE \to F, AG \to K\}$

Then find E^+?

Sol: $E^+ = \{E, A, D, C, F\}$

Q: $F: \{B \to C, BC \to AD, D \to E, CF \to B\}$. Find B^+?

Sol: $B^+ = \{B, C, A, D, E\}$

Q: $F: \{A \to B, B \to C, C \to D, A \to E\}$ Find closure of F.

Sol:

$A^+ = \{A, B, C, D, E\}$

$B^+ = \{B, C, D\}$

$C^+ = \{C, D\}$

$F^+ = \{A \to A, A \to B, A \to C, A \to D, A \to E, B \to B, B \to C, B \to D,$

$C \to C, C \to D\}$

* Finding candidate key & super key in a relation:

* Given relation R & set of FD's F that hold on R, find all CK's for R.

$R(A, B, C, D, E, F)$; $F = \{AB \to C, AC \to B, AD \to E, BC \to A, E \to F\}$

We need to follow few steps:

1. Let γ = set of attributes not present in RHS of any FD.
   The set of attributes in γ must be a part of any CK of R.
2. Let β = set of attributes appear on RHS but not on LHS of any FD. Set of attributes in β can't be part of CK of R.
3. Find closure γ^+. If $\gamma^+ = R \Rightarrow \gamma$ is CK.
4. If $\gamma^+ \neq R$, then each attribute x in $R - \beta$
   (i). Test whether $\gamma \cup \{x\}$ is a CK
   (ii) If not, add another attribute from $R - \beta$ to γ & test whether it is a CK.
   (iii). Repeat this step until all CK's found.

Sol:

1: In RHS side, we have A, B, C, E, F. we don't have D.

$\gamma = \{D\}$

2. Attributes that appear on RHS but not on LHS

In RHS, A, B, C, E, F. In which F didn't appear in LHS.

$\beta = \{F\}$

3. Find $\gamma^+ = D^+ = \{D\} \neq R$.

4. $R - \beta = \{A, B, C, D, E, F\} - \{F\} = \{A, B, C, D, E\}$

Now $\gamma \cup x \Rightarrow$ $(AD)^+ = \{A, D, E, F\} \neq R$ $(CD)^+ = \{C, D\} \neq R$

$(BD)^+ = \{B, D\} \neq R$ $(ED)^+ = \{E, D, F\} \neq R$

Let us add one more attribute,

$(ABD)^+ = \{A, B, D, C, E, F\} = R$ — CK

$(ACD)^+ = \{A, C, D, B, E, F\} = R$ — CK

$(AED)^+ = \{A, E, D, F\} \neq R$

$(BCD)^+ = \{B, C, D, A, E, F\} = R$ — CK

$(BED)^+ = \{B, E, D, F\} \neq R$

$(CDE)^+ = \{C, D, E, F\} \neq R$

So, (ABD), ACD, BCD are Candidate Key of R

$(ABCD)^+ = \{ABCDEF\} = R$ It is super key but not CK

Because, its subset consists of keys.

* prime attributes:

whatever the attributes are forming the candidate key those are prime attributes. i.e., part of CK.

* Non prime attributes:

The attributes which are not forming the CK. Those are non prime attributes.

Eg: $R(A, B, C, D, E, F)$; $FD: \{A \to B, B \to C, C \to D, D \to E\}$

Sol: (i). $\delta = \{A\}$

(ii). $R = \{E\}$

(iii). $A^+ = \{A, B, C, D, E\} \neq R$

$(AF)^+ = \{A, B, C, D, E, F\} = R$ — CK.

prime attributes = $\{A, F\}$

Non prime attributes = $\{B, C, D, E\}$

* Irreducible Functional Dependency Set / Canonical Covers / Minimal Covers:

If F is a functional dependency then F' is irreducible FD if it satisfies (i) F' should not have extraneous / redundant attributes

(ii) " " " " Redundant FD.

To get irreducible set of functional dependency,

Step1: Apply splitting rule at Right Hand Side (dependent side)

i.e, RHS side it should have single attribute.

Step2: Remove extraneous attributes (: trivial attributes directly neglect)

Step3: Remove Redundant FD by finding closure sets.

Eg: $F : \{AB \to C,\ C \to AB,\ B \to C,\ ABC \to AC,\ A \to C,\ AC \to B\}$

Sol: Step1: $\{AB \to C,\ C \to A,\ C \to B,\ B \to C,\ ABC \to A,\ ABC \to C, A \to C,\ AC \to B\}$

Step2: Here $ABC \to A$, $ABC \to C$ } trivial attributes. So directly discard them

Now, $\{AB \to C,\ C \to A,\ C \to B,\ B \to C,\ A \to C,\ AC \to B\}$

In this, $AB \to C$

also $B \to C$ {: C alone depends on B}

So, we can neglect (A) $B \to C$

$AC \to B$, $C \to B$ {: B alone depends on C}

So, we can neglect (A) $AC \to B$

So, Now we will get, $\{C \to A,\ C \to B,\ B \to C,\ A \to C,\ \cancel{B \to C},\ \cancel{C \to B}\}$

Step3: $C \to A$ ✓ $C^+ = \{C, A, B\}$ $\quad$ $C \to B$ ✗ $C^+ = \{C, B, A\}$ $\quad$ $B \to C$ ✓ $B^+ = \{B\}$ $\quad$ $A \to C$ $A^+ = \{A\}$

$F' = \{C \to A,\ B \to C,\ A \to C\}$ It may not be unique same

Eg: $F = \{x \to w,\ wz \to xy,\ y \to wxz\}$ Find Canonical Covers

Sol: Step 1: $\{x \to w,\ wz \to x,\ wz \to y,\ y \to w,\ y \to x,\ y \to z\}$

Step 2: we don't have any trivial or any redundant attributes

so, we need to find closures accordingly remove redundant FD

$x \to w$ ✓
$x^+ = \{x\}$

$wz \to x$ ✗
$(wz)^+ = \{w, z, x, y\}$
If we remove also
$(wz)^+ = \{w, z, y, x\}$

$wz \to y$ ✓
$(wz)^+ = \{w, z\}$

$y \to w$ ✗
$y = \{w, x, y, z\}$
If we remove $y \to w$
$y = \{y, x, w, z\}$

$y \to x$ ✓
$y = \{y, z\}$

$y \to z$ ✓
$y = \{y, x, w\}$

So, Final Canonical cover is, $\{x \to w,\ wz \to y,\ y \to x,\ y \to z\}$
$y \to xz$

Eg: $F = \{B \to A,\ AD \to BC,\ C \to ABD\}$ Find Canonical cover.

Sol: Step 1: $B \to A,\ AD \to B,\ AD \to C,\ C \to A,\ C \to B,\ C \to D$

$B^+ = \{B, A\}$

$(AD)^+ = \{A, D, B, C\}$

$C^+ = \{C, A, B, D\}$

~~$B \to A$~~ $\Rightarrow B^+ = \{B\}$

✗ $AD \to B \Rightarrow (AD)^+ = \{A, D, C, B\}$

~~$AD \to C$~~ $\Rightarrow (AD)^+ = \{A, D, B\}$

✗ $C \to A \Rightarrow C^+ = \{C, B, D, A\}$

✓ $C \to B \Rightarrow C^+ = \{C, D\}$

✓ $C \to D \Rightarrow C^+ = \{C, B, A\}$

So, Canonical cover is $\{B \to A,\ AD \to C,\ C \to BD\}$

Eg: $\{w \to x,\ y \to x,\ z \to wxy,\ wy \to z\}$ Find Canonical Covers?

Sol: $F = \{w \to x,\ y \to x,\ z \to wxy,\ wy \to z\}$

* Normalization:

» is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, updation anomalies. Normal forms are used to eliminate or reduce redundancy in DB tables.

(i). First Normal Form:

- If a relation contains composite or multivalued attribute, it violates 1st NF. If a relation doesn't contain composite or multivalued attribute that is in 1 NF.
- A relation is in 1 NF if every attribute in that relation is single valued attribute.

Eg: Student

rno	name	phone_no	state	country
1	Raama	9876543210, 8765432109	AP	India
2	Sita	9999999999	TS	India

Here, we have multivalued attribute. ie., phone_no. Now, we need to convert it into 1NF.

rno	name	phone_no	state	country
1	Raama	9876543210	AP	India
1	Raama	8765432109	AP	India
2	Sita	9999999999	TS	India

(ii). Second Normal Form:

- A relation must be in 1 NF.
- A relation must not contain any partial dependencies. ie., no non prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

* Second NF:

1 NF doesn't remove/eliminate redundancy, but it eliminates repeating groups

- A relation is in 2NF,

  (i). If the relation is in 1NF.

  (ii). It should not contain any partial dependency.

  i.e., No partial dependency (No non prime attribute is dependent on any proper subset of any Candidate Key of the table).

  (or) Proper subset of CK → non prime attribute

  Every non primary key attribute is fully functionally dependent on primary key.

- In 2NF, we will remove partial dependencies in 1NF.

Eg: Teacher (key attribute)

Teacher_id	subject	age
123	English, maths	30
246	Computers	25

This table is not in 1st NF.

1NF →

Teacher_id	subject	age
123	English	30
123	maths	30
246	computers	25

This is in 1NF.

In this table Non Key attribute ie, age is depending on teacher_id. which is proper subset of Candidate Key. To, convert this into 2NF, we need to compose into 2 tables.

① Teacher_details

teacher_id	age
123	30
246	25

② teacher_subject

teacher_id	subject
123	English
123	maths
246	computers

Eg: Employee (Emp_id, Emp_name, project_id, project_name) (Key attributes)

(i). emp_details (emp_id, project_id, Emp_name)

(ii). project_details (project_id, project_name)

* 2NF Example:

Student (std_no, course_no, course_fee);

std_no	course_no	course_fee
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

Here, we have C_1, C_3 & same fee of 1000/- . C_4, C_5 has 2000/-

So, course fee can't decide course_no or std_no

course_no + course fee also can't decide std_no.

Candidate Key (std_no, course_no)

Table1 (std_no, course_no) ⟶

std_no	course_no
1	C1
2	C2
1	C4
4	C3
4	C1
2	C5

Table2 (course_no, course_fee)

course_no	course_fee
C1	1000
C2	1500
C3	1000
C4	2000
C5	2000

If we have 1000 students also, course table will have only 5 records.

* R(A,B,C,D) ; FD: {AB→CD , C→A, D→B}. Convert into 2NF?

Sol: (i) $\gamma = \{\emptyset\}$, (ii) $\beta = \{\emptyset\}$

$A^+ = \{A\} \neq R$; $C^+ = \{C, A\} \neq R$

$B^+ = \{B\} \neq R$; $D^+ = \{D, B\} \neq R$

✓ $(AB)^+ = \{A, B, C, D\} = R$ — CK

$(AC)^+ = \{A, C\} \neq R$

✓ $(AD)^+ = \{A, D, B, C\} = R$ — CK

✓ $(BC)^+ = \{B, C, A, D\} = R$ — CK

$(BD)^+ = \{B, D\} \neq R$

✓ $(CD)^+ = \{C, D, A, B\} = R$ — CK

AB, AD, BC, CD are candidate keys

Here, prime attributes are {A, B, C, D}. we don't have any non prime attributes. So It is in 2NF

* R(A,B,C,D) ; FD: {A→B, B→C, C→D} Convert into 2NF?

Sol: $\gamma = \{A\}$, $\beta = \{D\}$

$\gamma^+ = A^+ = \{A, B, C, D\} = R$ ✓

A is CK. So, ~~prime attributes are {A,B,C,D}~~. It is 2NF

- If all CK is having single attribute, then, Relation will be in 2NF

(Ex:)

* R (A,B,C,D) ; FD: {A→B, B→C} Convert into 2NF?

Sol: $\gamma = (A, D)$, $\beta = \{C\}$

$A^+ = \{A, B, C\}$, $D^+ = \{D\}$

✓ $(AD)^+ = \{A, B, C, D\} = R$ — CK

Prime Attribute → NPA

A, D → B, C

A→B — Partial dependency

Now Find closure of A ie., $A^+ = \{A, B, C\}$

Split R into $R_1(A,B,C)$, $R_2(A,D)$

F_1^+ : ?

$A^+ = \{A,B,C\} = \{B,C\}$

$B^+ = \{B,C\} = \{C\}$

$C^+ = \{C\}$

Not required — $(AB)^+ = \{A,B,C\} = \{C\}$

$(BC)^+ = \{B,C\}$

Not required — $(AC)^+ = \{A,C,B\} = \{B\}$

$F_1^+ = \{A \to BC, B \to C\}$

AB, AC not required. Because $A \to BC$ FD we have.

$A^+ = \{A,B,C\} \longrightarrow CK$.

$B^+ = \{B,C\}$

$C^+ = \{C\}$

A is prime attribute & single so no partial dependency.

It is in 2NF now.

F_2^+ : ?

$A^+ = \{A,B,C\} = \{A\} = \emptyset$

$D^+ = \{D\} = \emptyset$

$(AD)^+ = \{A,D,B,C\} = \{A,D\} \neq \emptyset$

$F_2^+ = \{\}$

* $R(A,B,C,D)$; FD: $\{A \to B, C \to D\}$

Sol:

$\alpha = \{A,C\}$, $\beta = \{B,D\}$

$A^+ = \{A,B\}$

$C^+ = \{C,D\}$

$(AC)^+ = \{A,C,B,D\} = R$ — CK

PA $\longrightarrow$ NPA

$(A,C) \to (B,D)$

$A \to B$, $C \to D$ are partial dependencies. we need to decompose.

$R_1(A,B)$, $R_2(C,D)$, $R_3\{A,C\}$

$F_1^+ \cup F_2^+ \cup F_3^+ = F$

$F_1^+ = \{A \to B\}$

$A^+ = \{A,B\}$

$B^+ = \{B\}$

$(AB)^+ = \{A,B\}$

$F_2^+ = \{C \to D\}$

$C^+ = \{C,D\}$

$D^+ = \{D\}$

$(CD)^+ = \{C,D\}$

$F_3^+ = \{\}$

$(A)^+ = \{A,B\} = \{B\} \neq R_3$

$(C)^+ = \{C,D\} = \{D\} \neq R_3$

$(AC)^+ = \{A,C,B,D\} = \{B,D\} \neq R_3$

* 3NF:

- A relation will be in 3NF if it is in 2NF & not contain any transitive partial dependency
- 3NF is used to reduce the data duplication. & also to achieve data integrity.
- If there is no transitive dependency for non prime attributes, then the relation must be in 3NF.
- A relation is in 3NF if it holds atleast one of the following conditions for every non-trivial functional dependency $X \rightarrow Y$.
  - (i) X is a super key
  - (ii) Y is a prime attribute i.e, each element of Y is part of some candidate key

Eg: Employee_details table

(Emp_id, Emp_name, Emp_zip, Emp_state, Emp_city)

Here, Emp_id is the prime (or) key attribute

Super Key : {Emp_id, Emp_name}, {Emp_id, Emp_name, Emp_zip}, {Emp_id}....

Non Key (or) Non prime attributes: are Emp_name, Emp_zip, Emp_state, Emp_city

Here, Emp_state and Emp_city dependent on Emp_zip ie., Emp_zip → Emp_state, city

Emp_zip dependent on Emp_id i.e., Emp_id → Emp_zip

→ {Emp_state, Emp_city} transitively dependent on {Emp_id}.

We need to convert this table into 2 tables.

(i). Employee (Emp_id, Emp_name, Emp_zip)

(ii) Emp_zip (Emp_zip, Emp_state, Emp_city)

Eg: Student (rollno, game, fee)

→ student_details (rollno, game)

→ game_details (game, fee_structure)

* 3rd NF:

(i). It should be in 2NF

(ii) No transitive dependency for non prime attributes

NPA → NPA

- A table is in 3NF; if and only if for each of the non-trivial functional dependency atleast one of the following conditions holds.

(i). LHS is super key

(ii). RHS is prime attribute.

Eg: R(A, B, C, D) ; FD: {A→B, B→C, C→D}

Sol: $X = \{A\}$, $B = \{D\}$

$X^+ = A^+ = \{A, B, C, D\} = R$ ✓ — CK

PA → NPA

{A} → {B, C, D}

A is not proper subset of CK A. So, A→B is not Partial dependency

PA → NPA

B→C, C→D. These are having following

NPA NPA NPA NPA

B→C — $B^+ = \{B, C, D\} \neq R$ — not super key

RHS is C is not prime attribute

C→D — $C^+ = \{C, D\} \neq R$ — not SK

RHS is D is not prime attribute.

So, B→C, C→D are not in 3NF. (Those are having transitive depen

Now, $B^+ = \{B, C, D\}$ — $R_1(B, C, D)$

$C^+ = \{C, D\}$, $R_2 = (CD)$

$R_3 = \{A, B\}$ SK among all B, C, D

remaining attributes

$R_1 = \{B, C, D\}$

$B^+ = \{B, C, D\} = R$

$C^+ = \{C, D\}$

$D^+ = \{D\}$

$(BC)^+ = \{B, C, D\}$

$(BD)^+ = \{B, D, C\}$

$(CD)^+ = \{C, D\}$

$A^+ = \{A, B, C, D\} = \{B, C, D\}$

$B^+ = \{B, C, D\} = \{C, D\}$

$(AB)^+ = \{A, B, C, D\} = \{C, D\}$

$C^+ = \{C, D\}$

$D^+ = \{D\}$

$(CD)^+ = \{C, D\}$

F^+: {B→CD, C→D, ~~BC→D~~, ~~BD→C~~ A→B} :

$A^+ = \{A, B, C, D\} \rightarrow CK$

$B^+ = \{B, C, D\}$

$C^+ = \{C, D\}$

$D^+ = \{D\}$

A is prime attribute & single CK. NPA are $\{B, C, D\}$

$B \rightarrow CD$ $\quad C \rightarrow D$ $\quad A \rightarrow B$

$C \rightarrow D$ ↓ TD

CK ~~SK~~ $B^+ = \{B, C, D\}$

$C^+ = \{C, D\}$

$D^+ = \{D\}$

PA → NPA

$\{B\}$ $\{CD\}$

$C \rightarrow D$ is in transitive dependency so ~~$B \rightarrow CD$~~ we need to decompose

$R_1(B, C, D)$ on $F: \{B \rightarrow CD, C \rightarrow D\}$

$C^+ = \{C, D\}$

~~$D^+ = \{D\}$~~

$R_{11}(C, D)$, $R_{12}(B, C)$

Now check for dependency preserving or not

$C^+ = \{C, D\}$ $\quad B^+ = \{B, C, D\}$

$D^+ = \{D\}$ $\quad C^+ = \{C, D\}$

$(CD)^+ = \{C, D\}$

$F^+ = \{C \rightarrow D$, $B \rightarrow CD\}$

Now again check these 2 are not having ~~trivial~~ transitive dependency

$C^+ = \{C, D\}$ $\quad B^+ = \{B, C, D\} = R$ — CK ✓

$D^+ = \{D\}$

PA → NPA

$\{B\} \rightarrow \{CD\}$

We have 2 $C \rightarrow D$ FD's so we can neglect one $C \rightarrow D$ in R_{11}

$R_{12}(B, C)$, $R_2(C, D)$, $R_3(A, B)$ FD: $\{A \rightarrow B, C \rightarrow D, B \rightarrow C\}$

* Dependency Preservation:

- In " " , atleast one decomposed table must satisfy every dependency.
- If a relation R is decomposed into R_1 & R_2, then the dependencies of R either must be a part of R_1 or R_2 or must be derivable from the combination of FD's of R_1 & R_2.

Eg: Relation R(A, B, C, D) with FD set $A \rightarrow BC$.

If R is decomposed into $R_1(ABC)$, $R_2(AD)$. which is holding dependency preserving because FD: $A \rightarrow BC$ is part of $R_1(ABC)$.

Eg: R(A,B,CD) & FD: {AB→C, C→D, D→A}. R is decomposed into $R_1(A, B, C)$ & $R_2(C, D)$. Check whether decomposition is dependency preserving or not.

Sol: Let us find closure of F_1 & F_2

To find F_1^+, find all combinations of ABC i.e., A, B, C, AB, BC, AC.

$A^+ = \{A\}$, $B^+ = \{B\}$, $C^+ = \{C, D, A\}$

$= \{C, A\} \rightarrow D$ is not present in R_1

$C \rightarrow AC \because \Rightarrow C \rightarrow A$ { : remove 'C' as it is trivial}

$\{AB\}^+ = \{A, B, C, D\} = \{A, B, C\}$

$AB \rightarrow ABC \Rightarrow AB \rightarrow C$

$\{BC\}^+ = \{B, C, D, A\} = \{A, B, C\}$

$BC \rightarrow ABC \Rightarrow BC \rightarrow A$

$\{AC\}^+ = \{A, C, D\}$

$AC \rightarrow ACD \Rightarrow AC \rightarrow D$

$F: A \rightarrow C,\ B \rightarrow DE,\ D \rightarrow C$

CK ? $\{A, B\}$ = key attributes

(12)

$F_1^+ = \{ C \rightarrow A,\ AB \rightarrow C,\ BC \rightarrow A \}$

=

Find F_2^+ ie., C, D, CD.

$C^+ = \{C, D, A\}$

$C \rightarrow CDA \Rightarrow C \rightarrow CD$

$C \rightarrow D$

$D^+ = \{D, A\}$

$D \rightarrow D$

$F_2^+ = \{C \rightarrow D\}$

= In the given relation FD is $\{AB \rightarrow C,\ C \rightarrow D,\ D \rightarrow A\}$

$AB \rightarrow C$ is present in F_1^+

$C \rightarrow D$ " " " F_2^+

$D \rightarrow A$ is not preserved

$F_1 \cup F_2$ is subset of F. So, given decomposition is not dependency preserving.

Rules for dependency preservation:

A decomposition $D = \{R_1, R_2, R_3 \ldots R_n\}$ of R is dependency preserving with respect to a set F of FD if,

$(F_1 \cup F_2 \cup \ldots \cup F_m)^+ = F^+$.

Consider a relation R, R is decomposed into R_1 with $\{F_1\}$ & R_2 with $\{F_2\}$.

(i) $F_1 \cup F_2 = F \rightarrow$ Decomposition is dependency preserving.

(ii) $F_1 \cup F_2$ is subset of F $\rightarrow$ not dependency preserving

(iii) $F_1 \cup F_2$ is superset of F $\Rightarrow$ NOT possible case.

* BCNF (Boyce codd normal form):

(i). It should be in 3NF

(ii). For each non-trivial FD $X \to Y$, X must be a super key

Eg: R(A,B,C) ; FD={ $A \to B$, $B \to C$, $C \to A$ } Is it in BCNF or not?

Sol:

$A^+ = \{A, B, C\} = R$ — CK

$B^+ = \{B, C, A\} = R$ — CK

$C^+ = \{C, A, B\} = R$ — CK

CK = {A,B,C} = Prime attributes

There are no partial dependencies & no transitive dependencies.

$A \to B$, A is super key

$B \to C$, B is SK

$C \to A$, C is SK

So, given relation is in BCNF.

Eg: R (A,B,C,D,E); FD = { $A \to BCDE$, $BC \to ACE$, $D \to E$ }

Sol:

$\emptyset = \{\}$, $B = \{E\}$

$A^+ = \{A, B, C, D, E\} = R$ ✓ — CK

$B^+ = \{B\}$

$C^+ = \{C\}$

$D^+ = \{D, E\}$

$E^+ = \{E\}$

PA $\to$ NPA

(A) $\to$ (B,C,D,E) — no partial dependencies

$(BC)^+ = \{B, C, A, E, D\} = R$ — SK

PA $\to$ NPA

BC $\to$ ADE

So, $BC \to ACE$ — no transitive dependency

$D \to E$ — Here, we have transitive dependency.

So, given relation's highest normal form is 2 NF

* Normalize a relation from 1NF to BCNF:

Eg: $R(A, B, C, D, E, F, G, H)$ $FD := \{A \to BD, B \to C, E \to FG, AE \to H\}$

Sol: $\alpha = \{A, E\}$, $\beta = \{C, D, F, G, H\}$

$\gamma^+ = A^+ = \{A, B, D, C\}$

$E^+ = \{E, F, G\}$

$(AE)^+ = \{A, E, B, D, F, G, C, H\} = R$ — CK

P A $\longrightarrow$ NPA

$(A, E) \longrightarrow \{B, C, D, F, G, H\}$

2nd NF: no P.D

$A \to BD$ = PD

$B \to C$ ✗

$E \to FG$ = PD

$AE \to H$ ~~PD~~ ✗ not proper sub of CK ie., AE.

we found 2 PD's we need to decompose,

$A^+ = \{A, B, D, C\}$

$R_1 = \{A, B, C, D\}$

$F_1^+ = \{A \to BCD, B \to C\}$

$A^+ = \{A, B, C, D\}$

$B^+ = \{B, C\}$

$C^+ = \{C\}$

$D^+ = \{D\}$

CK = A → no PD

PA = A

NPA = BCD

3rd NF: $B \to C$ is TD

$A \to BCD$ is not TD

$E^+ = \{E, F, G\}$

$R_2 = \{E, F, G\}$

$F_2^+ = \{E \to FG\}$

$E^+ = \{E, F, G\}$

$F^+ = \{F\}$

$G^+ = \{G\}$

CK = E → No PD

PA → NPA

E → FG

No PD

3rd NF ✓

E is superkey

So, BCNF ✓

$R_3 = \{H, A, E\}$

$F_3^+ = \{AE \to H\}$

$H^+ = \{H\}$

$A^+ = \{A, B, C, D\}$

$E^+ = \{E, F, G\}$

$(AH)^+ = \{A, H, B, D, C\}$

$(AE)^+ = \{A, E, B, D, F, G, C, H\}$

$(HE)^+ = \{H, E, F, G\}$

CK = AE

PA → NPA

AE → H 2 NF ✓

AE is SK. 3NF ✓

So BCNF ✓

R_1 (A, B, C, D) should be decomposed

$F: \{A \to BD, B \to C\}$

$A^+ = \{A, B, C, D\}$

$B^+ = \{B, C\}$

$R_{11}(B, C)$, $R_{12}(A, D, B)$

$B^+ = \{B, C\}$

$C^+ = \{C\}$

R_{11} (B, C)

$B^+ = \{B, C\}$

$C^+ = \{C\}$

$(BC)^+ = \{B, C\}$

$F: \{B \to C\}$

B is CK & PA & SK

3NF ✓

BCNF ✓

$R_{12}(A, D, B)$

$A^+ = \{A, B, D, C\}$

$B^+ = \{B, C\}$

$D^+ = \{D\}$

$F: \{A \to BD\}$

A is CK & PA & SK

3NF ✓

BCNF ✓

$R_2 = \{E, F, G\}$, $R_3 = \{A, E, H\}$, $R_{11} = \{B, C\}$, $R_{12} = \{A, D, B\}$

* Comparision of BCNF and 3NF:

3NF

(i). There should be no transitive dependencies.

(ii) Less Stronger than BCNF

(iii) FD's are already in 1NF & 2NF

(iv) Redundancy is high

(v) There is preservation of all FD's

(vi) lossless decomposition can be achieved easily

BCNF

(i). For any relation $A \rightarrow B$, A should be superkey of relation.

(ii). more stronges than 3NF

(iii). FD's are already in 1NF, 2NF & 3NF

(iv). Redundancy is comparitively low in BCNF

(v) may or may not be preservation of all FD's

(vi) lossless decomposition can be achieved hardly.

* DBMS - Indexing:

Search Key	Data reference value

Data stored in the form of records. Every record has a key field which helps it to be recognized uniquely.

- Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done.

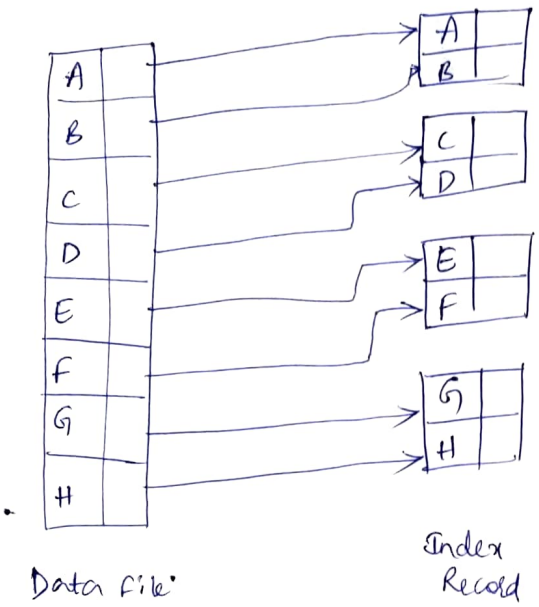
(1) primary Index:

It is defined on an ordered data file. The data file ordered on a key field. The key field is generally the PK of the relation.

- If the index is created on the basis of PK of the table, then it is known as primary Indexing.
- As PK's are stored in sorted order, the performance of the searching operation is quite efficient.
- Primary index can be classified as Dense & Sparse Index.

(i). Dense Index:

- " " contains an index record for every search key value in the data file. It makes searching faster.
- The no of records in the index table is same as the no of records in the main table.
- It needs more space to store index record itself. Index records have the search key & a pointer to the actual record on the disk.
- For every search key value in the data file, there is an index record.

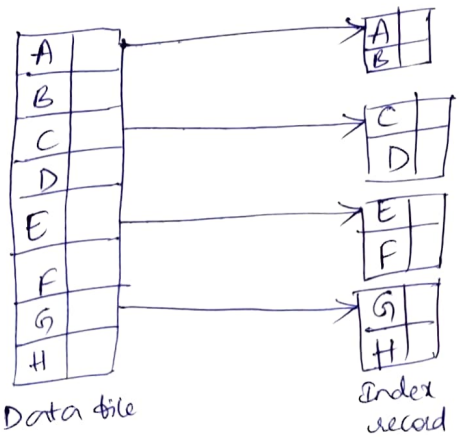


For every search value in Data file, there is an index record

15

(ii) Sparse Index:

- The index record appears only for a few items in the data file. Each item points to a block.
- Instead of pointing to each record in the main table, the index points to the records in the main table in a gap.

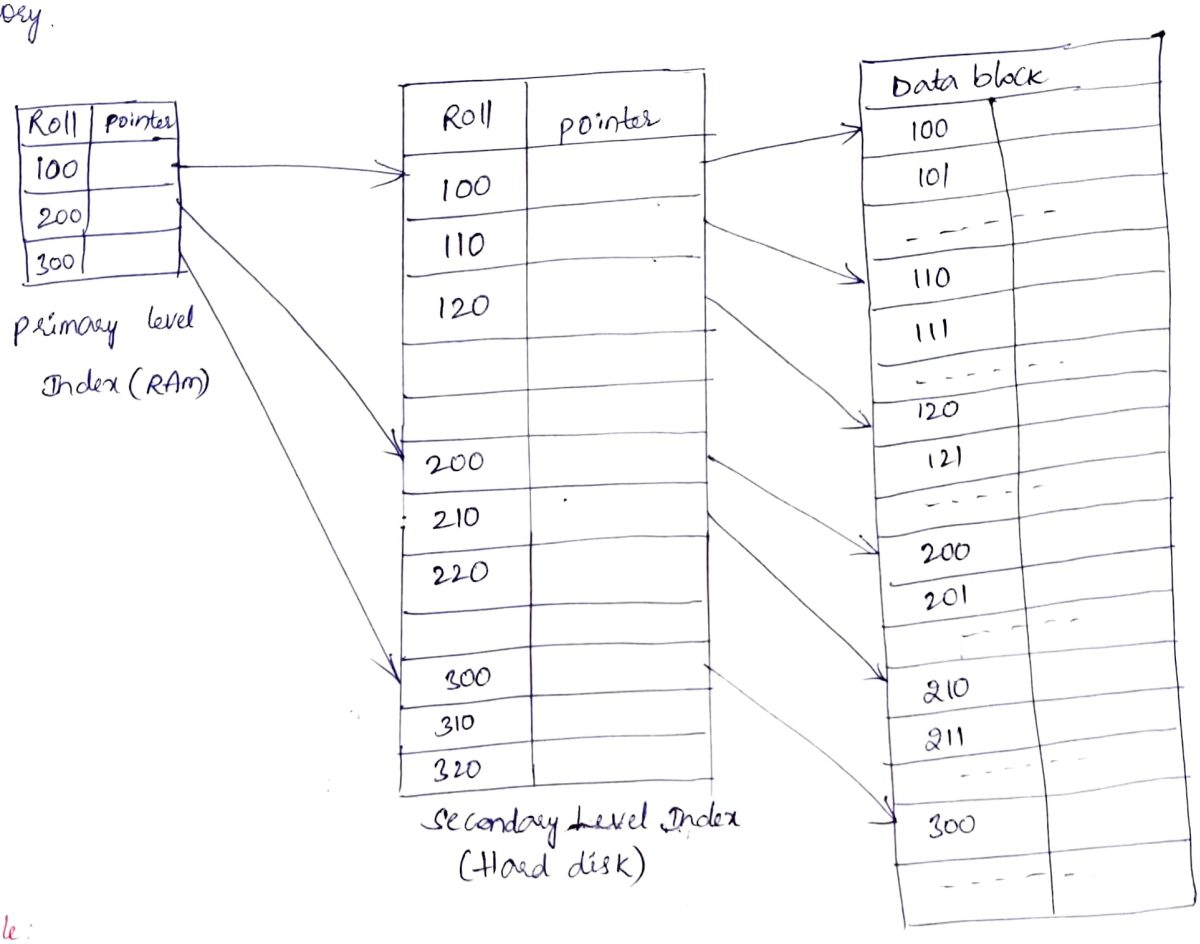


For very few search value in Data file, there is an index record.

* Secondary Index:

- " " is created for each record in a data file which is CK.
- " " is a type of dense index & also called as non clustering index.
- In sparse indexing, as table size grows, the size of mapping also grows. These mappings are usually kept in primary memory so that faster access. Secondary address memory searches actual data based on the address got from mapping. If mapping size grows, fetching address becomes slower.

In Secondary indexing, to reduce the size of mapping. The huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of first level is stored in primary memory, fetching address is faster. The mapping of second level & actual data stored in secondary memory.



Example:

- If we want to find the roll 111, then it will search the highest entry which is smaller than or equal to 111 in the first level index. It will get 100 at this level. In the second level again max(111) <= 111. It goes to 110. Now, It will go to data block & starts searching each record till it gets 111.
- In the same way insertion, deletion & updation will be done.

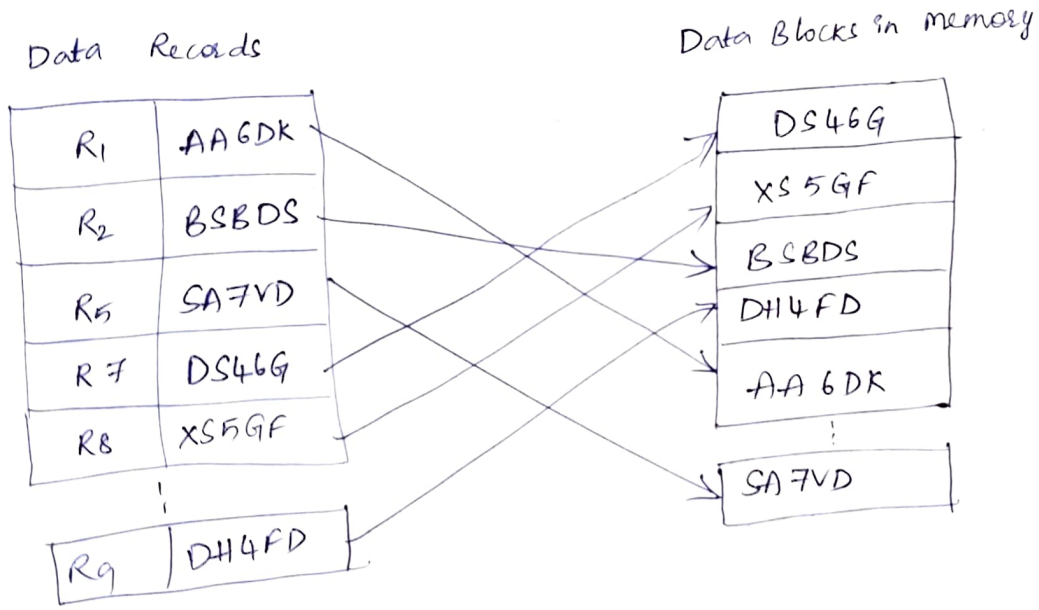
* Tree structured Indexing:

" " " techniques support both range & equality searches.

- ISAM is static, B^+ tree is dynamic.
- As for any index, 3 alternatives for data entries K^*:
  - Data record with key value K
  - K, rid of data record with search key value K
  - K, list of rids of data " " "

(i). ISAM (Indexed Sequential Access method):

" is an advanced sequential file organization. In this method, records are stored in the file using PK. An index value is generated for each PK & mapped with the record.



Advantages:

- each record has the address of its data block, searching a record in a huge DB is quick & easy
- It supports range retrieval & partial retrieval of records since the index is based on PK, we can retrieve the data for given range of va partial value can also be easily searched. ie., data record address starting with 'SA' can be easily searched.

Disadvantages:

- It requires extra space in the disk to store the index value.
- when the new records are inserted, then these files have to be reconstruct to maintain the sequence.
- when the record is deleted, then the space used by it needs to be releas Otherwise, performance of DB will slow down.

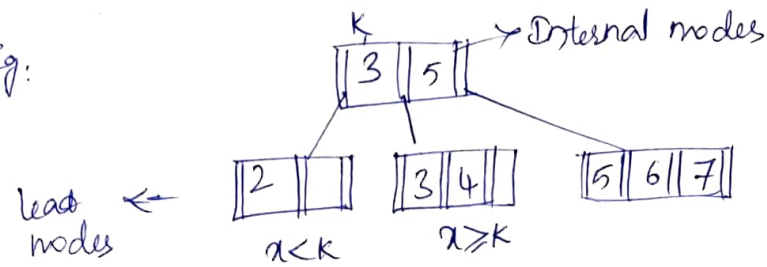
(ii). B^+-Tree Index File:

- B^+-tree index is a multi level index.
- " " is a rooted tree satisfying below properties:
  - (i) All paths from root to leaf are equally long.
  - (ii). Each node isn't root or a leaf, it has b/w $\lceil n/2 \rceil$ and 'n' children.
  - (iii). A leaf node has between $(n-1)/2$ & $(n-1)$ values.
- Structure of any node of the tree is:

P_1	K_1	P_2	---	P_{n-1}	K_{n-1}	P_n

- In B^+ tree, leaf nodes denotes actual data pointers
- All leaf nodes remains at same height.
- the leaf nodes are linked using linked list.
- B+ tree occupy a little more space than B-tree
- Internal nodes stores just keys.

Eg:



Eg. 2, 4, 7, 10, 17, 21, 28 ; n = 3