Database

Database

1. Database
2. Database Management System
3. Flat File System
4. Relational File System (R. J Codd)
   - (i) Data must store in tabular Form.
     - (a) Attributes (Col. Name)
     - (b) Records or Tuple (Row of table)
     - (c) Arity (No of column)
     - (d) Cardinality (No of row)
     - (e) Domain (set of possible values that a attribute can accept)
     - (f) Relational Schema (Table Structure) (Details of table)
     - (g) Relational Instance (Data of table) (set of records)
   - (ii) No two records of table should be same. Two records must have atleast one different value.
   - (a) Candidate Key: Minimal attribute (or minimal set of attribute) which can differentiate record uniquely. (Minimal Super key)
     - * 'X' is candidate key of R if and only if:
       - * No proper subset of 'X' can differentiate record unique.
       - * No two records whose 'X' values are same.
   - (b) Primary key: Any CDK, whose field Must Not "NULL" and Must Distinct.
   - (c) Alternate key: In it field value can be "NULL".
   - (d) Simple Candidate Key
   - (e) Compound Candidate key
   - (f) Prime Attribute or Key Attribute
   - (g) Super Key (may or may not be minimal) (set of attribute)
   - (h) Foreign Key or Referential Key
   - (i) Foreign Key Integrity Constraint
     - → Deletion (Referenced relation)
     - → Updation (Referenced & Referencing relation)
     - → Insertion (Referencing Relation)

(5) Normalization of DB table. → Remove → Updation Anomaly
→ Deletion Anomaly
→ Insertion Anomaly

(6) Functional Dependency
(i) Trivial FD
(ii) Non Trivial FD
(iii) Semi-Non Trivial FD
(iv) Amstrong Rule → (a) Reflexive ($X \to X$)
(b) Transitivity ($X \to Y$, $Y \to Z$)
(c) Augmentation ($X \to Z$)
$X \to Y$
then $XZ \to YZ$
(d) Split and Merge Rule.

(7) Attribute Closure: (main idea is to obtain Minimal SK).

(8) Membership Test of FD's (one FD derive from other)

(9) Equality of FD set ($F \equiv G$) ↔ ($F \supseteq G$ && $G \supseteq F$)

$F \subseteq G$	$G \subseteq F$	
Yes	Yes	$F \equiv G$
NO	Yes	$F \supset G$
Yes	NO	$F \subset G$
NO	NO	incomparable

(10) Canonical cover or Minimal cover
(i) Minimal cover is logical equivalent to FD set.
(ii) Minimal cover may not be unique. (More than one MC possible because of Extraneous)
(iii) Rules to find Minimal Cover: (a) Split the FD's
(b) Find & delete Redundant FD
(c) Find & delete Redundant FD on LHS. (Extraneous Att)
ex: $AB \to C$
if A^+ contain B, then delete B.
if B^+ contain A, then delete A.

(11) Decomposition
(i) Loss Less Decomposition
$R_1 \bowtie R_2 \bowtie R_3 \bowtie \dots \bowtie R_n \supseteq R$
(a) Loss less
$R_1 \bowtie R_2 \bowtie \dots \bowtie R_N = R$
(b) Lossy
$R_1 \bowtie R_2 \bowtie \dots \bowtie R_N \supset R$
(Due to superfluous record)
(b) Rule $R_i \cap R_j = R_i$ or R_j
If all i & j merge to 1 then it is Lossless JOIN
@ Rules: (a) $R_1 \cup R_2 \equiv R$
(b) $R_1 \cap R_2 = R_1$ or $R_1 \cap R_2 = R_2$
} Set of attributes → Common attribute must be candidate key of either R_1 or R_2
R_1 R_2 R_3
R_1

(11) Dependency preserving Decomposition

(a) $F_1 \cup F_2 \cup F_3 \ldots \cup F_n \subseteq F$

(b) $F_1 \cup F_2 \cup \ldots \cup F_n \equiv F$
then it is dependency preserving decomposition

(c) $F_1 \cup F_2 \cup \ldots \cup F_n \subset F$
then it is non depending preserving.

Note: (i) $R_1, R_2, \ldots R_N$ is for attributes
$F_1, F_2, \ldots F_N$ is for Functional Dependency.
(ii) Loss Less is for values of data base.
(iii) Dependency is for functional dependency.

(12) Normal Form: It is rule to identify degree of redundancy

(13) Types of Normal Form

(i) First Normal Form: Each row must be atomic
(b) NO MVD.
(c) Redundency Present If
* $X \to Y$
* $X \to Y$ is Non trivial FD and X is not Superkey

(ii) Second Normal Form:
(a) Any Prime key can not determine any non prime key.
Means NO Partial Dependency.
(Proper Subset of candidate key) $\to$ (Non Prime Attribute) ⇒ Partial Dependency.

(iii) Third Normal Form: $X \to Y$ Non trivial FD then.
(a) X must be a Super key
or
(b) Y must be a prime key.

(iv) BCNF (Boyce Codd NF): $X \to Y$ non trivial FD then
'X' must be a Super key.

Note: To check 2NF, First Find all CK, then check Subset closer, if it derive non prime the it is Partial Depend.
For 2NF, First check given dependency is in 2NF or not if given, satisfy the find Subset closer to further verify.

(4) (13) Decomposition of relation in Higher NF

(a) maintain, closure in other table.

(b) Dependency Not always preserve in case of BCNF.

Ex: $R = \{A\ B\ C\}$

$FD = \{AB \to C,\ C \to A\}$

$CK = \{AB, BC\}$

It is in 3 NF but not in BCNF

R(BC)	R(CA)
BC → BC	C → A

(i) Loss less ✓

(ii) Dependency NOT preserving.

* Inner JOIN (only matching) common

* Natural Join
No common attribut : $n \times m$

R(ABC)	R(A)
BC → BC C → A AB → C	C → A

Not in BCNF

(14) Procedural Queries (Relational Algebra)

(15) Non Procedural Queries (Relational Calculus)

(16) Relational Algebra operators

(i) Projection (π) (ii) Selection (σ) (iii) Cross product (X) (iv) Rename (ρ)

(v) Union ($\cup$) (vi) Set Difference ($-$) (vii) Intersection ($\cap$) = {"$-$"}

(viii) Join ($\bowtie$) = {"X, σ, π"} (ix) Division ($\div$) : {"$\pi, X, -$"}

(17) Natural Join ($\bowtie$) : (a) $R \bowtie S \Rightarrow \pi\ (\sigma_{\text{condition}}(R \times S))$ (Default common attribute condition)

(b) Its result is same as cross product or Cartesian product if condition is not mention.

(c) $R_1 \to n$, $R_2 \to m$ tables:

$R_1 \times R_2 \to n \times m$

$R_1 \bowtie R_2 \to$ min: 0, max: $n \times m$

(18) Conditional Join ($\bowtie_c$) : (a) $R \bowtie_c S \Rightarrow \sigma_c(R \times S)$ (b) $R \bowtie_c S \Rightarrow$ min: 0, max: $n \times M$

(19) Left outer Join ($\bowtie$) : (a) $R \bowtie S \Rightarrow (R \bowtie S) \cup$ (Fail Records of R with NULL)

(b) $R \bowtie S \Rightarrow$ min: n, max: $n \times M$

(20) Right outer Join ($\bowtie$) : (a) $R \bowtie S \Rightarrow (R \bowtie S) \cup$ (Fail records of S with NULL)

(b) $R \bowtie S \Rightarrow$ min: m, max: $m \times n$

(21) Full outer Join ($\bowtie$) : (a) $(R \bowtie S) \cup$ [Fail record of Both R & S with NULL]

(b) $(R \bowtie_c S) \cup$ [Fail record of Both R & S]

(c) $R \bowtie S \Rightarrow$ min: max(m,n), max: $n \times M$

(23) Division Operation (÷)

min: 0
max: $\left\lfloor \frac{n}{m} \right\rfloor$

(a) $R_{A,B} / S_B \Rightarrow R_A$ (Every cases)

(b) $\pi_A(R) - \pi_A\left(\pi_{AB}(R_A \times S_B) - \pi_{A,B}(R_{AB})\right)$

① ② ③

① Each and every
② Not invalue in some
③ Invalue in everything

(c)

$$\pi_A(R) - \pi_A\left[\left[\begin{array}{l}\text{Make each and}\\ \text{every A is associated}\\ \text{with B.}\\ (R_A \times S_B)\end{array}\right] - \left[\begin{array}{l}\text{Original A}\\ \text{which is}\\ \text{Associated with}\\ \text{B. } \pi_{AB}(R)\end{array}\right]\right]$$

Those A which are not associated with B.

Those A which are associated with each and every B

(22) Union, intersection, Difference

Union compatable

(a) Arity of both relation must be same.
(b) Domain of each attribute of R must be same as domain of each attribute of S.

Rules of Union(∪) ≡ Rules for intersection(∩) ≡ Rules of set Diff(−)

(c) $R \cup S = \{ X \mid X \in R \vee X \in S \}$ (OR)

$R - S = \{ X \mid X \in R \wedge X \notin S \}$ (But NOT)

$R \cap S = \{ X \mid X \in R \wedge X \in S \}$ (AND)

$R \cap S = R - (R - S)$ (on)

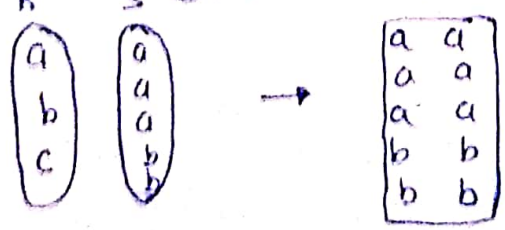
(d) $T_1 \cap T_2 \equiv T_1 \bowtie T_2$
(e) $T_1 \cup T_2 \equiv T_1 \bowtie T_2$

Difference is only condition for union & intersect

(24) Types of Mapping

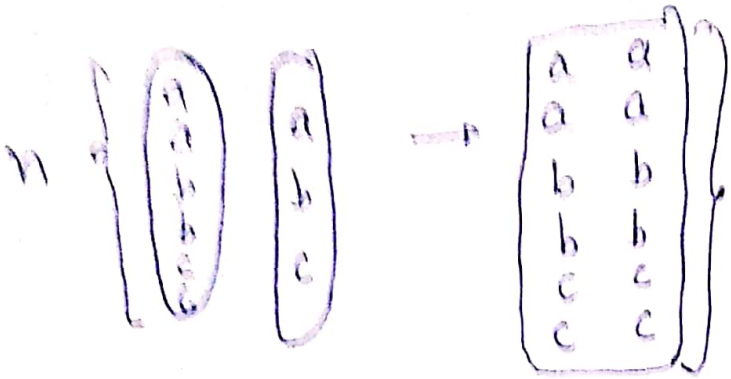
$R \to n$, $S \to m$

(a) One to Many: If common attribute is key for 1st Relation not for 2nd.
(b) Maximum possible pair: m



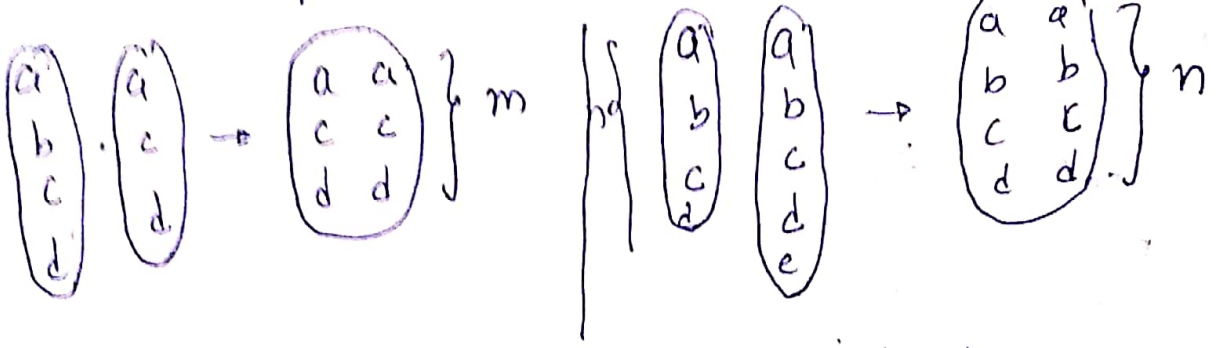
(ii) Many to One : If common attribute is key for second one not.

Maximum pair : n



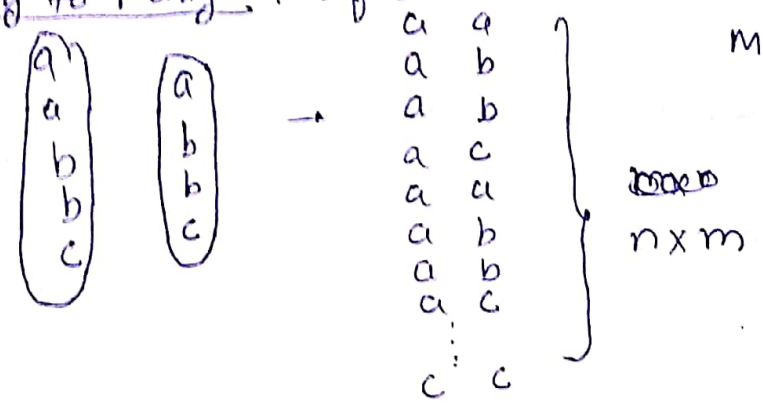
(iii) One to One : If common attribute key for both.

Maximum : min (m, n)



(iv) Many to Many : If common attribute is key for no one.

max : n x m



(25) Structured Query Language (SQL) : It is a programming lang to deal with data.

(26) Types of SQL lang

(i) Data Definition Lang (DDL) : It is used to define an modify schema Database table.

(a) CREATE (b) ALTER (c) DROP (d) TRUNC

(ii) Data Manipulation lang (DML) : It is use to modify, update and access data records

(a) INSERT (b) SELECT (c) UPDATE (d) DELETE

(iii) Data Control Lang : (DCL) : It is use to control user access to data.

(a) GRANT (b) REVOKE

(iv) Transaction Control Lang (TCL) : It is use to manage changes affecting data.

(a) COMMIT (b) ROLLBACK (c) SQL Data it be

Syntax of Clause :

SELECT (5) DISTINCT (6) $A_1, A_2, A_3, \ldots A_N$

FROM (1) $T_1, T_2, T_3, T_4, \ldots T_N$ } Cartesian Product

WHERE (2) (CONDITION)

GROUP BY (3) (Attribute)

HAVING (4) (CONDITION) // If is only used with Group By

ORDER BY (7) (Attribute) [ASC/DESC] ;

Precedence :

FROM → WHERE → GROUP BY → HAVING → SELECT → DISTINCT → ORDER BY

(25) **Aggregate Function** : If calculate aggregation of non NULL value.

(i) count (ii) SUM( ) (iii) AVG( ) (iv) MIN( ) (v) MAX( )

↓ all excluding null not include in counting total also

Exception : COUNT(*), count NULL because of ordering (Row id)

(24) **GROUP BY CLAUSE**

(a) Every attribute of group by clause must be in select clause.

(b) Not allow any other attribute in select clause other than group by.

(c) Aggregate fn allow in select and compute for each group

(1) GROUP BY ($A_1, A_2, \ldots A_N$)

(2) If also work on NULL.

(3) ~~Without Group by clause~~

(4) Aggregate function not allowed in where clause.

(26) **EXCEPT or MINUS** : If is equivalent to Relation Algebra Minus.

Eg: Select eid from EMP

EXCEPT

Select eid from EMP T_1, EMP T_2

where T_1.sal < T_2.sal

} Maximum salary of all

(31) SUB Query in from class treated as table, sub Query in where clause treated as Scalar value.

(8) **With Clause**: It is used to reuse result of ~~sub~~ query many times.

ex: Employee having second heighest Salary

$$Temp = \Pi_{eid,sal}(EMP \bowtie_{EMP.sal < T.sal} \rho(T, EMP)) \quad // \text{ not include } 1^{st} \text{ Max}$$

$$= \Pi_{eid}(\Pi_{eid}(Temp) - \Pi_{eid}(Temp \bowtie_{Temp.sal < T.sal} \rho(T, EMP)))$$

Now,

```
with Temp(eid, sal) as
(Select distinct eid from EMP T1, EMP T2
 where T1.sal < T2.sal)
Select Eid From Temp
EXCEPT
Select eid from Temp T1, Temp T2 where T1.sal < T2.sal
```

---

(32) Expressive power of Safe TRC ≡ Expressive power of Relational Algebra.

(33) Functions for nested Query. (a) IN/NOT IN (b) ANY (c) ALL (d) EXISTS (e) UNIQUE/NOT UNIQUE

---

(33) **File Organisation**

Database $\xrightarrow{\text{Collection of}}$ Files or table $\xrightarrow{\text{Collection of}}$ Blocks $\xrightarrow{\text{Collection of}}$ Records

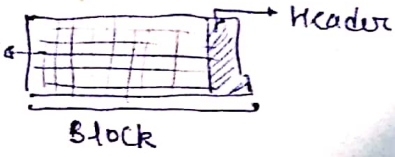
---

(34) **File Organization** : (a) Spanned org. (suffer from external Frag. only) * Prefer for variable rec. (b) Unspanned (suffer from both internal & External Frag) * Prefer for fixed length rec.

(35) **Block Factor** : (Maximum record accomodate in one Block)

(b) Block Size : B bytes, Record Size : R bytes, Header size : H

$$\text{Block Factor (unspanned)} = \left\lfloor \frac{B - H}{R} \right\rfloor$$



(36) **I/O Cost or Access Cost** : No. of secondary memory block required to transfer from disk to main memory. in order to access some record.

(b) Based on ordered field: $\lceil \log_2 N \rceil$

(c) Based on unordered field : N

Indexing : It is use to reduce access cost.

(b) Index records are always in order.

(c) Structure of Index file: | Search Key | Record Pointer |

(d) Block factor of index file :

Block Size : B
Size of search : K
Pointer size : P
Header Size : H

Maximum index per block $= \left\lfloor \frac{B-H}{K+P} \right\rfloor$ entries/Block

(e) In multilevel indexing, we go upto level only one Block is left.

---

(38) categories of Index

(i) Dense Index (Use by both Ordered & Unordered Record)

(ii) Sparse Index (Only Use by Ordered records)

* For dense Index, no. of records in database = no. of index record.

---

(39) Notes on Index :

Number of records in database : N
Record size of database : R bytes
Block Size = B bytes
Search key Size = K bytes
Record Pointer Size = P bytes.

Block factor of DB $= \left\lfloor \frac{B}{R} \right\rfloor$ rec/block

Total no. of block in DB $= \left\lceil \frac{N}{BF_{DB}} \right\rceil$ blocks

Index cost per Level
$\log_2 P$
Total cost $= (\log_2 P) + 1$

(40) Using Index :

(i) Dense Index

Block Factor of Index : $\left\lfloor \frac{B}{K+P} \right\rfloor$ index Rec/Block

Total No. of Block : $\left\lceil \frac{N}{BF_{Index}} \right\rceil$ Blocks (Dense)

(ii) Sparse Index

Block Factor of Index : $\left\lfloor \frac{B}{K+P} \right\rfloor$ index Rec / Block

Total No. of Block : $\left\lceil \frac{\text{Total No. of Block in } DB}{BF_{index}} \right\rceil$ Blocks (Sparse)

(10) (40) Types of index : (a) Primary index (b) Cluster index (c) Secondary

(a) Primary Index : If data base is ordered based on a key and same key is use for index then it is known as Primary Index. (Key is unique not repeated)

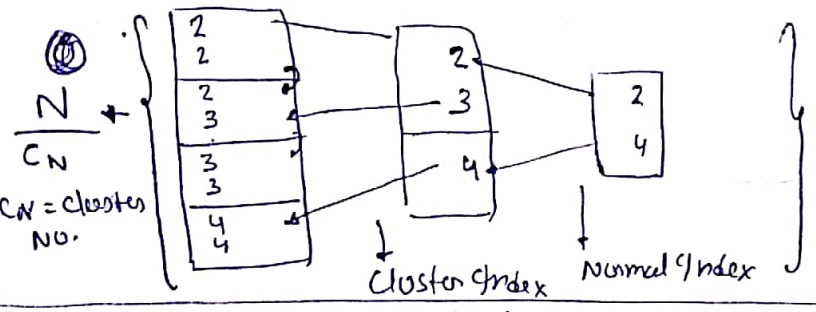
* Because of this we use Sparse index.
* Only one Primary index is possible in DB.

(b) Cluster Index : If database is order based on key but that key is not unique and have repeated data. That same key we use for index it form cluster index.

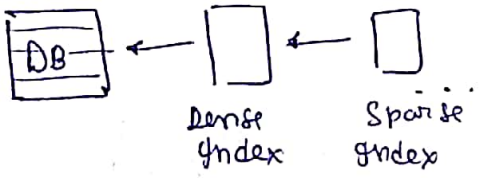
* Only one cluster index is possible.
* It is also sparse index
* In DB either Cluster or Primary index is possible not both.

I/O cost for Multilevel PI = (K + 1) Block

I/O cost for Mutilevel CI = (K+ 1 or more clusters) Block Block

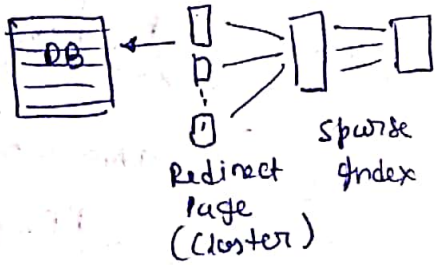


(c) Secondary Index : If database is unordered, but the key choose for indexing is unique. this type of index is secondary index with key.



I/O cost : $\frac{K+1}{\text{Level}}$

(d) Secondary index without key : If database is unordered, but the key choose for indexing is not unique, This type of indexing is secondary index without key.



I/O cost : K + $\frac{\text{NO. of Link in one redirect Page}}{}$

Static Indexing

(11) (a) Static Multilevel Index : Index not allow to modify if database records are inserted or deleted.

(b) Overflow Page : It increases the access cost.

(12) Dynamic Multilevel Index : Index can modify if there is insertion or deletion in database.

(b) Rules : Index block must be 50% utilise, otherwise merge it to other index block if less then 50%.

(13) Balanced Search Tree (Rules makes tree balance) (Dynamic Index)

(i) AVL (ii) B-Tree (iii) B^+-Tree

Search time : $O(\log n)$ Search (Block Factor)

Note : Degree for B & B^+ tree is no. of keys present in one block.

(14) Why B or B^+ Tree Prefer over AVL ?

(a) DB file Store in DISK i.e in secondary Memory, so Index to DB file also Store in SM.

(b) In Case of AVL, One node store one key so to store, high number of index, high no. of nodes are going, so it increases the level of Index and I/O cost.

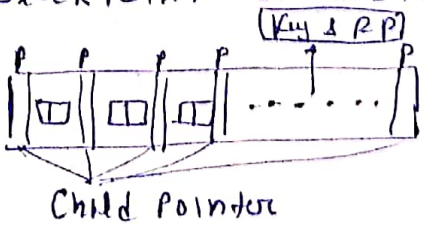
(c) In case of B & B^+ tree, One node store many keys so more index data store in One node of B & B^+ tree. So less no. of level & I/O cost.

(15) B Tree :

(i) Order P : It is maximum possible Block Pointer or Child Pointer in one Block.

Block Pointer : P

No. of keys and Record Pointer : (P-1)

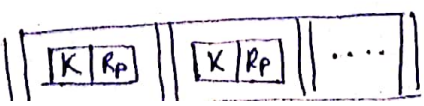


(ii) Rules for Balancing :

(a) All leaf node must at same level.

(b) Root Node
- Minimum
  - 1 Block
  - 2 Block Pointer & 1 key
- Maximum
  - 1 Block
  - P Block Pointer & (P-1) keys

(c) Internal Node
- Minimum
  - P Block
  - $\lceil P/2 \rceil$ Block Pointer & $\lceil P/2 \rceil - 1$ keys
- Maximum
  - P Block
  - P Block Pointer & (P-1) keys



Block Size : B bytes
Block Pointer : BP bytes
Key Size : K bytes
Record Pointer : R bytes

$$BP * P + (P-1)(K+R) \leq B$$

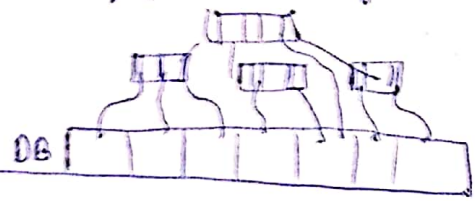
Order : P (Maximum Child Pointer)

(46) I/O Cost For B Tree

Normal Query : $(K+1)$ Blocks, where K is level of B Tree

Range Query : $(X \cdot (K+1))$, where X is range, K is level of B Tree

Note : Unorder data : Dense B Tree Index
Order data : Sparse B Tree Index

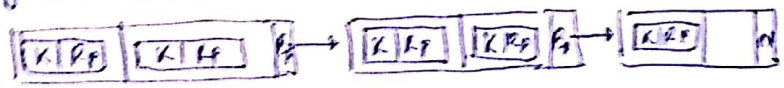


(47) B^+ Tree :

(i) Order P : Maximum possible pointer in One Block.

(ii) Types of Node :

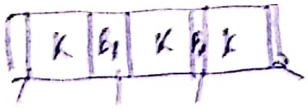
(a) Leaf Node : All keys along with Record Pointer resides here. and only one Block pointer present which point to next block



(P-1) Keys & RP
1 BP

* Left Biasing
* Right Biasing

(b) Internal Node : It only store keys and Block Pointer as child Pointer



P Block Pointer
(P-1) Keys

(iii) Balancing condition is same as B Tree.

I/O Cost	
N :	K+1
RQ :	K+X

(iv) Order for leaf

$$B_P + (P-1)(K+R) \leq B$$

Order for Internal Node

$$P * B_P + (P-1)K \leq B$$

(48) B^+ Tree Prefer over B Tree

(i) I/O Cost of B^+ Tree is less than B Tree in both Normal & Range Query.

(ii) If Block Size allocated For B Tree & B^+ Tree is same :

(a) Order P of B Tree < Order P of B^+ Tree.

(b) No. of Nodes in B Tree ≥ No. of Nodes in B^+ Tree

(c) No. of Levels in B Tree ≥ No. of Level in B^+ Tree

(49) Structure of table For Finding max/min Nodes or Key

Level	Max/Min No. of Nodes	Max/Min Block Pointer	Max/Min Key

For B^+ Tree = Last Level Key only
For B Tree = Summation of all level key

Root	Other

Bulk Loading : (B+ Tree)

(a) It is bottom top approach of designing a B+ Tree

(b) It is used when we have to design tree with given keys only. We have to find Min & Max level along with node.

(c) If to find Minimum level with Order P & K Keys

Leaf : Minimum Node : $\left\lceil \frac{\#K}{(P-1)} \right\rceil$ node

Internal Level : $\left\lceil \frac{\text{Leaf}}{P} \right\rceil$ node

Internal Level : $\left\lceil \frac{\text{Internal Level}}{P} \right\rceil$ node

minimum Level

2^{nd} Level :

1^{st} Level : 1 node

Full Factor Most satisfy

(d) To find Maximum Level with Order P

- Minimum Fill Factor = $\left\lceil \frac{P}{2} \right\rceil$ pointer

Leaf : $\left\lfloor \frac{\#K}{(\lceil P/2 \rceil - 1)} \right\rfloor$

Internal Nod : $\left\lfloor \frac{\text{Leaf}}{\lceil P/2 \rceil} \right\rfloor$

(51) Join Algorithm

(i) Nested Loop Join

X Blocks have → n record

Y Blocks have → m record

I/O cost : $X + nY$ or $Y + mX$

(ii) Block Nested Loop Join

I/cost = $X + X * Y$ or $Y + X * Y$

(51) (52) Transaction : (Collection of Operation forms single logical un

(53) ACID Property : (A: Atomicity), (C: consistency), (I: Isolation), (D: Dura

(53) Problem with Concurrent Transaction:

(i) Lost Update Problem (W-W conflict) {write just after write}

(ii) Dirty Read Problem (W-R conflict) {read data from uncommitted write}

(iii) Unrepeatable Read Problem (R-W conflict) {Data not same before write & after w
or Non-repeatable " "

{ R(A) | W(A) | R(A) }
data: a data: b

R(A)	W(A)
R(A)	

(iv) Phantom Tuple (Value of is not same at time of read & after some time)
(R-W) (if we read again)

(v) Incorrect Summary Problem

(54) Schedule : Order in which operation of transaction are executed.

(b) Total schedule possible (for two transaction as example) :

T_1 : n operation
T_2 : m operation

Total = $\frac{(n+m)!}{n!\,m!}$

(c) Total serial schedule for N transaction = N!

(d) Total non-serial schedule = $\frac{(n+m)!}{n!\,m!} - N!$

(55) Types of Schedule :

(a) Serial Schedule . (no interleaving of operation) .

(b) Complete Schedule : (either commit or abort in last)

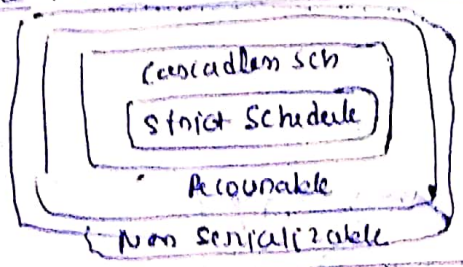
(c) Recoverable Schedule : (no commit before, Ti commit from which it reads value. uncommited read problem)

(56) Cascading Aborts or Rollback : (Don't read from uncommited write)

(57) Cascadless Schedule : (Read from commited write)

(58) Strict Schedule : (No read or write on data that is write by uncommited transaction)

(IMP)



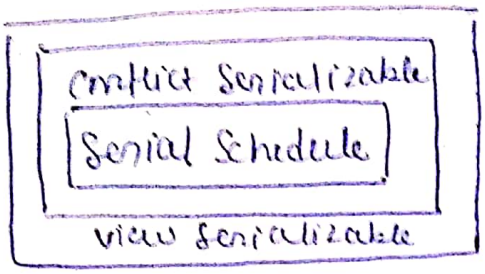
(59) Serializable Schedule: (If outcome equal to its serial schedule)

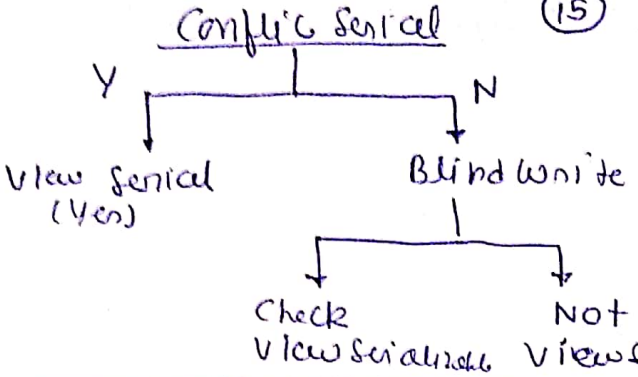
(i) Result equivalent Schedule: (Final results are same)

(ii) Conflict equivalent Schedule or Conflict Serializable (At least one write operation)
(Conflic Operation)
(order)
(same)

(iii) View equivalent or Uless Serializable.

No. of conflict schedule = No. of topological order





(61) Conditions for View Serializable :

(i) Final Update

(ii) Intial Read Final Update

(iii) Read just after write sequence

(62) LOCK : (A variable associated with data item to show its status).
(b) It is used for synchronization.

(63) Problems with Locks : (a) Deadlock (b) Starvation

(64) Purpose of Locks : (a) To resolve conflicts.
(b) To preserve DB consistency.
(c) To enforce isolation among conflicting trans.

(65) Types of Lock : (a) Binary Lock (b) Shared Lock (Read lock) (c) Exclusive Lock (Write lock)

(66) 2 Phase Locking : (a) Growing or Expanding Phase
(b) Shrinking Phase

Provide Serializability

(c) Basic 2 PL : Problem : Deadlock, not irrecoverable Free

(d) Conservative 2 PL : Problem : Deadlock, not Irrecoverable Free (aquire all Lock before start)

(e) Strict 2 PL : Exclusive locks held until Commit or Abort.
Problem : Deadlock, but recoverable.

(f) Rigorous 2PL : All locks release after Commit or Abort.
It is Deadlock Free & recoverable.

(67) Time Stamp Lock (Thomas write Rule) {Serializability & Deadlock free}
(a) Read Time stamp (RTS) (b) Write Time Stamp (WTS)

(c) Rule :
(i) If transaction wants to read, if WTS of that transaction is greater than current one then roll back. Otherwise set RTS of that transaction as $MAX(RTS_A, .TS)$.

(ii) If transaction wants to write, if RTS of that transaction is greater than current then roll back otherwise write allow and WTS = Current TS.

(16) (58) Entity : Real world object.

(59) Entity set : A group of similar entity.

(b) Strong entity set : An entity set that has a primary key.

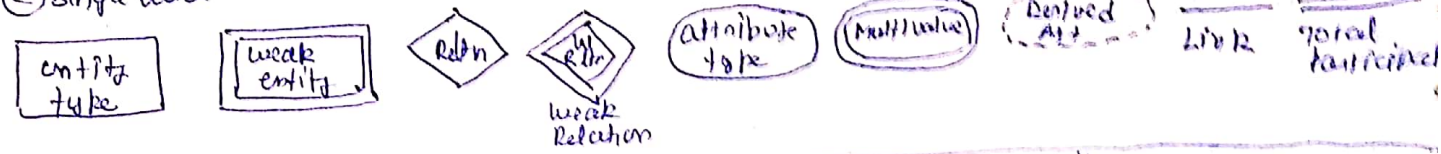
(c) Weak entity set : An entity set that not have sufficient Attribute for primary key.

(d) Owner entity set : It is used as identity of weak entity set.

(60) Attribute : Properties that describe entity.

(a) Composite Attribute (b) Derive Attribute (c) Descriptive Att (Associate with a Relationship)

(d) Single valued Att (e) Multivalued (f) Prime Attri



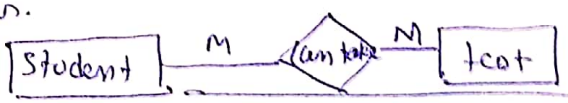
(71) Mapping Cardinalities / Cardinality Ratio / Types of Relationship

(i) One to One : An entity in A is associated with atmost one entity in B & vice versa

(→ or ←)

(ii) One to Many : An entity in A associated with 0 or more entity in B.

(iii) Many to Many : An entity in set A & B associated with one or more entity with each other.



(—)

(72) Candidate key & Relation Merge

(i) One to Many : Candidate key is from M side. and Relation Merge at M or Many to one side. (2 table)

(ii) Many to Many : Candidate key is from both table & it not merge on any side. (3 table)

(iii) One to One : Candidate key of any side & Merge to any side. (2 table)

(b) If one of relation is totally participated then merge all three table to one. (1 table)

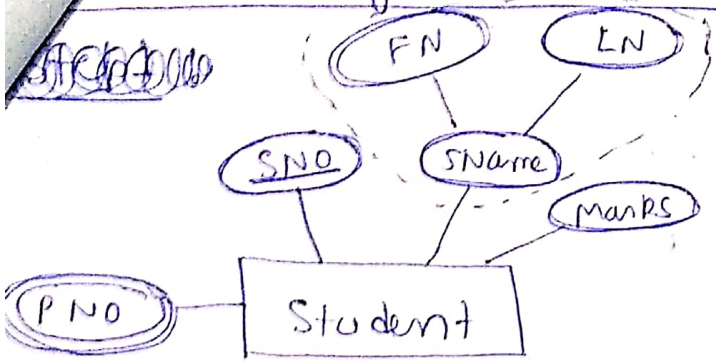
(73) For weak entity set total participation must be there.

Note :

(a) If Conflict Serializable, than 2PL, allocs. if not then may or may not definitely not allows.

(b) Multilevel Granuality : It is tree protocol, while locking lock from top to bottom while unlocking unlock from bottom to top.

Conversion of ER Model to Relational Database



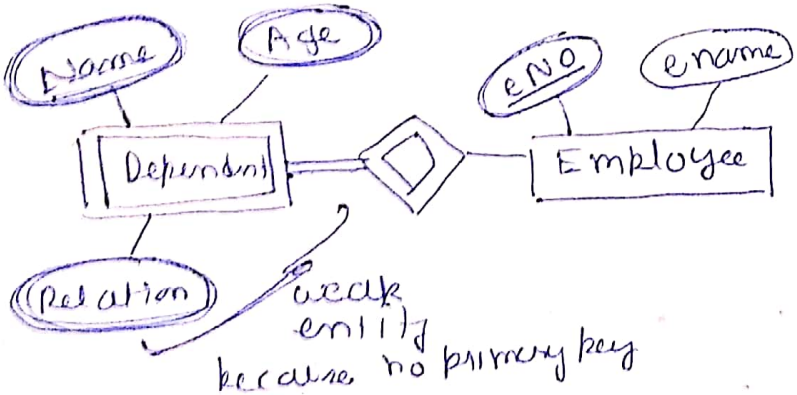
Step1 :

convert it into schema. with simple attribute.

Student → simple attribute along with composite attribute as simple attribute

SNO	Marks	FN	LN

Step2 : Convert weak entity to relation



(a) Here there is no primary key. So we use **Partial Key**, it is the key using which we can identify uniquely for **particular entity**.

(b) Let name as **partial key**.

Dependent

Name	Age	Relation

Employee

ENO	EName

↓ add full primary key of owner entity into weak.

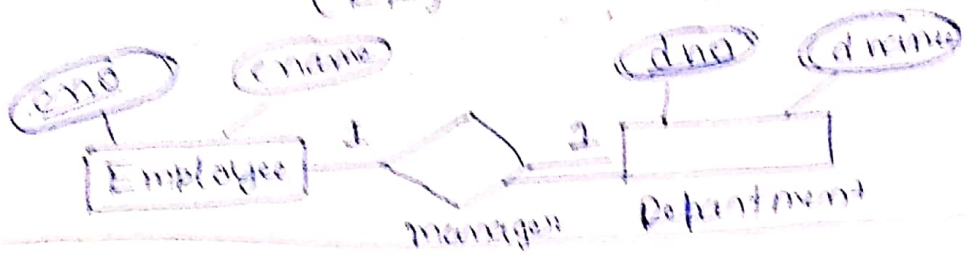
ENO	Name	Age	Relation

ENO	Ename

* When we delete owner entity, it is suppose to delete all entity of weak entity.

Step 3 : Convert Relationship to Relation (Table)

(i)



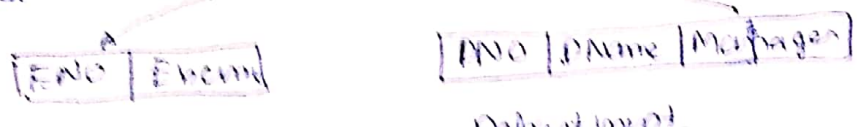
Employee

| ENO | Ename |

Relationship

| DNO | Dname |

(ii) Include primary key of one entity to total participation side.



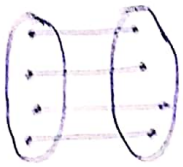
Employee

Department

(iii) We added total participating side because, it doesn't have any constraint input as NULL.

(iv)

Max : 1 participation
Min : 1 participation



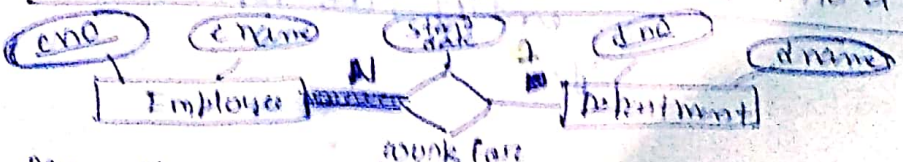
| PNO | Pname |

| LNO | L.Score |

↓ combine both tables

| PNO | Pname | LNO | L.Score |

Step 4 : Convert 1 to N relationship into a table.

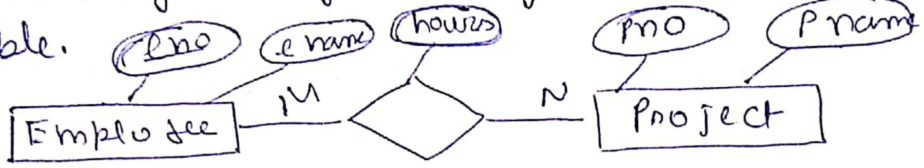


Max : N
Min : 1

N Merge relation to N side

| ENO | ename | dno | start date | dno | dname |

5: Converting Many to Many relationship into table.



works on

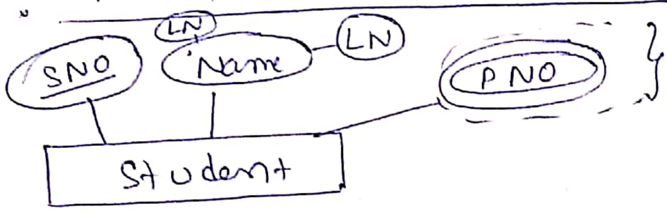
eid	e name

Pno	P name

eno	pno	works

* we not able to merge it to any side.

Step 6: Multivalue attribute



} Multivalue attribute

(a) For Multivalue attribute create one new table.

PNO	StudentNo
1	P1
1	P2
1	P3
:	:

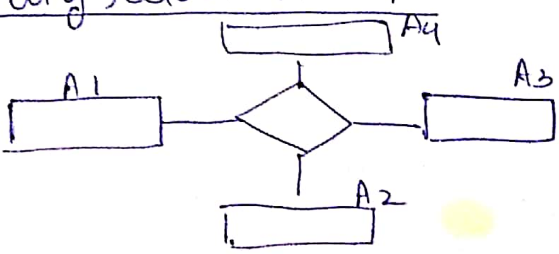
SNO.	FN	LN

PK = {(PNo, student No)}

(b) If multivalue itself a composite attribute, then break it into simple.

SNO	Std	PNO

Step 7: N-ary relationship



A1	A2	A3	A4

(a) Create a table represent a relationship.
(b) Add Forign key to all table.
(c) Combination of all attribute act as primary key for relation

Summary

ER Model	Relational Model
(a) Entity type	"Entity" Relation
(b) 1:1 or 1:N relationship type	"Foreign key" (or new table)
(c) M:N relationship type	"relationship relation" + 2 FK's
(d) n-ary relationship type	relationship "relation" + n FK's
(e) simple attribute	Attribute
(f) Composite attribute	set of simple component attrs
(g) Multivalue attribute	Relations and FK
(h) value set	Domain
(i) Key attribute	Primary key

Multi
(i)

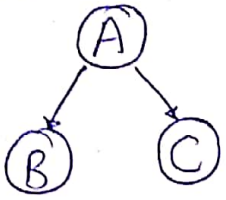
Database Transaction

Multi Granularity Locking

(i) It is tree based protocal

(ii) It apply locks from top to bottom and apply unlock from bottom to top.

eg:



IntoiveSharedLock(A)
GS(B)
U(B)
U(A)

(2) Wound Wait and Wait Die Algorithm

(i) There are pessimistic deadlock avoidance algorithm.

(ii) There algorithm cause more transaction abort than needed.

Wait Die: T & S
T → old, S → new
Need resource held by T, then S abort
T need resource held by S, then T wait

Wound wait
T need resource held by S, then S will abort.

(3) (i) Conflict equivalent and Conflict Serilizable

(ii) View equivalent and view serilizable

(4) Log Recovery

(i) All transaction which are commited after checkpoint must Redo and All uncommited Transaction after checkpoint must be Undo.

(ii) When checkpoint occur all data is permanently save into database.

(5) Schedule (IMP)

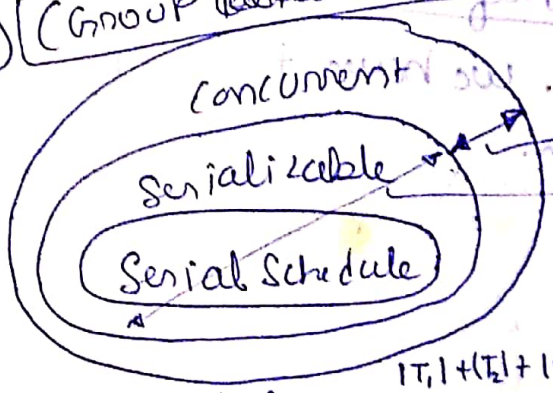
(Group combination) (Transaction coherication) not change

T_1: r_1 r_2 . . . .

T_2: op_1 op_1 . . .

T_3

T_n:



Total concurrent schedule: $|T_1|+|T_2|+|T_3|+\cdots|T_n|$

Total serial Schedule: $n!$

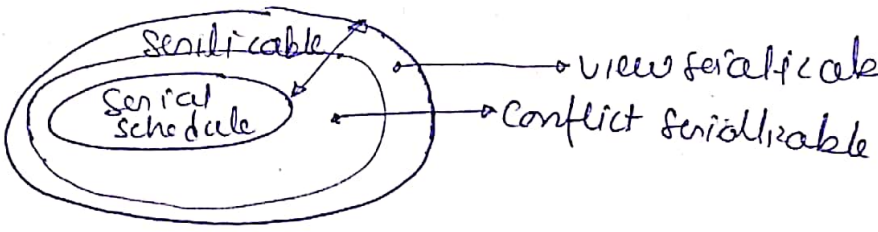
Non serial = Total conc − Total serial

$|T_1|$ * (1) only one way to place in order as per condition

different ways we can select schedule from T_1

(b) Schedule are conflict equal then precede graph are same. vice versa not true.

(c)



* If S_1 and S_2 are conflict equal schedule then both are also view equal.
* If S_1 and S_2 are not conflic equal schedule then both may or may not view equal.
* If S is conflict serializable schedule than, S is also view serializable schedule.
* If S is not conflict serializable then S may or may not view serializable

(d) All conflict serializable schedule are ~~conflict equal~~ view equal serializable. If schdule is conflict serializable then 2 phase locking allow.

(e)

(e) EMP Topological order gives only no. of serial schedule conflict equal to schedule S.
To find no. of schedule conflict equal to given Schedule S. we have to use permutation and combination.

Relational Calculus

Formal Language

- Relational Algebra (Procedural)
- Relational Calculus
  - Tuple RC
  - Domain RC (Non Procedural)

(i) All are equivalent in power.

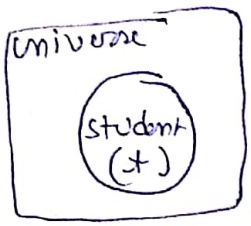
(ii) Relationally Complete : If language is able to express every thing that relational algebra express, then it is relationally complete.

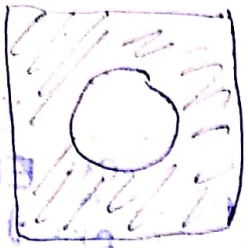
(iii) Relational Algebra act as measuring stick because it develop first, so all language measure on the basis of it.

(iv) In TRC, we examine whole tuple at a time, if we define a variable, then that variable take one complete tuple as input.
In DRC, we examine over column mean variable define over column one complete.

(v) TRC is unsafe because:

Student(t) · ; ~t





complete universe except Student, which is infinite.

② Syntax in tuple relation calculus

Student

FN	LN	Mark

{ t | }

ⓐ declare a tuple variable.

ⓑ Specify range over that variable

{ t | table_name(t) }

Range is complete table

ⓒ Specify condition :

{ t | table_name(t) AND t.attribut 1 Var }

ⓓ TO get only specific attribute as output

{ $t_1.A_1, t_2.A_2, \ldots$ | $T_1(t_1)$ AND $T_2(t_2)$ .... AND condition }

---

③ Bounded and Free variable

ⓘ { $t_1.A_1, t_2.A_2$ | ____ }

$t_1.A_1, t_2.A_2$: Bounded

Expression

- Atomic or Simple expression
  - ex: EMP(t)
  - ex: $t.A \gtrless 20$
  - ex: $t_1.A_1 > t_2.A_2$
- Complex or composite expression

Every atomic exp. are composite exp. or combination of atomic exp is also composite exp.

ⓐ $A_1 \vee A_2$ ⓓ $\exists t(\_)$

ⓑ $A_1 \wedge A_2$ ⓔ $\forall t(\_)$

ⓒ $\neg A_2$

ⓘⓘ If variable have any quantifier than that is bounded variable.

$\exists t\,(t(A) > 50)$ { Here t- range over table }

(a) ∃ means there exsist atleast one tople such that A value for that tople is greater than 50.

(b) We get final result after processing all tople of table and in final we get True if such tople exsist and False if there is no such tople. { There exsist }

(ii) $\forall t\,(t(B) < 10)$ { For all }

(a) ∀ means for all tople the value of attribute B is less than 10.

(b) After processing each tople one by one if we get true for all then final result is TRUE and if one tople fails then it stop there and return False.

(v) $\sim \exists t\,(t(A) > 50)$ any tople

(a) ∄ means there does not exsist, whose A value is greater than 50.

(b) This one too procces each tople one by one same as ∀ (for all case)

Example Table

Social Security NO.

Employee (Fname, Minitial, Lname, SSN, Bdate, Address, Sex, Salary, Super_SSN, Dno)

Department (Dname, Dnumber, Manager_SSN, Mgr_start_date)

Dept_location (Dnumber, Dlocation)

Project (Pname, Pnumber, Plocation, Dnum)

Works_on (ESSN, Pno, Hours)

Dependent (ESSN, Dependent_name, Sex, Bdate, Relationship)

Q List Fname and address of all employee who work for Dpno = 10.

Sol: $\{ t.FN, t.Add \mid EMP(t) \wedge t.dno = 10 \}$

Q List all the dependent names of an employee whose SSN is 10.

Sol. $\{ t.d\_name \mid Dependet(t) \wedge t.ESSN = 10 \}$

Q List all the dependent names of an employee whose first name is "Ram".

Sol: $\{ t.dname \mid depen(t) \wedge (\exists e)(Emp(e) \wedge e.SSN = t.essN \wedge e.FName = "Ram") \}$

t: Free variable

e.FName = "Ram": return TRUE or False

(a) It keep one searching employee table till it found one tuple which satisfy this relation, it stop there and return true if found.

(b) Work as NESTED LOOP JOIN.

Q List the name and address of all employee who work for the "Research Department".

Sol: $\{ t.name, t.add \mid \underbrace{emp(t)}_{\text{Free variable}} \wedge \underbrace{\exists e(dept(e) \wedge e.dname = "Research" \wedge t.dno = e.dno)}_{\text{bounded variable}} \}$

Q For every project located in "Stafford", list the project name, the controlling department number and the department manager SSN.

Sol: $\{ t.pname, t.dno, e.mgr\_ssn \mid \underbrace{project(t) \wedge dept(e)}_{\text{cartesian product}} \wedge (t.location = "stafford" \wedge t.dno = e.dno) \}$

If there are more than one free variable then we have to do cartesian product b/w these two tables.

⑤ For every project located in "Stafford", list the project number, the controlling department number and the department manager last name, birth date and address.

Sol: $\{ p.pn, p.dn, e.ln, e.bd, e.add \mid Project(p) \wedge Employee(e) \wedge \exists d (Depart(d) \wedge d.dn = p.dn \wedge d.mgrssn = e.ssn) \}$

---

⑥ Make a list of project number for projects that involve an employee whose last name is "Smith", either as a worker or as a manager of the controlling department for the project.

Sol: $\{ p.pnumber \mid Project(p) \wedge ((\exists e)(\exists w)(Employee(e) \wedge Worker(w) \wedge w.pno = p.pno \wedge e.lastn = "Smith" \wedge e.ssn = w.essn)) \vee ((\exists m)(\exists d)(Employee(m) \wedge department(d) \wedge p.dno = d.dno \wedge d.mgrssn = m.ssn \wedge m.lname = "Smith")) \}$

---

⑦ Conversion b/w quantifiers.

	A	B	ⓒ	ⓑ	ⓐ	
	1		T	F	T	T
	2		T	T	T	T
	3		F	F	T	T
	4		F	T	T	T

① ⓐ $\exists t.A > 0$ → T

ⓑ $(\forall t)(t.A > 0)$

ⓒ $t.A \leq 0$

ⓓ $\sim(t.A \leq 0)$ ⓔ $\forall t(\sim(t.A \leq 0))$ ⓕ $\sim(t.A > 0)$ ⓖ $\forall t(\sim(t.A > 0))$

ⓗ $\sim(\forall t(\sim(t.A > 0)))$ ⓘ $\sim(\exists t(t.A \leq 0))$

There doesn't exists any t, for which $t.A \leq 0$.

② $\forall x(P(x)) \equiv \sim(\exists x(\sim P(x)))$

---

⑪

	A	B	
	1		F
	2		F
	3		F
	4		F

ⓐ $x.A = 1$ ⓔ $\forall x(x.A \neq 1) = F$

ⓑ $\forall x(x.A = 1) = F$ ⓕ $\sim(\forall x(x.A \neq 1)) = T$

ⓒ $\exists x(x.A = 1) = T$ ⓖ It is not true that for all values of $x.A \neq 1$.

ⓓ $x.A \neq 1$ ⓗ $\exists x(P(x)) \equiv \sim(\forall x(\sim P(x)))$

(iii)

A	B	
1	1	F
2	2	F
3	3	F
4	4	T
5	5	T
6	6	T

Check: Whenever A is greater than 3 check if B is greater 3 or not.

(a) $\sim(x.A > 3 \wedge x.B > 3)$

(b) $\forall x\,(x.A > 3 \wedge x.B > 3) = F$

Here A stop checking, but our focus is on B.

(c) $\sim(x.A > 3) \vee (x.B > 3)$

Here we make A not important now whole result depends on B.

(d) $\forall x\,(\sim(x.A > 3) \vee (x.B > 3))$

Check all table if A value is greater than 3 then check B is greater than 3 or not.

(e) Shortcut $\boxed{\forall x\,(x.A > 3 \rightarrow x.B > 3)}$

whenever this one is false then don't worry ~~about~~ Because our focus is not it.

P	→	Q	
F	→	F	→ T
T	→	T	→ T
T	→	F	→ F
F	→	T	→ T

Focus

Irrelevant

(f) $\forall x(P \wedge Q) \cong \sim \exists x(\bar{P} \vee \bar{Q})$

$\exists x(P \wedge Q) \cong \sim \forall x(\bar{P} \vee \bar{Q})$

(g) $\boxed{\forall x\,(P(x) \rightarrow q(x)) \equiv \forall x\,(\sim P(x) \vee q(x))}$

$\{e.\text{Fname} \mid \text{Employee}(e) \wedge ((\forall d)(\text{Dependent}(d) \wedge d.\text{SSn} = e.\text{SSn} \wedge (d.\text{sex} = \text{"M"})) \rightarrow d.\text{Age} > 30))\}$

Employee

	FN	SSN	Ser
e →	Ram	1	M
	Shyam	2	M
	Ravi	3	M
	Sita	4	F

d →

essN	sex	Age
1	M	25
1	F	25
1	M	15
2	M	50
2	F	13
3	F	20
4	F	24

Sol: (a) Query is checking For each employee, all the department table.

(b) For each employee d is initialise to first now always this is why it is written inside.

(c) Condition of Query is, If dependent of employee is male then check it's age is greater than 30 or not. And it must true for all dependent table for each emp.

(d) Output

(i) ~~Ram 1 M~~ ✗ → T, T, F → F

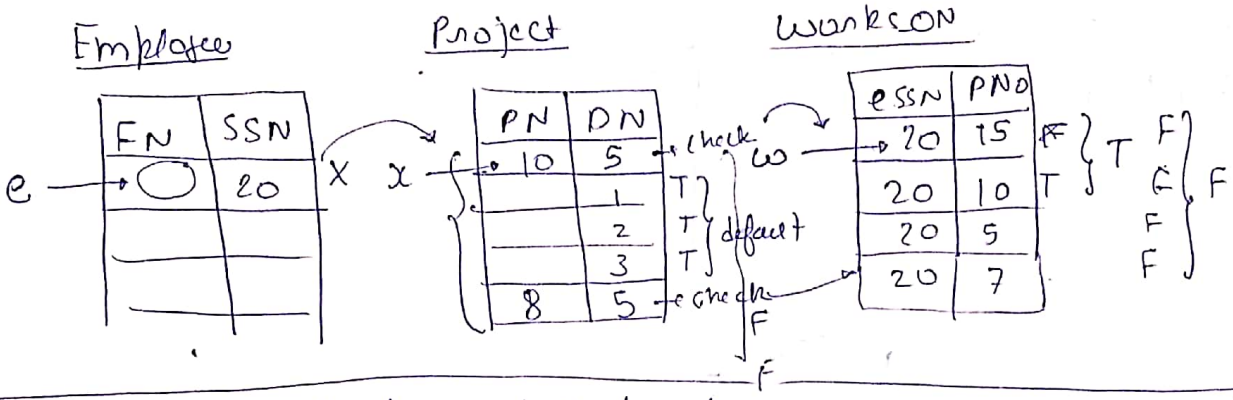
(ii) Shyam 2 M ✓ → T, T, T, T, T, T → T (output)

(iii) Ravi 3 M ✓ → T, T, T, T, F, T → F (output)

(iv) Sita 4 F ✓ → T, T, T, T, T, T → T (output)

Ex: List the Fname of employee who work on all project controlled by department number 5. (Atleast one project is controlled by dpno = 5).

Sol: $\{ e.Fname \mid Employee(e) \text{ AND } ((\forall x) (Project(x) \text{ and } D.n=5) \rightarrow (\exists w) ((works\_on(w) \wedge w.essn = e.SSN \wedge x.pno = w.pno))) \}$



(5) Unsafe relational calculus

ex: which of the following relational calculus expression is not safe ?

(a) $\{ t \mid \exists u \in R_1 (t[A] = u[A] \wedge \neg \exists s \in R_2 (t[A] = s[A]) \}$

(b) $\{ t \mid \forall u \in R_1 (u[A] = \text{"X"} \Rightarrow \exists s \in R_2 (t[A] = s[A] \wedge s[A] = u[A])) \}$

✓ (c) $\{ t \mid \neg (t \in R_1) \}$

(d) $\{ t \mid \exists u \in R_1 (t[A] = u[A]) \wedge \exists s \in R_2 (t[A] = s[A]) \}$

* If tuples are infinite then it became unsafe.
* Relation calculus with unsafe operation do not powerful.
* whenever there is negation, then there is chance of unsafe.