Teacher's Sign / Remarks

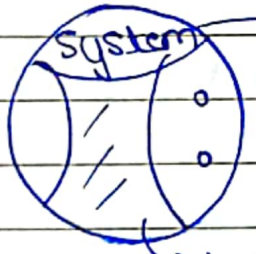
classmate Date Page 1

DATABASE

imp

- File organisation didn't have any data abstraction, so DBMS.
- DBMS: concept / hypothesis to folic user to store, manage & access data.
- RDBMS is the implemented version. [External view, Logical view, Physical view].

→ Any Piece of data is physically stored in files only, but logically in the form of tables in dbms.



System.dbf contains the metadata; user info, list of objects, list of cols, location of data.

* Our SQL Query (ext. view), hits system.dbf, → logical view → Physical view. extracts data out of file blocks & back to user. [Vimp]

* INTEGRITY CONSTRAINTS

- data integrity constraints → unique, not Null, check, primary key.

validate correctness accor to value.

→ Unique constraint cant validate whether the value is NULL/not.

* FUNCTIONAL DEPENDENCIES

– determinant → dependent

imp
- If det. gets duplicated, dep also gets duplicated.
- If det. is unique, dep. can have any value.

(i) R → R always valid [Tautology]
(ii) A → BC ≡ A → B & A → C.
(iii) A → B & B → C valid.
∴ A → C also valid.
(iv) ∵ BC → BC ; ∴ BC → B & BC → C also valid.

eg → Pg 7 [Instance of relation]

4) Super Keys: if all attributes of relation can be derived by them. can be single/composite.
2) Candidate Keys: if AB & A can derive all attributes, ∴ A ⊂ AB
AB → SuperKey; A → CK.
∴ CK is minimal Subset of SK.
imp

1) * Normalization: Process applied on relation to reduce dep of redundancy.
→ Data redundancy causes: more memory, update/insert/del^n anom
eg → Pg 11 [redundancy]

* NORMAL FORM: indicates amt of redundancy present in it.
imp
NF ∝ 1/(dep of redundancy)
Normalis^n
- Step 1: find out highest NF.
- Step 2: decompose into higher NF.

Step 1: Find out highest NF:

Step (1) Find out all CKs.
eg → Pg 13 (Sno, Pno)
Step (2) Find prime/Non-Prime attributes.
eg → Pg 6 (Rev Copy) [Imp]

* Problem with redundancy:
eg → Pg 15 [I/U/D anomaly]

Step (3) Identify dependencies.

(i) Full Dependency
CK/SK → Prime/Non-Prime
– never causes redundancy.

RHS = Non-Prime (PD & TD)

(iii) Partial dependency

Part of CK → Non-Prime

NOTE: If simple CK, no PD exists

imp

(iii) Transitive dependency

Non-Prime → non-Prime

Part of CK + Non-Prime → non-Prime

NOTE: If a relation has only Prime Attri.; [PD & TD = 0] Imp

(iv) Overlapping CK dependency

Prime/Key → Prime/Key

Step 4) Types of Normal Forms imp

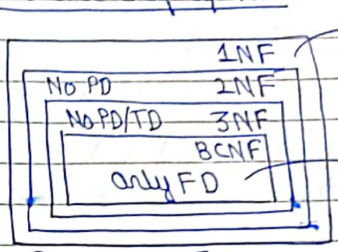
(i) 1NF: all values in relation must be atomic. Every relation → 1NF.

(ii) 2NF: No PD

(iii) 3NF: No PD, no TD

(iv) BCNF: only FD (no OCD).

* Hierarchy of NF



weakest

Vimp

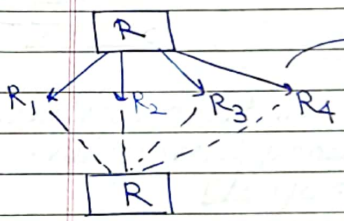
strongest.

eg → Pg 22 [Q2] (OCD)

eg → Pg 23 [Q3] (CK defin)

eg → Pg 25 [Choose but clause & comb. missing attributes]

STEP-2: Decompose into higher NF



Properties of Normal:

(i) lossless decomp.

(ii) Dependency preserv.

(ii) lossless decomposition: By join R, R2... → quer. availib. of data if we can derive all data in eg. R.

(iii) dependency preserving: By using the projected FDs, if we can derive all FDs. Guarantees integrity constraints.

eg → Pg 29 [1NF → 3NF form] imp [manual normalisation]

imp [other Method also]

1* Procedure to find decomposition is lossless or not.

[Y→Y ✓ Y→X ✓ X→Y ✗]

eg → Pg 11 [Rev Copy] (if intersection filled with X, lossless decomp)

2* Procedure to find decomposition is dependency preserving or not.

(i) find all covers. (Vimp)
(ii) apply the FD properties on decomposition
(iii) Check if you can derive all FDs of eg relation [using FD properties]

eg → Pg 12

* BCNF Algorithm. (Good Q)

Guarantees lossless decomposition, but not dependency Preservation.
[may/may not]

eg → Pg 34 [X→Y an (not FD)] X+ → R−[Y]

→ G covers F if all the FDs of F are implied by G.

* 3NF Decomposition Algorithm

Guarantees lossless + dependency Preserving decomposition.

→ not always unique.

- Minimal cover: eliminates unnecessary attributes on determinant side.
  AB→C, A→C ; AB→C, A→B. (imp)
- Canonical cover: on dependent side.
  AB→CD ∵ A already det. C
  A→C

eg → Pg 39 R(A, B, C, D, E)

(2) RELATIONAL ALGEBRA

(ii) Set operations: { ∪, ∩, −, × }
(iii) Native operations: { σ, Π, ρ, ÷, ⋈ }

- degree: No. of attributes/columns
- cardinality: No. of rows
- domain: a char, b int (imp)

* To be union compatible, degree & domain of respective attributes must be same.

(i) Set Theoretic operators

(ii) Union: d(R∪S) = d(R) = d(S)
max(|R|, |S|) ≤ |R∪S| ≤ |R|+|S|

(iii) Intersection: d(R∩S) = d(R) = d(S)
0 ≤ |R∩S| ≤ min(|R|, |S|) (imp)

(iii) Set-diff: $d(R-S) = d(R) = d(S)$, $0 \le |R-S| \le |R|$

(iv) Cross-Product (imp): $d(R \times S) = d(R) + d(S)$, $|R \times S| = |R| * |S|$
↳ rel^n need not be union compat.

(vi) NATIVE OPERATIONS [like relations]

(1) Join: subset of Cartesian Product
- Join Type: inner / outer / left / right outer
- Join Condition: natural / ON / using.

a) Join Condition (imp)
→ Natural: atleast 1 common column, only '=' used, eliminates duplicate of common columns [Equi Join]
→ ON cond / Theta Join: no prerequisite, any operator used, duplicates not removed. (imp)
→ Using: same / all col used.
If all common columns are used, result will be like Natural Join.

b) Join Type.
→ Inner: allows only matched rows
→ LOJ: Matched + left's Unmatched
→ ROJ: Matched + Right's Unmatched

eg → Pg 48

Join consider a row as Qualified, if it gets matched atleast once.

(2) DIVIDE: (imp)

For $R \div S$: Att. of $S \subseteq$ Att. of R

Divide considers a row as qualified, if it gets matched with all the ref. values.

eg → Pg 49

(3) Selection (σ): Produce logical Horiz subset of rows.
– 'where clause in SQL'.
$d(\sigma_{result}) = d(original)$
$|\sigma_{result}| \le |original|$

(4) Projection (Π): Produce virtual subset of rows by elimination duplicate.
– 'Select with Distinct' clause in SQL (imp)
$d(\Pi_{result}) \le d(original)$
$|\Pi_{result}| \le |original|$

(5) Rename: doesn't change deg / cardinality

③ SQL

- Ansi Stand SQL → DDL: create, alter, rename, truncate, drop
  → DML: Insert, upd:, del, sel
  → DCL: grant, revoke
  → TCL: commit, rollback

DDL → truncate: only user data gets del
drop : user + Metadata gets deleted

imp * Truncate: delete all rows of Table
delete: delete 1/some/all rows

* SELECT (DML Command)

imp

eg → Pg 52 [order of exec]

eg → Pg 53 [operator precedence] → OR, AND, NOT

where deptno = 10 or deptno = 20 or deptno = 30	≅ where deptno IN(10,20,30)

imp * sal ≤ 5000 and sal ≥ 2500 ≅ sal BETWEEN 2500 and 5000

* LIKE: where ename like '____' (3 char name)
where ename like 'S%'

eg → Pg 55 [Exc]

* FUNCTIONS IN SQL

Single Row funct
- Applied on 1 row at a time to get one result per row.

Multi row / Grp / Aggregate funct
- max(), min(), sum(), avg(), count()
- Can't be used in where clause
- imp [can be used with having]

* Group By

diag → Pg 22 [Rev Copy] (6 cases with Group By clause → Vimp)

NOTE: All the non-group exp that exists in the Select clause along with group exp must & should be present in the group by clause. imp
(Agg. funct.)

* Order By → sorting by an attrib.

- By default, SQL displays the result set in the same behaviour/order as data got inserted.

eg → Pg 60 [money exp in order by clause]

* SubQueries

- Named SQ → Single Row SQ, Multi Row SQ
- Correlated SQ

1) (ii) Single Row Sub Query
↳ SQ returns 1 Value
eg → Pg62 [Q1]

(iii) Multi Row Sub Query
eg → Pg24 [Rev Copy]

imp
Normal SQs, get executed only once irrespective of no. of rows, we have in Table.

2) Correlated SubQueries
- SQ gets executed as many times as no. of rows in Table.
eg → Pg66 [Imp] ← imp

* SET OPERATIONS IN SQL
- Union in SQL eliminates duplicates
eg → Pg66 [Imp]
↓
where clause maps row by row

* PREVIOUS YEARS.
1) If instance of a relation is given, we can say A → B (may/maynot)

→ We need full picture.
2) Any relation with 2 attributes is always in BCNF (Binary Rel).

eg → Q2-40 [Max no. of SK] [ME]

eg → Q2-24, Q2-32 [Max Tuples when NJ is applied] min(|R|,|S|)

* SQL Query will work even if there are no indexes on relations. SQL
imp → doesn't permit attribute names to be repeated in same relation.

* To check for lossless decomposition:
1. Att(R1) ∪ Att(R2) = Att(R).
2. Att(R1) ∩ Att(R2) ≠ ϕ
3. Att(R1) ∩ Att(R2) ⇒ must be a CK for a relation [Vimp]

4) TRANSACTION MANAGEMENT

Transaction: group of DML inst. to convert data from 1 CS to another. It must satisfy ACID Properties:

(i) Atomicity: either all instructions of trans. must get executed / None. embedded b/w [Begin_trans, end_tra

maintained with help of Shadow Paging (imp)

(iii) Consistency: data b4 trans. & after trans. must be consistent (Consistency)
→ Shadow Paging used.
↳ copy of data item before & after transaction is kept.

(imp) TCL
(1) commit: all modif performed by trans. is transferred from DB cache to SM.
(2) rollback: Ignore all modifications & replace such values with previously commited values.

(iv) Isolation: a trans cant effect any other trans. during execution
Schedule → serial
→ concurrent

→ Serial Schedule (imp)
· all init. of trans. get executed non-preemptively.
if n Transaction = n! Serial Sch
Adv: Guarantees data consistency
Disadv: High WT, less Throughput

→ Concurrent Schedule
Adv: less WT
disadv: cant guarantee consistency due to

* Conflicting Operations

diff trans / same data item
W—W: Lost Update Problem
W—R: dirty read Problem
R—W: unrepeatable read prob (imp)

* Serializable Schedule
A concurrent Schedule is serializable if result = result of any of Serial S (imp)

eg → Pg 75 [WW conflict]
(Common DB cache, local Cache).

eg → Pg 30 [Rcv] Pg 78 (Imp Q)

If conflicting operations ⇒ Possibility of data inconsistency not always

(imp) if Serializable ⇒ data is consistent

⇒ Conflict Serializable: If given conc. schedule is conflict equivalent to any of the serial Schedules.
[Same for view Serializable]

Schedule [based on Consistency]
- Serial (C)
- Concurrent
  - Serializable (C)
    - Conflict
    - View
  - Non-Serializ (I)

(imp)

* if Schedule is either conflict/view both ⇒ Serializable. (Gate)

1) Conflict Serializable

→ 2 Schedules are conflict equivalent if order of conflicting oper is same.

eg → Pg 81

To find conflict Serializ: use Precedence Graph Algo.

- Perform Topological ordering on Graph to find equivalent Serial Sched. for concurrent Schedule. (Gate)

eg → Pg 84 [Gate 2015]

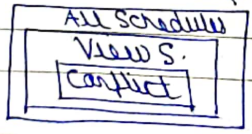
eg → Pg 86

2) View Serializable (imp)

(ii) Trans. that perform init read on S1, should also perform init read on S2, also on same data item.

(iii) Same with final write.

(iii) If RWConflict exist on same data item in S1, same RWConflict must exist in S2 also on same data item.



Serializable (Imp)

eg → Pg 89 [Q 1] (View Serial)

eg → Pg 90 [If, there exist more than 1 trans, with init read, & atleast 2 perform write on that data] not VS (imp)

init read = more prior no write.
final write = more head no write.

(iv) Durability: All modif. performed by commited trans. are assured to be present in SM (no matter what type of failure). (Gate)

* WAL Protocal: All modif by every commit. trans. must be written to a redo log file before actual data file.

used in:

* RECOVERY → Analysis Phase, Redo Phase, Undo Phase.

redoing all entries of redo log file

eg → Pg 93 [Diag] (Gate)

eg → Pg 94 [Diff b/w commit & Check Point] (imp)

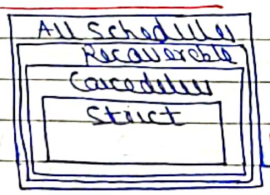
* Schedules based on Recoverability

(i) Recoverable Schedule : if $T_i \rightarrow T_j$ exists, T_i = write, T_j = read; then completion order must also be T_i, T_j. (imp)

(ii) - Cascading rollbacks are caused.

(iii) Cascadless Schedule : dirty read not allowed. Trans can read an data item updated by only committed transaction.

(iii) Strict Schedule : No dirty / lost update ; Trans can't read / write on the / modify / done by other trans. unless committed.



eg → Pg 99

eg → Pg 100 / Pg 101

Find out highest recoverable Schedule. (imp)

* CONCURRENCY CONTROL

- RDBMS uses locking mechanism.

→ Lock Manager accept / reject lock req. of transactions using lock compatibility Matrix.

if trans. wants to perform
read : Shared
read / write : exclusive

(imp) → Concurrency control aims for produc. Strict Serializable Schedule always.

eg → Pg 36 [Rev Copy] lock compatibility Matrix.

As per RDBMS, Locking & Unlocking should be performed as per a protocol called 'Locking Protocol'.

1] 2 Phase Locking Protocol

- Growing Phase : trans only allowed to acquire locks.
- Shrinking Phase : trans only allowed to release acquired locks

Problem : (i) deadlock Problem. (imp)
(ii) Cascading rollback Prob. [not Strict]

→ always ensures Serializable Schedules

eg → Pg 104 (Deadlock Problem)

2] Strict 2 Phase locking Protocol

imp

- A trans. cant release all the acquired exclusive locks, till it commits
Drawback: deadlock Problem
Adv: Produces strict Serializable Schedule.

3] Time Stamp Based Protocol

- Unique TS will be assigned to each & every transaction.
Adv: No deadlock. Gate

(5) ER DIAGRAMS

* Referential integrity constraints

* FK: PK/CK of master table that exists in child table.
→ A value is allowed in the FK column if it has reference to one of the values of PK column of parent table

imp

Relations can have many CK, but out of them, only 1 becomes PK

must be defined

→ on delete cascade: If PK value gets deleted, all refer FK values also deleted
→ on delete No action: User manually updates FK values, so that if PK deletes, FK no action

* Basic ER Diagrams

combine child + relationship to min. no. of tables. → many side

imp

eg → Pg 40 (dias) [Rev Copy]

* Participants

Complete/ Total
- FK values should be NULL in child Table (By default) [Double line]

Incomplete/ Partial
- FK values can be NULL [Dash line]

imp

* Weak / Strong entities

- doesn't have PK
- in rel. with SE, with full Particip^n.

cal | list of cal. + PK of SE → PK
of WE

eg → Pg 114 [diag] Imp

* Gate QS.

eg → Pg 115 (2008) [ER diagram min]

eg → Pg 119 (2005) [Correlated SQ]

eg → Pg 121 [NJ, Union, Intersection, difference, divide] – RA.

eg → Pg 123 [RA & TC] imp

* Question on Normalization.

eg → Pg 42 (Rev Copy)

eg → Pg 43 [R(A,B,C,D,E,F,G)] imp

(6) FILE ORGANIZATION

In OS, we needed to access file from begin to end, but in dbms, we need same record from a file. ∴ file organization also diff.

table/file

→ mapped to PB in Hard disk.

* Strategies for storing file of records into blocks:

imp

(1) Spanned: allows partial part of record to be stored in block. No waste of mem. – variable length records.

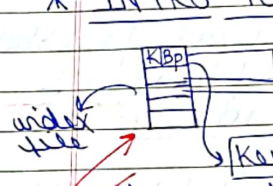
(2) Unspanned: No record can be stored in more than 1 block.
– fixed length record.

* Organization of records in a file

(i) Ordered file organization: records are ordered, based on a field
– Binary Search used to search record.
Insertion → expensive

(ii) Unordered file organization: records in file are inserted whenever space is available; at EOF. (Linear Search)

* INTRO TO INDEXING.



– Index = ordered file → also div in blocks
– To access a record using index:

imp

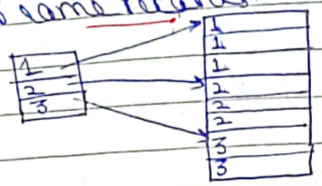
avg no. of block Access ⇒ $\log_2 Bi + 1$
(index block access; data block access)

* Classification of Indexing: imp

(1) Dense index: if index entry is created for every distinct Search Key value.

imp

(2) Sparse index : index created only for some records.



[Sparse & dense both]

* TYPES OF INDEXING

Single level index → PI, CI, SI

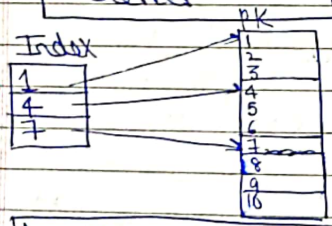
Multi level index [B / B+ Tree]

1) Primary Index [PK + Ordered]

- Index created for first record of each block

No. of index entries	=	No. of data Blocks

imp



[may/may not] Sparse index

∴ Avg Block Access = $\log_2 Bi + 1$

eg → Pg 7 [Primary indexing] imp

2) Clustering index (Non Key + ordered)

- ordered on non-key. ∴ repeated.

imp

→ for every distinct key value : 1 index record. [Sparse + Dense]

∴ Avg Block Access = $\log_2 Bi + 1$

3) Secondary index [Non Key / CK + ordered]

- index [may/may not] created for every record in file. [Dense, not Sparse] imp
- SI provides a secondary means of accessing file for which PK already exists.

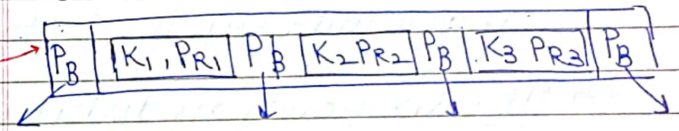
eg → Pg 10 [Secondary Index] Gate

An index is clustered if data records of file are organized in the same order as the data entries of index.

1) * B-TREES

→ Generalization of Multilevel index.
→ Node / Block Pointer } + Key (at each level).
→ Record Pointer

imp



∴ 4 Children, 3 Keys.

* P = order / max children node can have

(ii) Root: children ∈ [2, p]
Keys ∈ [1, p-1]

(iii) Internal node:
children: $\lceil p/2 \rceil$ to p
Keys: $\lceil p/2 \rceil - 1$ to p-1

Same for leaf node

eg → Pg 14 (max children / order = 25)

eg → Pg 16 [Vimp Q] (4 levels)

* UNDERFLOW / OVERFLOW in B-Tree

overflow: if Keys exceed 'p-1'.
underflow if Keys < $\lceil \frac{p}{2} \rceil - 1$.

Insertion problem deletion issue.

1) * B-Tree Search

- First go into root; if match: Stop. compare with Key, if < ; then go left
- All index records are distributed over B-Tree.
- Search the node, till we find a Key greater than the Key we are searching for, & go left

As, order ↑, Search time ↓ → depth ↓, breadth ↑
Time complexity = $\log_m n$

p → order of B-tree,
n → no. of Search Key

2) * B-Tree Insertion

- Split leaf node's Key into 2 parts & promote median Key to Parent.

eg → Pg 19 [Eg 1 on Insertion]

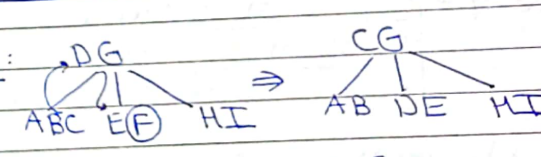
⇒ When we split the node:
if p = odd: $\frac{p-1}{2}$, $\frac{p-1}{2}$
p = even: $\lfloor \frac{p-1}{2} \rfloor$, $\lfloor \frac{p-1}{2} \rfloor + 1$

Split when overflow occurs.

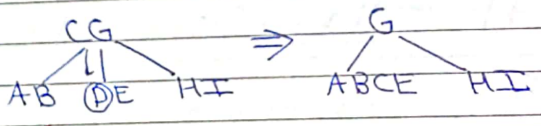
3) * B-Tree deletion

order = 5

Borrow:



Merge



② B+ TREES

- B-Trees : distribute (Key-record P) throughout the tree

→ B+ Tree :

Internal node : remove record Pointer only Key + Node Pointer.

imp → ∴ Breadth ↑, depth ↓, Time ↓

leaf node : (Key, Rp) pair + 1 NP.

* Internal Node: | K_1 | K_2 |

children : $\lceil P/2 \rceil$ to p

Keys : $\lceil P/2 \rceil - 1$ to $p-1$ → B Tree

* Leaf Node: | KPr | KPr | ... | KPr | P_N |

imp

P_N points to next sibling.

imp → Order of Non-leaf : max no. of children it contain.

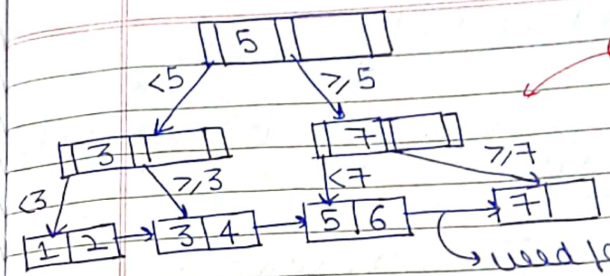
Order of Leaf : max no. of K-V pairs present inside leaf.

eg → Pg 26 [Order of B+ Tree]

4) Searching B+ Tree

Time complexity = $\log_p n$

p → much higher than B-Tree



imp

used for range based queries.

→ Search always terminates in leaf in B+ tree not B-Tree. imp

eg → Pg 28 [Gate-2015] 4 [nodes accessed 2]

* B / B+ Trees are perfectly balanced as distance from root to all leaf nodes are equal. Gate C

eg → Pg 32 [Gate-2008] [No. of Splits in B-tree insertion] 4

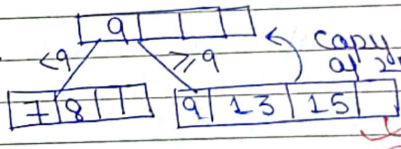
3) B+ Tree Insertion

Leaf : if order = 5, max Keys = 4.

left always small

[7 8 9 13 15]

| 7 | 9 | 13 | 15 | → insert 8.



copy smallest key of 2nd node in Parent.

imp

Non-leaf

10 20 30 40 ↓ insert 15

20

10 15 | 30 40

[10, 15, 20, 30, ...] [no copy]

eg → Pg 34 [B+ Tree insertion]

imp

Leaf: (1 4 7 10) $\left[\frac{P-1}{2}\right]$ in 1 node

& smallest in node 2, promoted

Non-Leaf: (7 17 20 25)

$\left[\frac{P}{2}\right]-1$ in 1 node ∴

17

7 | 20 25

* ht of B-tree > ht of B+ Tree

∴ B+ tree grows in breadth

∴ Search time ↓ ∵ only key is present in internal node ∴ more children

B+ tree: Range based queries easily implemented

3) B+ Tree Deletion

- If any of siblings have keys greater than minimum: Borrow
- If not: merge

In case of underflow

Key < $\left[\frac{P}{2}\right]-1$

In case of re-arrangements from Sibling: (Vimp)

⇒ replace the internal node key with the smallest element in right sub tree.

eg → Pg 36 (Eg 1) [Rearrangements]

eg → Pg 38 [Smallest element in right subtree is pushed up to satisfy B+ tree req] (imp)

eg → Pg 39 [Merging]

* PREVIOUS YEARS GATE QS

1) SQL

* SQL query can't automatically eliminate duplicates.

[RA | TC can]

eg → Pg 131 [Q3] [R has no duplic. & S is non-empty]

* Select distinct $a_1, a_2 \cdots a_n$ from $r_1, r_2, \cdots r_m$ where P.

↳ $\Pi_{a_1, a_2 \cdots a_n}\, \sigma_P(r_1 \times r_2 \cdots \times r_m)$

→ if not a correlated SQ, execute it, place it & execute the outside Q.

eg → Pg 133 [2014]

unique is used while creating Table

eg → Pg 134 [Gate 2014]

eg → Pg 136 [Form examples, see the option & run the SQL Query] (imp)

eg → Pg 138 [Gate 2014] (all customers having a good rating)

eg → Pg 139 [Gate-2016]

* Having clause should be applied only with Group By clause. All attributes must appear in Select Clause (if in Group by) else put in Aggregate function. (imp)

(2) Relational Algebra

→ if R = m tuples, S = n tuples,
max size of join = mn
min size of join = 0.

* Sum can't be computed using basic relational Algebra operations.

RA ≡ (same power) Safe Tuple Calculus (imp)

eg → Pg 142 [Gate-2004] (imp)
[Try to solve without eg]

* If common attribute 'a' is not a PK/CK, we get Spurious tuples. More tuples in case of lossy.

eg → Pg 144 [IT-2006]

eg → Pg 145 [atleast one girl not registered] (imp)

eg → Pg 146 [Use eg]

* For a join b/w relation R & relation S, using Nested loop Method
There are 3 buffers.
if size(R) < size(S),
fewer disk block accesses will be: (Size(R))
when; (R) is in the outer loop.

eg → Pg 150 [Gate 2014] [Try to solve by seeing without eg]

eg → Pg 151 [Gate 2007]
[supervisor Q]

eg → Pg 152 [2008]

eg → Pg 153 [Safe & unsafe relational calculus].

→ Safe : Tuples from domain.

* Sec usage of not (¬).
{t | ¬(t ∈ R_i)} → t may belong to many tuples in Universe ∴ unsafe.

3) TRANSACTION MANAGEMENT

eg → Gate Q-2012 [Pg 154] [Try all concurrent interleaving of T1 & T2]. [just conflicting operations]

eg → Pg 155 [Gate 2014]
r_1(x), r_2(x), w_1(x), r_3(x), w_2(x).
[Do it just].

eg → Pg 157 [Conflict Serializable & recoverable].

∵ No dirty read : cascadeless ; & recoverable Schedule
Here, t2 writes data b4 t1 aborts.

eg → Pg 158 Gate 2016

eg → Pg 159 [Strict 2PLocking]

4) FILE STRUCTURES (Prevs)

eg → 6.16. (Grp) (imp)

→ B+ tree is preffered to BST, since data transfer from disk is in Blocks.

eg → 6.5 (B-Tree Q)

eg → B+ Tree (Insertion & deletion)
Q6.12, 6.13 → imp Q.

eg → Q6.15 [Secondary indexing] (Vimp)

* TEST ANALYSIS.

* If asked CK, find closures of each option and check.

* For dependency preserv; find out closures, then project; and see if you can derive original FDs.

* If not conflict Serializable, Schedule won't be allowed by 2PL.

eg → Q5 (imp)

* No. of Block transfer in Nested loop Join Method =
$[B_{outer} + N_{outer\ (tuples)} \times B_{inner\ (block)}]$. (Gate)
[outer relation must be smaller]

* Previous Year Gate Qs

1) SQL

* An SQL Query will work suc, if there are no indexes on relations.
[SQL doesn't permit attribute names to be repeated in same relation].

eg → Q3 (Pg 131) [r has no duplicates & S is non-empty] ✓
if duplicates, result ≠ r.
(from $r_1, r_2 \cdots = r_1 \times r_2 \times r_3$.
where = Selection (σ)
Select = projection (π)

* Unique only used while creating Tables.
eg → Pg 134 [Gate 2005]

eg → Pg 136 [Whenever, they use raw sets take distinct values, not duplicates]

eg → Pg 137 [Gate 2007] (imp)
[corrected SQ, not exist].
(Take eg to answer this Q].

⇒ Natural Join → atleast one not
¬(atleast one not) = all.
¬(atleast one customer haven't given good rating) = all customers give good rating.

* [Always take small example].
eg → Pg 139 [Gate 2016]

* Having clause should be applied only with Group By clause.
All attributes must appear in Select clause if in group by. (Vimp)

* Relational algebra has same power as Safe relational calculus.

2) Relational Algebra

eg → Pg 142 [Gate 2004] (imp)
Solve just by reading expression.

* i) R(A,B,C,D) is decomposed into $R_1(ABC)$; $R_2(AD)$;
then joined to form S(ABCD) [NJ].
ii) $R_1 \cap R_2 = A$ (not (K); [lossy]
then, [S will contain spurious tuples]
∴ RCS

* $\Pi_{courseId}((\Pi_{studId}(\sigma_{sex=female}(Student)) \times \Pi_{courseId}(enrol)) - enrol)$

studid of female

all female students

all the female students paired with all courses.
[every female enrolled with every course]

eg → Pg 145 [courses in which only subset of female students have enrolled, not all] prop

eg → Pg 146 [Good Q on NJ] (column matched upto arity).

R

p	q	R_1	R_2	R_3
1	2	a	b	c
2	3	d	e	f

S

p	q	S_1	S_2
1	2	a	b
2	1	d	f

$\therefore \Pi_p(R) \bowtie \Pi_p(S)$

$\Rightarrow$ $\frac{1}{2}$ $\frac{1}{2}$ $\therefore$ $\frac{p}{1\ 2}$

eg → Pg 151 [Gate 2007] (imp)
[Correlated Subquery using Tuple Calculus]

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

(imp) return tuple such that there exists some u in R_1, such that $t(A) = u(A)$ and there exists no S belong to R_2, where $t(A) = S(A)$.

O/p comes from R_1 (Soln)

* $\{t \mid \neg(t \in R_1)\}$
tuple t, such that t doesn't belong to R_1. (Unsafe)

5) TRANSACTION MANAGEMENT.

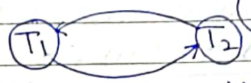
* T_1: read(P), read(Q), if P=0, then Q++, write(Q)
T_2: read(Q), read(P), if Q=0, then P++, write(P) (imp)

$\Rightarrow$ non-serial interleaving of T_1, T_2.
[Consider all cases of conflicting oper^{n}s and draw Precedence Graphs]

eg → Pg 165

Schedules

(A) $r_1(X); r_2(X); W_1(X); r_3(X); W_2(X)$



same data item, diff. transaction

∵ loop: not conflict serializable

B)

T_1	T_2	T_3	T_4
	r(X)		
W(X) com.		W(X) com.	
	W(Y) R(Z) comm.		
			R(X) R(Y) comm

→ recoverable ∵ [No dirty read also]. ∴ cascadeless & Strict too.

eg → Pg 159 [Strict 2 Phase locking] (unlock exclusive locks on data item only after commit)

* TRANSACTION CONTROL

→ If a Schedule is not conflict Serial moreover it doesn't have any blind write. (vimp)
it can also be View Serializable.

→ If not Serializable, can't accept 2PL Protocol.

* DataBase ← (vimp)
  - Pages
  - Table
  - Rows

As, we move downward, ↓ Concurrency increases.
∴ Deg of concurrency is highest, when lock applied on rows.

* A non-recoverable Schedule can't ensure transaction atomicity. (vimp)

* In Strict 2PL locking, 2 conditions must be met: (vimp)
  (i) write operation needs exclusive lock on data.
  (ii) unlock X: locks only after commit.

eg → Q 5.9

* To find if 2 schedules are conflict equivalent to each other, draw a list of conflicting operations and match. [if same, conflict equivalent].

eg → Q 5.10 [Soln is Imp formula list]

eg → Q. 5. 14