22/4/2024

Concurrency Control

→ **Protocol / technique of Concurrency Control**

- locks
- Use of time stamp
- Multiversion Concurrency Control
- Optimistic Concurrency Control

Locks ⇒

- Locking Mechanism for each element of Database.

Locks

- Binary Lock
- Shared / Exclusive lock

→ **Binary Lock** ⇒

- Two states of each element

lock (x)

unlock (x)

→ **Shared / Exclusive lock** ⇒

(Read)

[lock gets shared b/w multiple operations]

{only one can access}

{Write Lock}

Lock Table

<data-item-name, lock, no. of read, locking-transaction>

Conversion of Lock ⇒

Upgrading:
- Shared (x) ⟶ Exclusive (x)
- Read (x) ⟶ Write (x)

Downgrading:
- Write (x) ⟶ Read (x)
- Exclusive (x) ⟶ Shared (x)

Protocol used for Serializing

① 2-Phase Locking

- Basic 2-phase Locking
- Conservative 2-phase locking
- Strict 2-phase locking
- Rigorous 2-phase locking.

Basic 2-phase locking

→ A transaction is set to follow two phase locking protocol if all locking operations (read-lock or write-lock) must proceed the first unlock operation in transaction.

Two Phases ⇒

Expanding or growing

- New lock on item can be acquired but non of two lock may be release.

Shrinking phase ⇒ locks can be released but no new lock can be acquired.

Conservative 2-Phase Locking ⇒

- Require a transaction to lock all the items before the transaction begin. By pre-declaring read-set & write-set.

- If any of the pre-declared item cannot be locked the transaction doesn't lock any item and it waits until all the items are available for locks.

Strict 2-phase locking ⇒

- Transaction T doesn't release any of its exclusive locks until after it commits or abort. Hence no other transaction can read or write an item that is written by T unless T has commited.
{Recoverable}

Rigorous 2-phase locking ⇒

- Transaction doesn't release its any of the locks (shared/exclusive) until after it commit / abort.

Protocol: Use of Timestamp

Timestamp Ordering ⇒

$TS(T_1)$	$TS(T_2)$	$TS(T_3)$	$TS(T_4)$
1	2	3	4

- whichever timestamp is highest it is newest transaction.
- " " is lowest it is Oldest transaction.

Read_TS(x) → latest Read on x, returns timestamp

Write_TS(x) ⇒ Latest write on x by any transaction, will return timestamp.

Basic Timestamp Algorithm ⇒

Transaction (T) issue a write_item(x)

Operation if

1. if read_TS(x) > TS(T) or
   if write_TS(x) > TS(T) then
   abort & rollback T and reject operation.

2. If condition 1 doesn't occur, then
   execute write_TS(x) and set write_TS(x)
   to TS(T)

Transaction T issues a read_item(x)

1. if write_TS(x) > TS(T) then abort & rollback T
   & reject the operations.

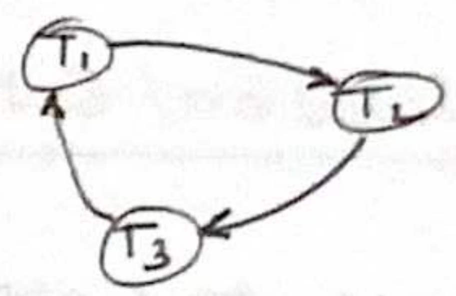
2. if write_TS(x) ≤ TS(T) then read(x) and set
   read_TS(x) to larger of TS(T) & read_TS(T).

Strict timestamp ⇒

A transaction T that issues a read_item(x) or write_item(x) such that TS(x) > write(x) has its read or write operation delayed until the transaction T that write the value x has committed or aborted.

Deadlock

- circular wait
- Mutual exclusion
- Non-preemitive



Deadlock prevention

1. → use Conservative 2-phase locking.
2. → Ordering all ~~data~~ items in database and making sure that a transaction that needs several items will lock them according to that order.
3. No waiting
4. use of time Out
5. Two Schemes of Timestamp
   - wait die
   - Wound wait.

wait-die ⇒ if $TS(T_i) < TS(T_j)$ then T_i is allowed to wait otherwise abort T_i and restart it with same time stamp.

wound wait ⇒ if $TS(T_i) \leq TS(T_j)$ then abort T_j and restart it with the same timestamp otherwise T_i is allowed to wait.

To Prevent Starvation

Protocol : Validation / Optimistic Concurrency Control

→ Here we talk about three phases :→

① Read & Execution phase – write to temp variable

② Validation phase → validate the action

③ write phase → commit otherwise rollback

– We use 3 time stamps :→

① Start (T_i) → when T_i started execution

② Validation (T_i) → when T_i enters validation phase

③ Finish (T_i) → when T_i finishes write phase.

Validation Phase – Algorithm (T_j)

if for all T_i with $TS(T_i) < TS(j)$ either of following conditions holds :→

→ finish (T_i) < Start (T_j)

→ start (T_j) < finish (T_i) < validation (T_j) and the set of data items written by T_i does not interest with the set of data items read by T_j Then the validation succeeds and T_j can be commited otherwise validation fails and T_j is aborted.

24/4/2024

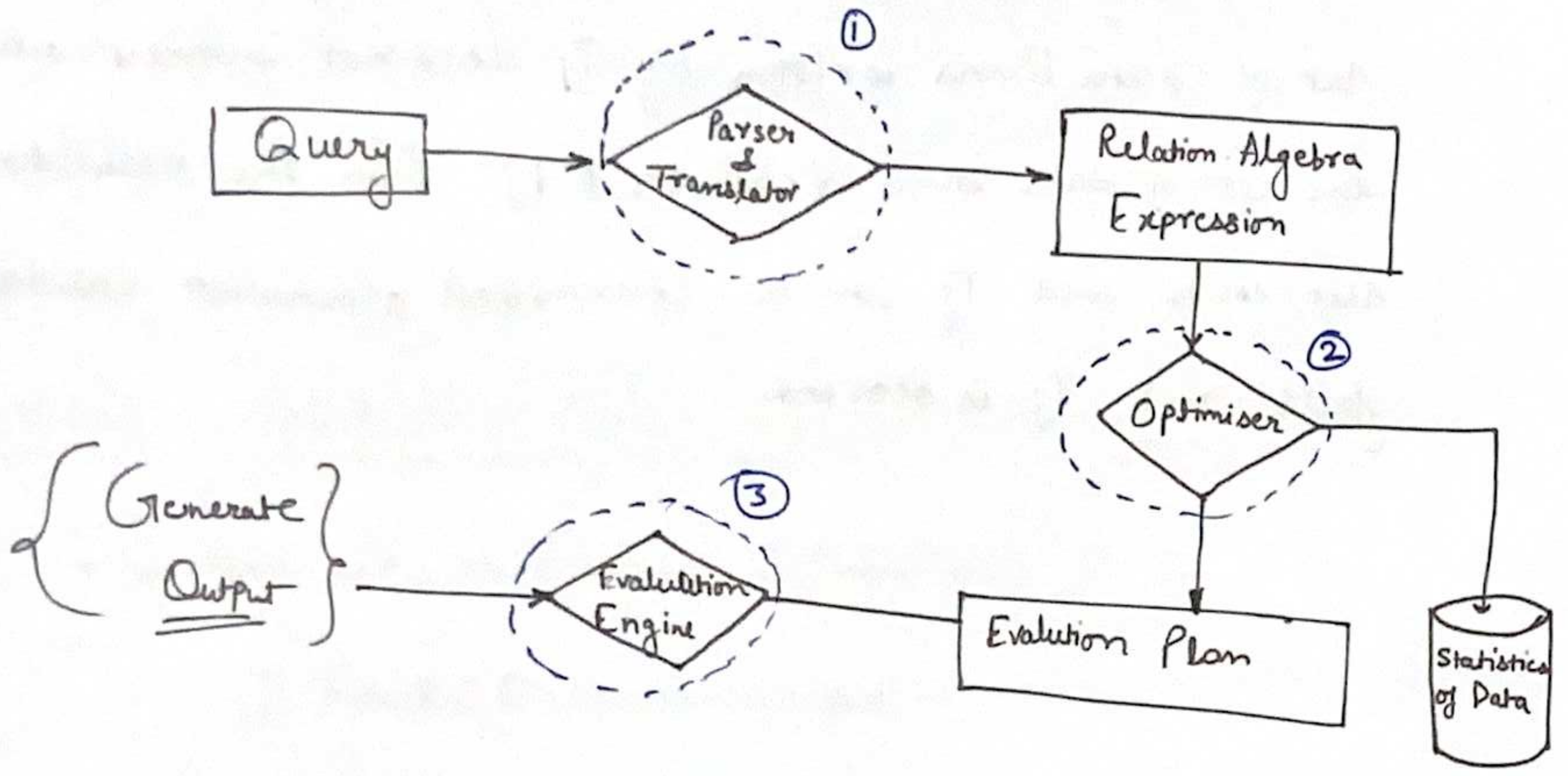
Query Processing

Three main things include :→

① Parsing & Translation

② Optimization

③ Evalution.



Parsing & translation → Converts query into ~~go~~ small query primitives & then convert it into an equivalent relational algebra expression.

→ To tell system "How" ~~of~~, as sql is a non-Procedural language

Optimizer → will see all alternative options of the query & will choose query which take least resources.

Equivalence Rules of Alternate Queries

① Conjuctive Selection Operations ⇒

$$\{ \sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E)) \}$$

eg

$$\sigma_{gender='m' \wedge salary>50000}(Emp) = \sigma_{gender='m'}(\sigma_{salary>50000}(Emp))$$

OR

$$\sigma_{salary>50000}(\sigma_{gender='m'}(Emp))$$

② Selection Operations are Commutative →

$$\{ \sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E)) \}$$

eg

$$\sigma_{gender='m'}(\sigma_{salary>50000}(Emp)) = \sigma_{salary>50000}(\sigma_{gender='m'}(Emp))$$

③ Only the final Operations in a sequence of projection operations are needed:→

$$\Pi_{L3}(\Pi_{L2}(\Pi_{L1}(E))) = \Pi_{L3}(E)$$

$$\{L3 \subseteq L2 \subseteq L1\}$$

eg ⇒

$$\pi_{eid}\left(\pi_{eid, ename}(Emp)\right) = \pi_{eid}(Emp)$$

④ Selections can be Combined with Cartesian Product & theta Join →

4.1 $\left\{ \sigma_{\Phi}(E_1 \times E_2) = (E_1 \bowtie_{\Phi} E_2) \right\}$

eg ⇒

Emp

eid	ename	dno
1	A	101
2	B	102
3	C	103
4	D	104
5	E	102

Dpt

did	dname
101	D1
102	D2
103	D3
104	D4

$$\sigma_{emp.dno = dpt.did}(Emp \times Dept) = (Emp) \bowtie_{emp.dno = dpt.did} (Dpt)$$

{Join based on these values}

4.2 $\left\{ \sigma_{\Phi_1}(E_1 \bowtie_{\Phi_2} E_2) = (E_1) \bowtie_{\Phi_1 \wedge \Phi_2} (E_2) \right\}$

5. Theta Join operations are Commutative →

$$\left\{ E_1 \bowtie_{\Phi} E_2 = E_2 \bowtie_{\Phi} E_1 \right\}$$

$\{(A+B)+C = A+(B+C)\}$

6. Natural Join Operations are Associative. →

6.1 $\left\{ (E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3) \right\}$

6.2 Theta Join are Associative in following Manner. →

$$(E_1 \bowtie_{\Phi_1} E_2) \bowtie_{\Phi_2 \wedge \Phi_3} (E_3)$$

$$= (E_1) \bowtie_{\Phi_1 \wedge \Phi_3} (E_2 \bowtie_{\Phi_2} E_3)$$

7. 7.1

$$\{\sigma_{\theta_1}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} E_2\}$$

7.2

$$\{\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} (\sigma_{\theta_2}(E_2))\}$$

eg ⇒

Emp

eid	ename	salary	dno
1	A	55000	101
2	B	60000	102
3	C	45000	103
4	D	48000	101
5	E	49000	102
6	F	52000	103
7	G	40000	101

Dept

did	dname
101	CS
102	IT
103	EC

θ = dno = did

θ_1 = salary > 50000

$\sigma_{\theta_1}(E_1 \bowtie_{\theta} E_2)$

$(E_1 \bowtie_{\theta} E_2)$

eid	ename	Salary	dname
1	A	55K	CS
2	B	60K	IT
3	C	45k	EC
4	D	48K	CS
5	E	49k	IT
6	F	52K	EC
7	G	40K	CS

eid	ename	salary	dname
1	A	55K	CS
2	B	60K	IT
6	F	52K	EC

$(\sigma_{\theta_1}(E_1)) \bowtie_{\theta} E_2$

$\sigma_{\theta_1}(E_1)$

eid	ename	salary	dno
1	A	55K	101
2	B	60K	102
6	F	52K	103

eid	ename	salary	dname
1	A	55K	CS
2	B	60K	IT
6	F	52K	EC

8. Set Operations union & Intersection are commutative →

8.1 $\{E_1 \cup E_2 = E_2 \cup E_1\}$

8.2 $\{E_1 \cap E_2 = E_2 \cap E_1\}$

Set difference is not commutative :→

$\{E_1 - E_2 \neq E_2 - E_1\}$

9. Set union & Intersection are Associative →

→ $\{E_1 \cup (E_2 \cup E_3) = (E_1 \cup E_2) \cup E_3\}$

→ $\{E_1 \cap (E_2 \cap E_3) = (E_1 \cap E_2) \cap E_3\}$

10 Distributivity among set operations :→

10.1

$$\left\{ \begin{array}{l} \{\sigma_p(E_1 - E_2) = \sigma_p(E_1) - \sigma_p(E_2)\} \\ \{\sigma_p(E_1 \cup E_2) = \sigma_p(E_1) \cup \sigma_p(E_2)\} \\ \{\sigma_p(E_1 \cap E_2) = \sigma_p(E_1) \cap \sigma_p(E_2)\} \end{array} \right\}$$

10.2

$$\{\sigma_p(E_1 - E_2) = \sigma_p(E_1) - E_2\}$$

Projection on union is distributed

11.

$$\{\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))\}$$

12

$$\{\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))\}$$

Projection with union of attributes is distributed over relations.

Attributes for Estimation of Cost ← Resources

n_r = The no. of tuples

b_r = no. of blocks containing tuple of relation r

l_r = The size of a tuple of relation r in bytes

f_r = the blocking factor of relation r that fits into one block.

$V(A,r)$ = no. of distint values apperias in relation r for attribute A.

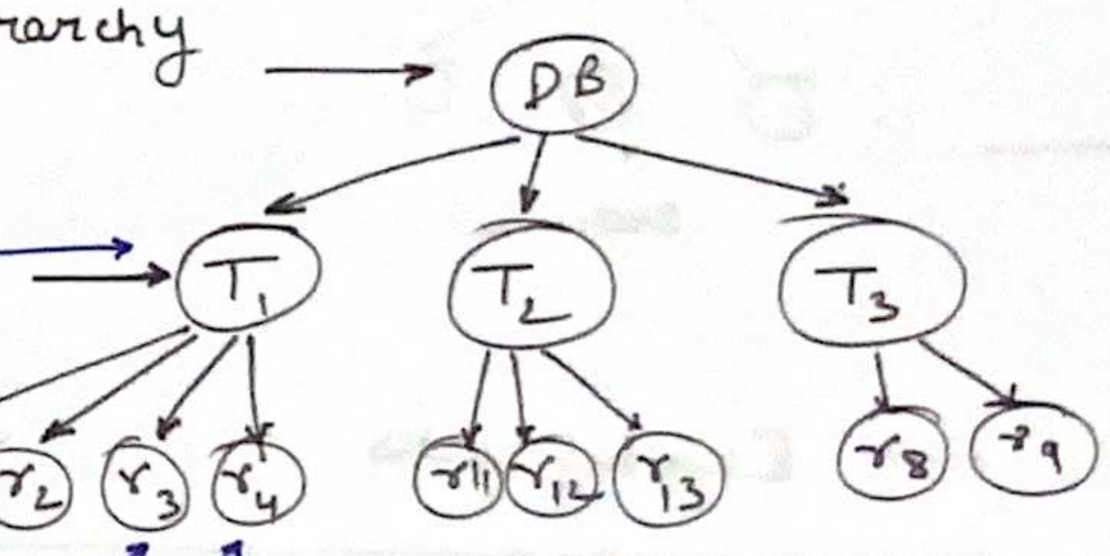
26/4/2024

Multiple Granularity {Concurrency Control}

Size of data items allowed to lock.

We lock on levels of Heierarchy → Multiple Granularity

eg Rather than applying Locks on idividual Rows we can apply it directly on Tables



Two types

→ Coarse Granularity → higher level like db or table

→ Fine Granularity → Lower level like rows or attribute.

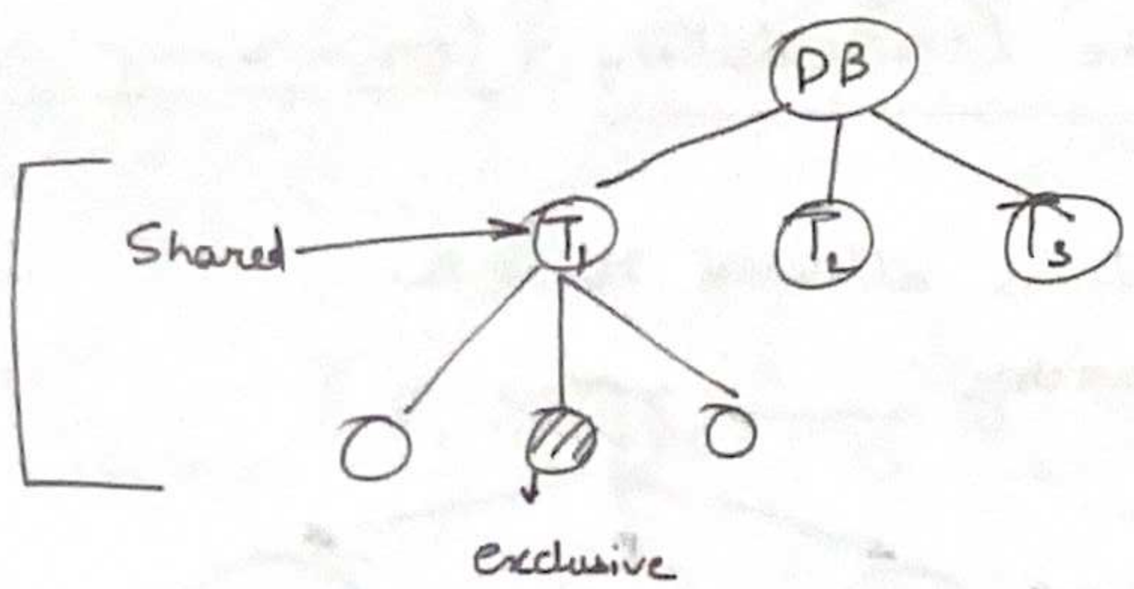
Locks in Case of Granularity

- ✓ → shared
- ✓ → Exclusive
- → Intension Shared (IS)
- → Intension Exclusive (IX)
- → Shared Intension Exclusive (SIX)

③ **IS Lock** →

If we are apply IS node on any node then it will be shared among its decendants as well.

SIX



→ Existing Locks

↓ Requested Lock

	IS	IX	S	SIX	X
IS	✓	✓	✓	✓	✗
IX	✓	✓	✗	✗	✗
S	✓	✗	✓	✗	✗
SIX	✓	✗	✗	✗	✗
X	✗	✗	✗	✗	✗

↑ Compatibility Matrix

Estimation of Cost {Not of Imp}

$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

→ Selection of size estimation →

$\sigma_{A=a}(r)$

Total no. of tuples $\frac{n_r}{V(A,r)}$

$\sigma_{A \le V}(R)$

$\min(A,r)$ $\quad$ $\max(A,r)$

$A \le V$ as 0 if $V < \min(A,r)$

n_r if $V \ge \max(A,r)$