Query Optimization

1. Conjunction of SELECT operations $\rightarrow$ cascade of selects
   $\sigma_{c1} AND \sigma_{c2} AND \cdots$
   $= \sigma_{c1}(\sigma_{c2}(\sigma_{c3}(\cdots)))$
2. Commutativity of SELECT
   $\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R))$
3. Cascade of Projections operations: only last projection step counts
   $\pi_x(\pi_y(\pi_z(\cdots))) = \pi_x(\cdots)$
4. Commutativity of σ, π
5. $\bowtie$ and X are commutative
6. Commutativity of σ with $\bowtie$ and X when σ condition applies to a single table involved in the $\bowtie$ or X
   $\sigma_{A[x]=\cdots} = (A \times B) = (\sigma_{A[x]=\cdots}(A) \times B)$
7. Commutativity of π with $\bowtie$ and X when join (or X) only involved attributes in projection
8. Associativity of $\bowtie, X$

Heuristic Algorithm for Optimizing Query Tree

1. Break up any conjunction of select conditions into a cascade
2.

Transactions

- Problem: multiple users accessing/modifying db at the same time
- Basic model:
  - Read an object
  - Write an object (e.g. blocks, rows, tables)

Concurrency Problems

1. Lost updates
2. Dirty read

Improve Model:

- Begin transaction
- Read/write
- Commit
- Rollback
- Goal: transaction treated as atomic unit of work ("all or nothing")
- Successful commit → all changes permanently written to database
- Rollback → all changes undone
- ACID Requirements
  - Atomicity
  - Consistency
  - Isolation
  - Durability

Serializability

- When we schedule concurrent transactions, the end result is same as if one followed another (serially)
- Managing transaction concurrency: locking, timestamp ordering, versioning

Locking

Two Phase Locking (2PL)

- Lock: a variable associated with an item descriving the state ofthe item with respect to the operations that can be performed on item
- Simple (binary locks): locked and unlocked
  - 2 states: locked and unlocked
  - Locked item is accessible only by the locker
  - All others have to wait
  - Too restrictive

Read/Write or Shared/Exclusive Locks

- 3 states: read-locked (share-locked); write-locked (exclusive-locked); unlocked
- A read (shared lock)
  - A locker may perform only reads
  - No other transaction can acquire a write-lock (must wait)
- Write (exclusive) lock: only locker can read or write the item

Require for Serializability

- Sufficient: require all locking operations in a transaction precedes any unlocking operations
-

Flavors of Locking

- Conservative: all locking done first, avoids deadlocks
- Strict: only releases write locks after entire transaction is committed or rolled back, does not avoid deadlocks

Disadvantages of Locking

- Overhead
- Deadlocks
- Starvation

Deadlocks

- T_1 locks X_1, requests Y
- T_2 locks Y_1, requests X

Prevention Protocols for Deadlocking

- Spurious aborts
- Deadlock detection
  - Build dependency graph
  - Look for cycles
  - Select a "victim"
- Timeouts

Timestamping

- Timestamp ordering

Multiversion

- Each transaction seeas a "snapshot of database
- In case of conflict, transaction may have to be rolled back or restarted