CS4224 Final Exam Cheat-sheet

1. Data Partitioning

1) *Desired fragmentation:* completeness, reconstruction & disjoint-ness.
2) *Fragmentation strategies:*
a. Horizontal fragmentation (also known as "*sharding*"): range partition, hash partition (modulo method, consistent hashing, with virtual nodes), primary horizontal partition, derived horizontal partition (need foreign key relationship).
b. Vertical fragmentation: use attribute affinity measure.
c. Hybrid fragmentation.
3) *Complete partitioning: F* is a complete partitioning of *R* with respect to *Q* if all tuples in each partition all belong or not belong to query result.
4) A min-term predicate writes all predicates in *P* in its conjunction form. Then, $MTPred(P)$ represents all min-terms predicates of *P*.
5) The min-term partitioning is always a complete partitioning.

2. Distributed Commit

1) *ACID properties:* atomicity, consistency, isolation & durability.
2) *Concurrency control manager:* ensure isolation according to isolation levels, schedule transactions, detect & prevent deadlocks.
2) *Recovery manager:* ensure atomicity & durability, maintains undo & redo log, always follow write-ahead logging (WAL) protocol and force at commit protocol.
3) *Distributed transaction:* the originating site acts as coordinator, others act as participants.
4) *Potential failures:* site failure (fail-stop, partial site failure, total site failure), communication failure.
5) *Two-phase commit (2PC) protocol:* prepare, vote commit (or abort), global commit (or abort), acknowledgment.
a. Every participant must reach the same global decision (commit/abort).
b. A participant cannot change its vote once it has voted.
c. A participant can abort before it votes (if it votes abort later).
d. Failures are detected by *timeouts*, handled by termination protocol and will use recovery protocol when failed site is restarted.
6) *Three-phase commit (3PC) protocol:* prepare, vote commit (or abort), prepare & ready to commit, global commit (or abort), acknowledgement.

3. Distributed Concurrency Control

1) *Different concurrency control strategies:* lock-based, timestamp-based, optimistic, multi-version and hybrid.
2) A schedule is *serializable* if it is equivalent to some serial schedule.
3) A schedule is *view serializable* if it is view equivalent to some serial schedule. Two schedules are view equivalent if read-from relationship (read initial value, read after write, write final value) remains the same.
4) A schedule if *conflict serializable* if it is conflict equivalent to some serial schedule. Two schedules are conflict equivalent if all pairs of conflicting actions are ordered in the same way.
5) Conflict serializable always implies view serializable.
6) To make a schedule *recoverable*, T_i must commit after T_j if T_i reads from T_j for all transactions in the schedule.
7) *Two-phase locking (2PL) protocol:* a transaction cannot acquire any lock anymore once it has released a lock.
8) *Strict two-phase locking (S2PL) protocol:* a transaction must hold all its locks until it commits (or aborts).
9) 2PL ensures conflict serializable, while S2PL ensures both conflict serializable and recoverable.
10) *Deadlock detection:* use waits-for graph (WFG). A deadlock is found if the WFG has a cycle.
11) *Deadlock prevention:* wait-die policy (T_i waits if T_i is older than T_j, otherwise T_i aborts), wound-wait policy (T_i waits if T_i is younger than T_j, otherwise T_j aborts).
12) *Different types of MVCC protocols:* MV 2PL, MV timestamp order, MV optimistic concurrency control, snapshot isolation.
13) A schedule is *multi-version view serializable* if it is multi-version view equivalent to a mono-version serial schedule. Two schedules are *multi-version view equivalent* if the read-from relationship remains the same. A schedule if *mono-version* if it always reads the most recent version. A schedule is *mono-version serial* if mono-version and serial.
14) View serializable always implies multi-version view serializable.

15) *Snapshot isolation:* every transaction sees a snapshot of the database that consists of updates by transactions committed before it starts.
a. Each write action creates a new version of the object.
b. Each read action reads its own update or the update done by the latest committed transaction before it starts.
c. Concurrent update policy: if multiple concurrent transactions try to update the same object, only 1 of them can commit. This policy can be enforced by first commit win (FCW) or first update win (FUW) rule.
d. Snapshot isolation does not guarantee serializability. It could lead to write-skew anomaly and read-only transaction anomaly.
e. Serializable snapshot isolation (SSI) protocol guarantees serializability.
f. To save space, garbage collection happens regularly to remove the old versions of objects which will not be referred by transactions anymore.
16) *Global schedule:* if local schedules are all its subsequence.
17) A global schedule is serializable if each local schedule is serializable and the local serialization orders are compatible.
18) *Distributed lock-based CC:* centralized 2PL and distributed 2PL.
19) *Distributed deadlock detection:* centralized approach (each site needs to maintain local WFG and periodically send to deadlock detector site), distributed approach (edge chasing algorithm).
20) *Distributed snapshot isolation:* centralized approach (one site to assign start & commit timestamp), distributed approach.

4. Data Replication

1) An execution is *one-copy serializable (1SR)* if equivalent to a serial execution on a one-copy database.
2) A replicated database is in its *mutually consistent* state if all copies of each object have identical values (*strong consistent*, *eventual consistent*).
3) *Replication method:*
a. System-level replication: statement-based replication, write-ahead log (WAL) shipping (file-based vs record-based).
b. Application-level replication.
4) *Different replication protocols:*
a. Eager or lazy: whether propagate updates to all replicas synchronously.
b. Centralized or distributed: where updates can occur (master vs all).
5) *Two eager centralized protocols:* eager single master (master acts as centralized lock manager, read one write all), eager primary copy (each object has 1 master copy, each site is the centralized lock manager for its master copies, generalization of single master).
6) *Eager distributed protocol:* each site as lock manager.
7) *Lazy centralized protocol:* lazy single master (use refresh transaction to update other copies after commit).
8) *Lazy distributed protocol:* each site as lock manager and use refresh transaction to update other copies after commit.
9) *Inconsistent update reconciliation:* last writer win heuristic (but only works for blind-write updates).
10) Failure handling for slave sites:
a. Eager centralized protocol: use read one write all available to relax.
b. Lazy centralized protocol: synchronize when the failed site is back.
11) *Failure handling for master sites:* to elect a new master site, simple or majority consensus or quorum consensus algorithm can be used.
12) *Quorum consensus (QC) protocol:* read threshold & write threshold, which makes $T_r + T_w > Wt$ and $2 \cdot T_w > Wt$ holds. For each read / write action, its read / write quorum ≥ read / write threshold.

5. Consensus (Raft)

1) *Three roles:* follower (passive), candidate (issue Request-Vote RPC to get elected), leader (issue Append-Entry RPC).
2) *Three timers:* election timer, leader timer, client timer.
3) *Two properties for election:* election safety, election liveness.

6. Data Consistency

1) *Consistency model:* strong consistency, consistent prefix, bounded staleness, monotonic reads, read my writes, casual, eventual consistency.
2) *Microsoft Pileus:* a key-value database, uses lazy centralized replicate protocol and distributed snapshot isolation for concurrency control.

7. Query Processing & Execution

1) *Distributed join strategy:* collocate, directed, broadcast, repartition.
2) Semi-joins can eliminate dangling tuples and reduce comm cost.
3) A full reducer exists for a query if and only if its hypergraph is acyclic.

--- End ---