Z2004: Database Management Systems Master Sheet

ER Modeling, Relational Algebra, SQL, Normalization, B+ Trees & ACID · innocent@iitmz.ac.in

1. ER-to-Relational Mapping Rules

- **Strong Entity E:** Create table E with simple attributes and primary key (PK).
- **Weak Entity W (owner E):** Table W adds PK(E) as FK; PK(W) = PK(E) ∪ discriminator; use `ON DELETE CASCADE`.
- **1:N Relationship:** Place PK of the 1-side as FK on the N-side table.
- **M:N Relationship:** Create junction table with (PK(A), PK(B)) as composite PK + relationship attributes.
- **EER Generalization:** (a) Single table + type tag (NULLs), (b) Table per subclass + PK FK to superclass, (c) Table per concrete class.

2. Relational Algebra Core & Derived Operators

Op	Meaning	Notes
$\sigma_p(R)$	Selection (rows)	Filters tuples satisfying p
$\pi_A(R)$	Projection (cols)	Set-based; eliminates duplicates
$R \bowtie_p S$	Theta / Natural join	$\sigma_p(R \times S)$
$R \cap S$	Intersection	Derived: $R - (R - S)$
$R \div S$	Division ("for all")	$\pi_{R-S}(R) - \pi_{R-S}((\pi_{R-S}(R) \times S) - R)$

3. SQL Evaluation Order & 3-Valued Logic

- **Logical Order:** `FROM/JOIN` → `WHERE` → `GROUP BY` → `HAVING` → `SELECT` / Window Functions → `DISTINCT` → `ORDER BY` → `LIMIT`.
- **Three-Valued Logic:** `NULL = NULL` yields `UNKNOWN`; `WHERE` keeps only `TRUE`. Always write `IS NULL` / `IS NOT NULL`.
- **Window Ranking:** `ROW_NUMBER()` $(1, 2, 3, 4)$, `RANK()` $(1, 2, 2, 4)$, `DENSE_RANK()` $(1, 2, 2, 3)$.
- **Moving Frame:** `ROWS BETWEEN 2 PRECEDING AND CURRENT ROW`.

4. Functional Dependencies & Normal Forms

Armstrong's Axioms: Reflexivity ($Y \subseteq X \Rightarrow X \to Y$), Augmentation ($X \to Y \Rightarrow XZ \to YZ$), Transitivity ($X \to Y, Y \to Z \Rightarrow X \to Z$).

- **Attribute Closure X^+:** Start $C = X$; repeatedly add B if $A \to B \in F$ and $A \subseteq C$. K is a superkey iff $K^+ = R$.
- **1NF:** Atomic domains (no repeating groups/nested arrays).
- **2NF:** 1NF + no partial dependency (non-prime attribute depends on proper subset of a candidate key).
- **3NF:** Every non-trivial $X \to A$ has X superkey OR A prime. Always lossless + dependency-preserving via canonical cover F_c.
- **BCNF:** Every non-trivial $X \to A$ has X superkey. Always lossless; may sacrifice dependency preservation.
- **Lossless Join Test (R_1, R_2):** $(R_1 \cap R_2)^+ \supseteq R_1$ OR $(R_1 \cap R_2)^+ \supseteq R_2$.

5. B+ Tree Storage & Join Cost Formulas

- **B+ Tree Capacity:** Max fanout $F = \lfloor B/(k+p) \rfloor$; tree height $h \approx \lceil \log_{F_{\text{avg}}} N_{\text{leaf}} \rceil$. Range scan of K keys: $h + \lceil K/\text{keys_per_leaf} \rceil$ I/Os.
- **Block Nested-Loop Join (M buffer pages):** I/O = $B(R) + \lceil B(R)/(M-2) \rceil \cdot B(S)$ (put smaller R as outer).
- **Two-Pass Sort-Merge Join:** I/O = $3(B(R)+B(S))$ when $M > \sqrt{\max(B(R), B(S))}$.
- **Grace Hash Join:** I/O = $3(B(R) + B(S))$ when $M > \sqrt{\min(B(R), B(S))}$.

6. ACID, Isolation Levels & Serializability

SQL Isolation Level	Dirty Read	Non-Repeat.	Phantom
Read Uncommitted	Possible	Possible	Possible
Read Committed	Prevented	Possible	Possible
Repeatable Read	Prevented	Prevented	Possible*
Serializable	Prevented	Prevented	Prevented

Conflict Serializability: Build precedence graph with edge $T_i \to T_j$ if T_i precedes T_j on the same item and at least one operation is a write. Schedule is conflict-serializable iff graph is acyclic.

7. Strict 2PL & ARIES Crash Recovery

- **Strict 2PL:** Acquire shared (S) / exclusive (X) locks before access; hold all X locks until `COMMIT`/`ABORT` (avoids cascading aborts; deadlocks resolved via wait-for graph cycle detection).
- **Steal / No-Force Policy:** Buffer manager may evict dirty pages before commit (Steal ⇒ requires Undo log) and need not flush all dirty pages at commit (No-Force ⇒ requires Redo log).
- **ARIES 3 Phases:** (1) **Analysis:** scan forward from last checkpoint to reconstruct Transaction Table (TT) and Dirty Page Table (DPT); (2) **Redo:** repeat history forward from smallest recLSN in DPT; (3) **Undo:** roll back uncommitted loser transactions backwards writing CLRs.

8. CAP Theorem, NoSQL & RAG Vector Search

- **CAP:** Under network partition (P), choose Consistency (CP: HBase, MongoDB majority) or Availability (AP: Cassandra, Dynamo).
- **Polyglot Choice:** Relational (ACID ledgers), Document (nested profiles), Key-Value (session cache), Vector (dense embeddings via HNSW/IVF + cosine similarity $\frac{q \cdot d}{\|q\|\|d\|}$ for RAG grounding).

9. Worked SQL & Normalization Checklist

- **Outer-Join Filter Rule:** Place predicates on the nullable inner table inside `ON` rather than `WHERE`, otherwise `NULL` rows turn into `UNKNOWN` and vanish.
- **BCNF Algorithm:** If $X \to Y$ violates BCNF (X not a superkey, $Y \not\subseteq X$), decompose R into $R_1 = X^+$ and $R_2 = (R \setminus X^+) \cup X$; repeat until every table satisfies BCNF.

- **3NF Synthesis:** Compute canonical cover F_c, create one table XA per FD $X \to A \in F_c$, and add one candidate key table if no created table contains a candidate key of R.
- **Index Column Order:** In composite B+ tree index `(a, b)`, equality predicates on `a` plus range predicates on `b` use the index prefix; range on `a` stops index pruning for `b`.

10. Essential SQL Window & CTE Patterns

- **Top-k per Group:** Wrap `DENSE_RANK() OVER (PARTITION BY dept ORDER BY salary DESC) AS rk` inside a `WITH ranked AS (...)` CTE, then filter `WHERE rk <= 2`.
- **Recursive Hierarchy Walk:** `WITH RECURSIVE Ancestors(id, mgr) AS (SELECT id, mgr FROM Emp WHERE id = :target UNION ALL SELECT e.id, e.mgr FROM Emp e JOIN Ancestors a ON e.id = a.mgr)`.
- **Anti-Join Pattern:** Prefer `NOT EXISTS (SELECT 1 FROM Orders o WHERE o.cid = c.cid)` over `NOT IN (...)` whenever the subquery column may contain `NULL`.