CS2102 Notes.

Introduction

Date No.

ACID Principles.

Atomicity, Consistency (no erroneous state)
Isolation (as if executions are sequential), Durability (lasting effects of executions).

Equivalent Execution

Two executions are equivalent if they have the same effect on data.

3-Tier Architecture of DBMS.

- External, logical, physical

Domain

A set of atomic values that an attribute can be, e.g. INT, TEXT. For a value of attribute A_i, $v \in dom(A_i)$ or $v = NULL$.

Relation.

A set of tuples. A relation is an instance of a schema, that defines a list of attributes and their domains, usually denoted as $R(A_1, \ldots, A_n)$.

Relation Database.

A collection of tables (relations)

Integrity Constraints

- Structural: key, foreign key, domain
- General: check, triggers.

Superkey: subset of attributes that uniquely identifies a tuple.

(Candidate) key: a minimal superkey.

Primary key cannot be NULL.

Date No.

Foreign key: either contains NULL or appears as the primary key in the referenced relation

RA.

selection (σ): $\sigma_{[condition]} R$

projection (π): $\pi_{[col_1, col_2]} R$: no duplicate cols, order matters, remove duplicate rows after π.

renaming (ρ): ① $\rho_{[B_i \leftarrow A_i]} R$ ② $\rho_{(B_1, B_2, B_3 \ldots)}(R)$ just renaming, reorder attributes. ⇒ need to write down all attributes.

Three-Valued Logic.
AND: True ∧ NULL ∧ False → { If there is a False → False; No False, but a NULL → NULL; No False or NULL → True }
OR: True ∨ NULL ∨ False.

(NULL >, <, ≤, ≥, =, <>, + − × ÷ A) => NULL.
However, NULL ≡ NULL ⇒ True, NULL ≡ A ⇒ False.
A ≡ B ⇒ A = B.

"Union Compatible": R and S have the same number of attributes and each corresponding attribute has the same or compatible domains.

Cross Product: $R \times S$: R and S, unlike SQL, must be disjoint.

JOIN:
Inner {
θ-Join: $R \bowtie_{[\theta]} S = \sigma_{[\theta]}(R \times S)$.
Equi-Join: θ is based on equality (= or ≡)
Natural Join: $R \bowtie S \Rightarrow R(A_1, \ldots, A_i, B_1, \ldots, B_j, C_1, \ldots, C_k)$
}
order sometimes matters; only ∈ R; ∈ R ∩ S; only ∈ S.
those ∈ R first; those ∉ R next.

Date No.

Outer
- Left Join : $\bowtie_{[\theta]}$ Include dangling tuples from left.
- Right Join $\bowtie_{[\theta]}$ ... ... right
- Full Join $\bowtie_{[\theta]}$ ... ... both sides
- Pad missing values with NULL
- May or may not be natural.

Strong VS Weak Equivalence.

A SE B $\Leftrightarrow$ (A and B produce error) or the same result

A WE B $\Leftrightarrow$ (A or B produces error) or the same result

Principle of Acceptance.
- Perform only when the condition is True.
- Used in WHERE clause and RA. and PL/... IF statment

Principle of Rejection.
- Reject only when the condition is False (perform otherwise)
- Used in integrity constraints.

SQL.

DDL Data Definition
CREATE, ALTER, DROP, ADD

DML Data Manipulation
INSERT, UPDATE, DELETE

DQL Data Query
SELECT

DCL Data Control (not covered).

TCL Transaction Control
BEGIN, COMMIT, ROLLBACK.

SQL is case-insensitive / INSERT operation is atomic if there are multiple rows. Missing values are replaced with NULL or default value. / Principle of Acceptance is applied to DELETE (delete if true). / Use CONSTRAINT name to name a constraint.

Use Principle of Rejection to check UNIQUE constraint
UNIQUE ⇔ <> evaluates to true or NULL

Foreign Key Constraint. → Set on the referencing table.
NO ACTION, RESTRICT. (= NO ACTION + non-deferred check), CASCADE, SET DEFAULT (the default value must be there on the referenced table), SET NULL.

INITIALLY IMMEDIATE is always IMMEDIATE if the transaction does not set it to DEFERRED. Even if it does, the effect only lasts within the scope of it.

Query. → at most one NULL returned.
SELECT [DISTINCT] <attr-list> FROM <rel_list>
WHERE <conditions>.

Operators: |/ : sqrt, || concat.

Pattern Matching: _ : any char % : string of 0 or more char

SET operations: ALL keeps duplicates: every value has a count. Without ALL, normal set operations are done. UNION/INTERSECT/EXCEPT.

JOIN without NATURAL, whether INNER or OUTER, should be accompanied by ON [conditions].

Universal Schema Assumption.
All tables can be represented by a (possibly very wide) table. ⇒ May not always hold.

can be replaced by inner join → IN: (∃ row ∈ <subquery> : row = <expr>) evaluates to True.
(row: selected · <expr>: cannot be NULL)
outer join → NOT IN: (∀ row : row <> <expr>) evaluates to True.
∴ It is possible that both return nothing. (esp. when there is NULL in <subquery>)

EXISTS: <subquery> returns at least one row
NOT EXISTS: otherwise.

~~In the jargon of Querying, "known" → NOT NULL.~~

Date No.

ANY: ∃ row; <expr> <op> row = True. } <subquery> must
ALL: ∀ row; <expr> <op> row = True. } have only one col.

IN ≡ (=ANY) / <e1><op>ANY <Q> ≡ EXISTS (··· AND <e1> <op> <e2>).

NOT IN ≡ (<> ALL).

If duplicate removal is needed for SELECT, then the attr by which the table is sorted must be also in the SELECT clause

*Smaller lexicographic order has higher rank → TEXT ↘ ASC

Aggregation Functions.
- Only applicable to non-NULL values, except *.
- Can have DISTINCT options.
- If all values are NULL, MAX(A) ⇒ NULL, COUNT(A) ⇒ 0.
  - COUNT(*) ⇒ n.

Grouping.
- t and t' will be in the same group if
  $(\bigwedge_i t.a_i$ IS NOT DISTINCT FROM $t'.a_i)$

- Restrictions: If A∈R is in the SELECT clause (HAVING) then.
  1. A is in the GROUP BY clause, or
  2. A is one input for aggregation function in the SELECT (HAVING) clause, or
  3. The primary key of R is in the GROUP BY clause.

HAVING.
- Similar to WHERE, both using PoA.
- Restrictions: just the same as above except the blue part.

Evaluation Order.
From → WHERE → GROUP BY → HAVING → SELECT → ORDER BY → LIMIT/OFFSET.

CASE
  WHEN <ex> THEN <val>
  ELSE <default>
END.

CASE <ex>
  WHEN <result> THEN <val>
  ELSE <default>
END.

COALESCE → return the first non-NULL value.

$$NULLIF(v,u) = \begin{cases} NULL, & (v=u) \Rightarrow True. \\ v, & otherwise. \end{cases}$$

Common Table Expressions (At most 2 in exams!)

```
WITH [(attr1, ..., attrn)],
    CTE-1 AS (Q-1) ...
  Q.
```

View:

```
CREATE VIEW <name> AS (Q).
```

Universal Quantification.

- Double Negation Solution.
  $\forall x: P(x) \Leftrightarrow \neg\neg \forall x: P(x) \Leftrightarrow \neg \exists x: \neg P(x)$.
- Cardinality Solution.
  Use set operations and COUNT([DISTINCT] A / *). ("must" / or)

Recursive Query.

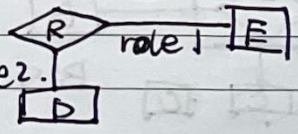
- WITH RECURSIVE ... L AS (
  ... count
  UNION [ALL] SELECT ... count+1 FROM L.
  ...).
- Can have lazy evaluation: move the termination condition from CTE to the main query. (WHERE count < a)

ER.

Entity sets ▭ cannot be proper nouns.

Attributes: ① key (attr) ② composite (A: B, C) ③ multivalued (A) ④ derived (A)

Relationship sets

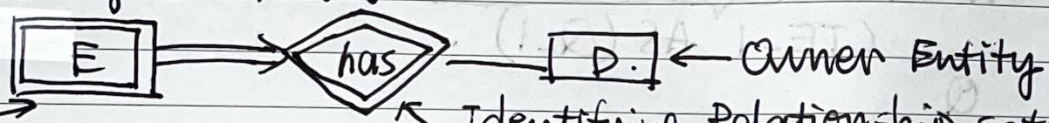


Relationship Constraints.

- Cardinality Constraint: many/one-to-many/one.
  - Key Constraint: at most one relationship per entity. →
- Participation Constraint:
  - ~~Partial Participation: lower limit = 0.~~
  - Total Participation: lower limit = 1.

Key + Total Participation ⇒
One and only one relationship instance per entity.

Dependency Constraint.

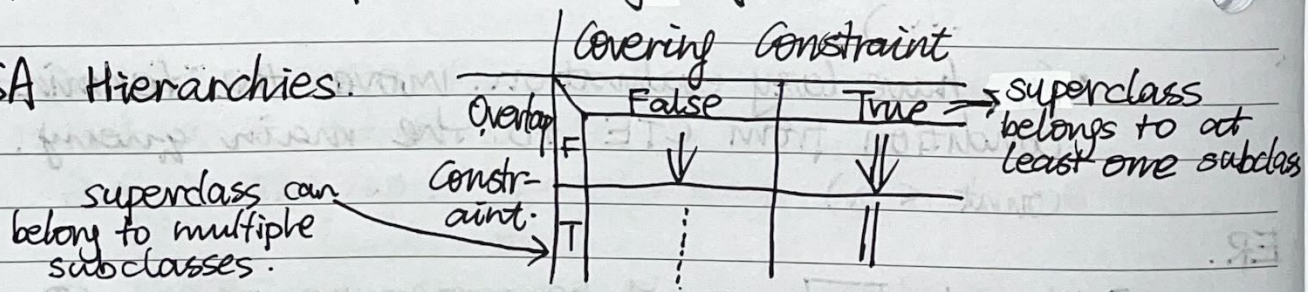


Weak Entity Sets
Only have partial keys.

Multivalued Attributes Mapping.
① Split to columns ⇒ Limited number of possible values.
② A separate table ⇒ Query 2 tables at a time.

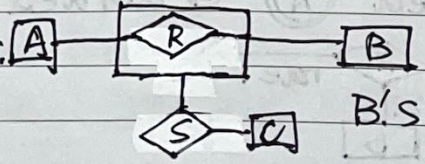
ER Mapping.
Many-to-Many: Set a table with 2 foreign keys.
Many-to-one: The "one" table references the "many" table
One-to-one: Both tables have a unique FK (but no "NOT NULL").
Key + Total: set the foreign key to be UNIQUE and NOT NULL
Weak Entities: plus the key of the owner.

ISA Hierarchies.



- Remember to do ON DELETE CASCADE in subclasses.

Aggregation:



For mapping, use A's and B's keys in S.

SQL Programming.

Dynamic SQL:

EXECSQL
- PREPARE var FROM :stmt;
- EXECUTE var USING para1, para2;// ← parameters passed.
- DEALLOCATE PREPARE var;

Date No.

Function

CREATE OR REPLACE FUNCTION func([IN/OUT] p1, [IN/OUT] p2 ...)
RETURNS TYPE/TABLE_NAME/SETOF TYPE/SETOF RECORD/ TABLE(attr1, attr2 ...)/TRIGGER AS $$.
DECLARE ... ... ;
BEGIN. ... ... ;
END; $$ LANGUAGE plpgsql/c/sql/...;

(one value; a tuple; a set of tuples)

Use CALL for procedures and SELECT for functions.

Cursor

curs CURSOR FOR(SELECT ...); OPEN curs; LOOP.
FETCH curs INTO r; EXIT WHEN NOT FOUND;
[RETURN NEXT] /* not curs' next*/; END LOOP; CLOSE curs;

has something to do with the function signature.

※ FETCH [PRIOR/FIRST/LAST/ABSOLUTE n] FROM cur INTO r;

SQL Injection Attacks

- Malicious instructions are inserted as input.
- Approach ①: Use function/procedure to isolate the input from the execution. } the input is regarded as a string constant.
- Approach ②: PREPARE

Trigger

CREATE (can be OR) TRIGGER name AFTER/BEFORE/INSTEAD OF INSERT/UPDATE/DELETE ON tname FOR EACH ROW/STATEMENT [WHEN (cond)] EXECUTE FUNCTION func() [DEFERRABLE ...].

Special variables a trigger can access:
- TG_OP: the operation that triggers
- TG_TABLE_NAME.
- OLD / NEW.

Return Values.

	Has effect			Has no effect		
	INSERT	UPDATE	DELETE	INSERT	UPDATE	DELETE
BEFORE	non-NULL NEW	non-NULL NEW	non-NULL OLD.	OLD/NULL	OLD/NULL	NULL.
AFTER	NO			Any		

INSTEAD OF
- Instead of doing something on a VIEW, do it somewhere else, usually the corresponding table.
- E.g., instead of modifying the current max-score VIEW, update the table directly.
- Always row-level.; • Return NULL to skip the original op.

Statement Level.
- No access to NEW, OLD...
- Ignores return values.
- Raise an exception to skip subsequent operations.
- Can be BEFORE.

Deferred Trigger.
- Only works with AFTER and FOR EACH ROW
- Activated by COMMIT.

Multiple Triggers.
- Order of activation
  1. BEFORE statement-level
  2. BEFORE row-level
  3. AFTER row-level
  4. AFTER statement-level

  } like a stack. break ties by alphabetical order
- If a BEFORE row-level returns NULL, then all subsequent triggers are skipped.

Normal Forms
- Reduces data redundancy and improves data integrity.

Redundancy → Anomalies.
- Update anomaly: partial update that does not cover all rows supposed to be updated.
- Insertion anomaly: inability to add data due to absence of other data, e.g., fk or pk.
- Deletion anomaly: cannot delete a data because it contains something that cannot be deleted, e.g., pk.

Functional Dependencies.
- $A \to B \iff$ if A is the same, then B is the same.
- An FD may hold on table A but not on B.
- Huang's Theorem: Let M, N be the set of rows uniquely id-able by $A \& \{A, B\}$, resp., then $(A \to B) \Rightarrow M = N$.

Armstrong's Axioms.
- Reflexivity $AB \to A$
- Augmentation $(A \to B) \Rightarrow (AC \to BC)$.
- Transitivity $(A \to B \wedge B \to C) \Rightarrow (A \to C)$.

Additional Rules

Date No.

Rule of Decomposition: $(A \to BC) \Rightarrow (A \to B \wedge A \to C)$

Rule of Union: $(A \to B \wedge A \to C) \Rightarrow (A \to BC)$

Closure

- $\{A\}^+ = \{ABC\ldots\}$ = the set of attributes that can be determined by A, directly or indirectly.
- Traverse every unused FD until no new attribute is added in a loop.

Finding Keys.

- List out all possible combinations of attributes and their closures → If a closure is complete ⇒ superkey
  - Trick ①: attributes that do not appear in any RHS are definitely part of the key.
  - Trick ②: try small attributes first. If a set of attributes is a superkey, so is its superset.
  - Trick ③: if $\{AB\}$ is a key, then other keys do not completely contain $\{AB\}$ ⇒ stop checking a set if it contains $\{AB\}$.

Prime Attribute:

- An attribute is a prime attribute if it appears in a key.

※ Soundness & Completeness.

- Sound ⇔ Everything provable is true.
- Complete ⇔ Everything true has a proof.

Non-Trivial & Decomposed FD.

- Non-trivial: RHS does not appear in LHS.
- Decomposed: RHS only has one attribute.
- Find them using closures

BCNF

- A table is in BCNF if every non-trivial and decomposed FD has a superkey as its LHS.

BCNF intuition:
- Every attribute should depend only on superkeys.
- Any dependancy on non-superkeys is prohibited.

Tricks: For a table of n attributes, only need to check "more-but-not-all" closures of $\leq n-2$ attributes

Date No.

ok to consider explicit FDs only.

BCNF Algorithm.
① Check the closure of each attribute subset.
② Check if there is a "more-but-not-all" closure.
③ Start from smaller sets. Skip sets that contain a confirmed superkey.

BCNF Decomposition.
① Input a table R. Suppose $\{X\}^+ \rightarrow \{X\}^+$ is a "more-but-not-all" closure.
② Split R into $R_1 = \{X\}^+$ → no repeated rows. and $R_2 = \{X\} \cup (R - \{X\}^+)$
③ Further check R_1 and R_2.
- Trick: If there are multiple choice of $\{X\}$, pick the smallest $\{X\}^+$. If $|R_1| = 2$, then R_1 must be in BCNF

FD Projection on R_i → can have multiple attributes.
① Enumerate all attribute subsets on R_i
② Derive the closures on R.
③ Remove attributes in the closure but not on R_i
- If there is still a "more-but-not-all" closure, then keep decomposing.
- If the union of all projected FDs (using closure!!) is equivalent to the original set of FDs, then we say that the decomposition is FD-preserving.

Properties of BCNF:
Good: No anomalies (update/insert/delete), small redundancy, reconstructable
Bad: Not FD-preserving.

Lossless Join: $R_1 \bowtie R_2 = R$. Lossy Join $R \subsetneq R_1 \bowtie R_2$
A decomposition is lossless iff the full set of common attributes in R_1 and R_2 constitute a superkey of R_1 or R_2.

Two sets of FDs S, S' are equivalent iff:
Every FD $\in S$ can be derived from S' and the other way around. → Using closure is efficient. S and S' should have the same closure.

Properties of 3NF: Lossless join, FD-preserving, small redundancy.

3NF: Every non-trivial and decomposed FD: LHS is a superkey or RHS is a prime attribute.

BCNF is stricter ⇒ (BCNF ⇒ 3NF but 3NF ⇏ BCNF).

3NF Check: For each given FD, check LHS & RHS.

3NF Decomposition Algorithm / 3NF Synthesis
1. Derive a minimal basis.
2. Combine FDs whose LHSs are the same.
3. Create a table for each FD remained.
4. If none of the tables contains a key, then create a table that contains a key → Ensures lossless join
5. Remove redundant tables.

Minimal Basis/Cover, M, for a set of FDs, S.
- M and S are equivalent.
- M contains only non-trivial and decomposed FD.
- If any FD is removed from M, then M cannot derive S. (No redundant FD in M)
- For every FD∈M, if we remove an attribute from LHS, then it cannot be derived from S.

Algorithm to get Minimal Basis
1. Transform each FD such that each RHS contains only one attribute.
2. Remove redundant attribute from each FD. E.g. AB→C. If A→C is implied by M, then remove B. (the current working set.)
3. Remove redundant FD if it can be derived from the rest of M. → Again, use closures.

Appendix.

Non-Trivial FDs:
$A \to B$ if $B \not\subseteq A$. This means $\{A,B\} \to \{A\}$ is trivial
Completely non-trivial if $A \cap B = \emptyset$

Armstrong's Axioms Tricks.
- $A \to B$, $A \to C$; $A \to AB \to A \to AC \Rightarrow ABB \to ABC \to BC$
  $\therefore A \to BC$.
- Augment given FDs to save steps. (obtained.)