Relational Algebra

- Definitions
  - ↳ attribute => cols. in a relation
  - ↳ tuples => rows in a relation
  - ↳ domain => type of an attr
  - ↳ schema => the structure of a relation
  - ↳ instance => content of a relation
  - ↳ key => attrs. that uniquely identify a tuple
- Uses SET SEMANTICS
  - ↳ No dupes + no ordering
- Operators
  - ↳ $\sigma_c R$
  - ↳ $\pi_A R$ => ~~$\pi_A(\pi_B R)$~~
  - ↳ $R \times S$
    - ↳ 1 output per pair of input tuples
    - ↳ SET SEMANTICS
  - ↳ $R \bowtie S$
    - ↳ Not case, generalizes into $\times$
    - ↳ $R \bowtie R = R$
  - ↳ $\rho_s R$ => rename
  - ↳ $R \cup S$, $R - S$, $R \cap S$
    - ↳ Should have same schema
    - ↳ $\cap$ not core => $R \cap S = R - (R - S)$

- Data modification
  - ↳ INSERT INTO <schema> VALUES (<val> | <query>)
  - ↳ DELETE FROM <schema> WHERE <cond>
  - ↳ UPDATE <schema> SET A=An, ... WHERE <cond>
- Operators
  - ↳ SELECT
    - ↳ + DISTINCT
  - ↳ FROM
  - ↳ WHERE
  - ↳ UNION, INTERSECT, EXCEPT
    - ↳ SET SEMANTICS
    - ↳ +ALL for bag semantics
  - ↳ NOT IN, IN
  - ↳ ALL, SOME
    - ↳ Ex: a > ALL R
- Subqueries
  - ↳ Nested SELECT
  - ↳ 1×1 => scalar
- Correlated subquery
  - ↳ References a table in the outer clause
  - ↳ EXISTS ( )
- Common Table Expr.
  - ↳ WITH <alias> AS <subquery>
  - ...

- ORDER BY / FETCH FIRST
  - ↳ ORDER BY <attr> ASC / DESC
  - ↳ <offset> FETCH FIRST <num> ROWS ONLY
    - ↳ Changes bag semantics
    - ↳ LIMIT <num> OFFSET <offset>
- General SELECT
  - ↳ SELECT ...
    FROM ...
    WHERE ...
    GROUP BY ...
    HAVING ...
    ORDER BY ...
    FETCH FIRST ...
- NULLs
  - ↳ NULL + arithmetic ops = NULL
    - ↳ DIV by 0 = NULL
  - ↳ NULL ≠ NULL
    - ↳ Logical ops w/ NULL = Unknown
  - ↳ Aggregates ignore tuples w/ NULL
    - ↳ Not COUNT(*)
    - ↳ Agg(NULL) = NULL
      - ↳ COUNT() = 0
  - ↳ Set ops treat NULL normally
  - ↳ IS NULL or IS NOT NULL
  - ↳ COALESCE() => 1st non-NULL

- Outer join
  - ↳ preserves dangling tuples
  - ↳ LEFT or RIGHT
- Recursion
  - ↳ With RECURSIVE ancestor(child, ancestor) AS (SELECT * FROM Parent) UNION (SELECT P.child, A.ancestor FROM Parent P, Ancestor A WHERE P.parent = A.child))
    SELECT ancestor FROM Ancestor WHERE child = 'Sue';

Basic SQL

- Types
  - ↳ CHAR(N)
  - ↳ VARCHAR(N)
  - ↳ INT
  - ↳ DECIMAL(N,M)
    - ↳ N sig figs, M after dec.
  - ↳ REAL
  - ↳ DOUBLE
  - ↳ TIMESTAMP
- TABLE creation
  - ↳ CREATE TABLE <name domain, ...>
    - ↳ PRIMARY KEY (attr) => can't be NULL
    - ↳ UNIQUE (attr)
    - ↳ DEFAULT <val>
  - ↳ DROP TABLE <schema>

Advanced SQL

- Aggregate func.
  - ↳ AVG, SUM, COUNT, MIN, MAX
  - ↳ results => single tuple
  - ↳ shouldn't appear in WHERE
  - ↳ GROUP BY: turns groups, where a single tuple exists for each unique attrs
    - ↳ SELECT has Agg. Func or unique attrs. only
  - ↳ HAVING: checks aggregate conds.
- Window Func
  - ↳ FUNCTION (attr) OVER ( )
  - ↳ Applies agg. over all inputs
  - ↳ PARTITION BY: agg. over partition
- Case Expr.
  - ↳ CASE <attr> WHEN <cond.> THEN <expr> ELSE <expr> END

· E/R Model

↳ 2 sets: entity and relationship

↳ entity:

↳ relationship:

↳ Cardinality

↳ # entities participate

↳ 1-to-1, 1-to-many, many-to-many

↳ arrow on "1" side

↳ total participation: =

↳ General notation

↳ a..b, * = infinite

· Roles: designated to edges from an entity set to label the role of that entity

· Superclass and Subclass

↳ Specialization and Generalization

↳ Subclass inherits all attr of superclass

↳ Subclass participates in the relationships of the superclass

↳ Subclass may participate in its own relationship

↳ Disjoint specialization vs. overlapping specialization

↳ Either-or vs. multiple specialization

↳ 

· Weak Entity Set

↳ not enough attributes

↳

↳ key comes from linked entity sets

↳ identifying relationship: weak entity set and owner

↳

Normalization

· FD

↳ u[x] => u[sid, name] => (100, Jones)

↳ X -> Y => no tuple can have $X_1 = X_2$, but $Y_1 \neq Y_2$

↳ Trivial (sid) -> sid

. Logical implication

↳ A -> B, B -> C => A -> C

· Closure

↳ F+ is the set of FDs logically implied by F

↳ A+ is set of all attributes that are functionally determined by X

↳ start with X+ = X and repeat

↳ X+ is a key if X+ = {all attr}

· Lossless Decomp. is when the shared attr are a key for one of the tables

· BCNF

↳ For every non-trivial FD, (X -> Y) ∈ F+, X contains a key

↳ For any R in the schema

If (non-trivial X -> Y holds and X doesn't contain a key)

Compute X+

Decompose R into R1 (X+) and R2 (X, Z)

↳ X is common

↳ Z is all attr in R except X+

Repeat

↳ Lossless, not unique