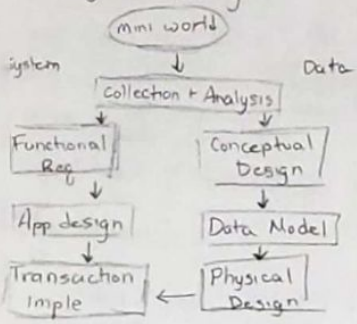
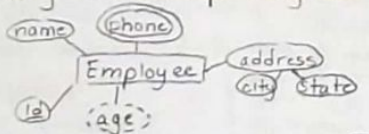
Principles of Info & Data Management

Data
- types (can be organized/categorized)
  - ↳ unstructured / structured / semistructured
- mini world source - restricted domain where we actually get data from
- Info System Design

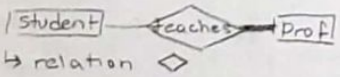


- concurrency - 2 users access the same file same time
- data base manag. sys. (DBMS)
  - ↳ efficient, relable, safe & convient
  - ↳ no data limit
  - ie
    - ↳ hiearchial, network, relation, obj oriented

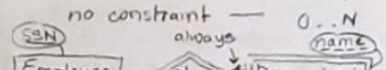
Entity Relationship Diagram



- ↳ entity ▭
- ↳ attribute ⬭
- ↳ multivalued ⬭⬭
- ↳ composite
- ↳ primary key
- ↳ derived
- candidate key - min. set of identifying attributes
- primary key - candidate key enforced by the system



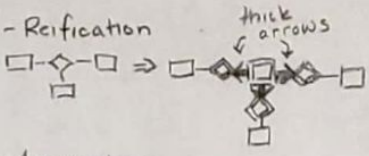
- ↳ relation ◇
- constraints
  - at least 1 ═ 1..N
  - exactly 1 ➔ 1..1
  - at most 1 → 0..1
  - no constraint — 0..N



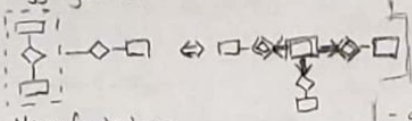
- ↳ weak entity ▭ (no pk)
- ↳ id relation ◇ (where key from)
- ↳ partial key ⬭ (id in set only)

- ↳ ISA hicarchy
  - ↳ common attributes
  - ↳ constraints
    - ↳ covering: complete/partial
    - ↳ overlap: disjoint/overlapping
  - relation btwn ISA subunits
- foreign key - field takes pk

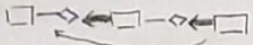
- Reification



- Aggregation

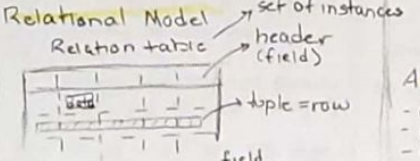


- Manifestation



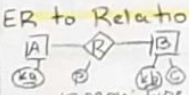
also has info from higher lvls

Relational Model



- schema (field, domain)
  - [name] ($f_1:d_1$, $f_2:d_2$, ...)
  - ↳ arity/degree - # fields
  - ↳ cardinality - # tuples
  - constraints/integrity - only insert data of same domain
  - ★ use null as empty placeholder
  - DELETE
    - ↳ make element null
  - INSERT
    - ↳ UPDATE a null or unused element w/ inserted info
  - CASCADE
    - ↳ push effects to referenced tables

ER to Relational



Basic Form

A (ka, primary key (ka))
B (kb, c, primary key (kb))
R (ka, kb, p, primary key (ka, kb), foreign key (ka) references A, foreign key (kb) references B)

- Reflexive
  - ↳ label IDs B_k and A_k
- One to Many
  - ↳ only need pk from the "one" side
- One to One
  - ↳ can choose pk from either side (but not both)

★ tables can be merged if the pk of 1 table is the foreign key of the other

- ISA → treat like a weak entity
- Aggregation → use pk of all entities connected inside

★ always merge on side w/ thick arrow!

SQL

"primary key" "not null"

→ CREATE TABLE [name] (
  [key name] [type] [cond]
  :
  FOREIGN KEY (~) REFERENCES (~)
  );

→ SELECT ~
  FROM ~
  WHERE ~ ;

- alias → FROM table t1, "as" rename column
  WHERE t1.xx
- subquery
  - ↳ select from where x = (select.. from..) (0 or more)
  - ↳ =/in, <, >, <=, >=, <>, %, _ (only 1)

→ INSERT INTO (f1, f2...) VALUES (v1, v2, ...)
  - ↳ all 1 line
  - ↳ f# = col name, v# = value to insert
  - → creates new row/rows

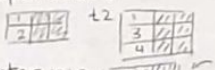
→ UPDATE [table] SET col = 'new value' WHERE primary key

→ DELETE FROM [table] WHERE [target rows]

Aggregate Ops

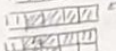
- count() → unique, *all
- max()
- min()
- sum()
- avg()
- group by [distinct value] → sum
  - ↳ having → like "where"
- join operators

t1 t2

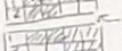


→ outer join ⟗ (nxt to each other)

⋈
→ inner join (only non null matches)



→ left join ⟕ (based on left table)

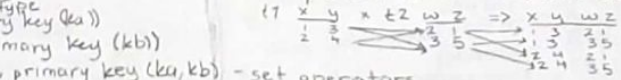


→ right join ⟖ (right)

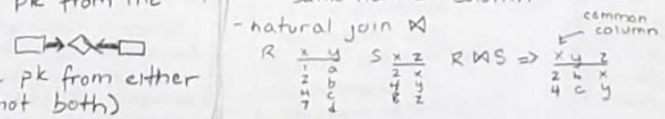
→ FROM [right] JOIN [left] USING (common field)

Relational Algebra

- projection (select) Π_{col} (table)
- selection (proj w/ cond) σ_{cond} (table)
- cross product (all combos) t1 x t2



- set operators
  - ↳ union ∪
  - ↳ intersect ∩
  - ↳ minus −
- rename $\rho_{(old \to new)}$ (table)
  - ↳ used when using 2 tables w/ same name column
- natural join ⋈ (common column)



  - ↳ ⋈ condition ... θ_{join}