## Primitive RA Operators

| | | | SQL DML |
|---|---|---|---|
| $\pi$ | Project (order does not matter) | (1) | SELECT DISTINCT (SELECT returns a bag, $\pi$ returns a set) |
| $\sigma$ | Select | (1) | WHERE |
| $\times$ | Cartesian Product | (2) | CROSS JOIN |
| $\cup$ | Union * | (2) | UNION |
| $-$ | Difference * | (2) | EXCEPT (ALL for bag) |

* must be union compatible
- Same no. of attributes
- domain of attributes must be compatible

**Project and Project**
$\pi_{\vec{X}}\pi_{\vec{Y}}R \equiv \pi_{\vec{X}}R$
You can eliminate any inner project (note that to be well formed $\vec{X}\subseteq\vec{Y}$)

**Project and Select**
$\pi_{\vec{X}}\sigma_{P(\vec{Y})}R \equiv \sigma_{P(\vec{Y})}\pi_{\vec{X}}R$
If $\vec{Y}\subseteq\vec{X}$ you can move a project of $\vec{X}$ inside a select of $\vec{Y}$

**Project and Product**
$\pi_{\vec{X}}(R \times S) \equiv \pi_{\vec{X}\cap Attrs(R)}R \times \pi_{\vec{X}\cap Attrs(S)}S$

**Project and Union**
$\pi_{\vec{X}}(R \cup S) \equiv \pi_{\vec{X}}R \cup \pi_{\vec{X}}S$

**Project and Difference**
$\pi_{\vec{X}}(R - S) \supseteq \pi_{\vec{X}}R - \pi_{\vec{X}}S$

**Select and Project**
$\sigma_{P(\vec{X})}\pi_{\vec{X}}R \equiv \pi_{\vec{X}}\sigma_{P(\vec{X})}R$

**Select and Select**
$\sigma_{P_x(\vec{X})}\sigma_{P_y(\vec{Y})}R \equiv \sigma_{P_x(\vec{X})\wedge P_y(\vec{Y})}R$

**Select and Product**
$\sigma_{P(\vec{X})}(R \times S) \equiv \sigma_{P(\vec{X})}R \times S \iff \vec{X}\subseteq Attrs(R)$
If $\vec{X}\subseteq Attrs(R)$ you can move a select predicate $P(\vec{X})$ onto R.

**Select and Union**
$\sigma_{P(\vec{X})}(R \cup S) \equiv \sigma_{P(\vec{X})}R \cup \sigma_{P(\vec{X})}S$

**Select and Difference**
$\sigma_{P(\vec{X})}(R - S) \equiv \sigma_{P(\vec{X})}R - S$

**Product and Union**
$R \times (S \cup T) \equiv (R \times S) \cup (R \times T)$

**Product and Difference**
$R \times (S - T) \equiv (R \times S) - (R \times T)$

**Union and Product**
$R \cup (S \times T)$ unable to move $\cup$ inside $\times$

**Union and Difference**
$R \cup (S - T)$ unable to move $\cup$ inside $-$

**Difference and Product**
$R - (S \times T)$ unable to move $-$ inside $\times$

**Difference and Union**
$R - (S \cup T) \equiv (R - S) - T$

## Derived RA Operators (2)

| | | | SQL DML |
|---|---|---|---|
| $\bowtie$ | Natural Join | $R \bowtie S = \sigma_{R.A_1=S.A_1 \wedge \dots \wedge R.A_m=S.A_m}(R \times S)$ | NATURAL JOIN |
| $\ltimes$ | Semi Join | $R \ltimes S = R \bowtie \pi_{Attr(R)\cap Attr(S)}S$ | |
| $\cap$ | Intersection * | $R \cap S = R - (R - S)$ | INTERSECT (ALL for bag) |
| $\div$ | Division | $R \div S = \pi_{Attr(R)-Attr(S)}R - \pi_{Attr(R)-Attr(S)}((\pi_{Attr(R)-Attr(S)}(R \times S)) - R)$ | SELECT DISTINCT x FROM t AS t1 WHERE NOT EXISTS(SELECT y FROM t EXCEPT SELECT y FROM t WHERE t.x=t1.x) } x÷y |
| $\overset{A=B}{\bowtie}$ | Equi Join | $R \overset{A=B}{\bowtie} S = \sigma_{R.A=S.B}(R \times S)$ | ↳ project the extra attributes in R that have each of the attributes of S |
| $\overset{\theta}{\bowtie}$ | Theta Join | $R \overset{\theta}{\bowtie} S = \sigma_{\theta}(R \times S)$ | JOIN.. ON |

$\pi_{cname,type}$ account

| cname | type |
|---|---|
| 'McBrien, P.' | 'current' |
| 'McBrien, P.' | 'deposit' |
| 'Boyd, M.' | 'current' |
| 'Poulovassilis, A.' | 'current' |
| 'Poulovassilis, A.' | 'deposit' |
| 'Bailey, J.' | 'current' |

$\pi_{type}$ account

| type |
|---|
| 'current' |
| 'deposit' |

$\pi_{cname,type}$ account ÷ $\pi_{type}$ account

| cname |
|---|
| 'McBrien, P.' |
| 'Poulovassilis, A.' |

## SQL Set Operations

- IN (WHERE x IN) — WHERE no IN (100, 101) listed produced
- EXIST — WHERE NOT EXIST (SELECT...) produced
- ALL — WHERE '11-jan-1999' < ALL (SELECT...) produced
- SOME — WHERE 'deposit' = SOME(SELECT...) produced

## SQL Null

nulls are combined in SETS

- x=null, null=null gives UNKNOWN
- x IS NULL, x IS NOT NULL gives TRUE/FALSE
- NOT IN (null, .....) always returns nothing!
- NOT EXIST(SELECT ....=null) always returns everything!

## SQL JOINS

LEFT/RIGHT JOIN $\overset{LR}{\bowtie}$ returns every row in left/right even if no row matches, filled with NULL instead

OUTER JOIN $\overset{O}{\bowtie}$ returns LEFT JOIN ∪ RIGHT JOIN — FULL OUTER JOIN IN SQL

## SQL Extensions

Pattern Matching: WHERE colum LIKE patten ESCAPE escape_char
- _ for single char
- % for any char
- [A-C] for single A to C
- [ABC] for single ABC

Cases: , CASE
WHEN ~
THEN ~
:
(ELSE ~) optional
END AS ~,

Modifications: ROUND(n, dp), SUBSTRING(str FROM $n_1$ FOR $n_2$),
(must be placed in SELECT clause) UPPER(str), CHAR_LENGTH(str), POSITION(char IN str)

Aggregate: SUM, COUNT, AVG, MIN, MAX excludes NULL

COALESCE(a,b,c) goes down list until non-null

---

## SQL OLAP

GROUP BY: only one output per group, aggregate functions for non grouped colums. NULL is grouped together

HAVING: similar to WHERE but for aggregates (SELECT.... FROM... WHERE... GROUP BY.... HAVING)

PARTITION: one output per row, AGG(x) OVER (PARTITION BY y) ....

ORDER BY ... DESC: default ascending, RANK() OVER (ORDER BY...) for numbering.

---

## DATALOG Head :- Body

1. If Body then Head.
2. Head is a single pred
3. Body is a conjuction of pred

a. cannot use same name for intentional (head) & extensional (body) preds.
b. preds start with small letter, vars start with capital letter
c. _ can be used for var that only appears once

Safe negation: A var in a negated ¬ pred must also appear in a non-negated pred.

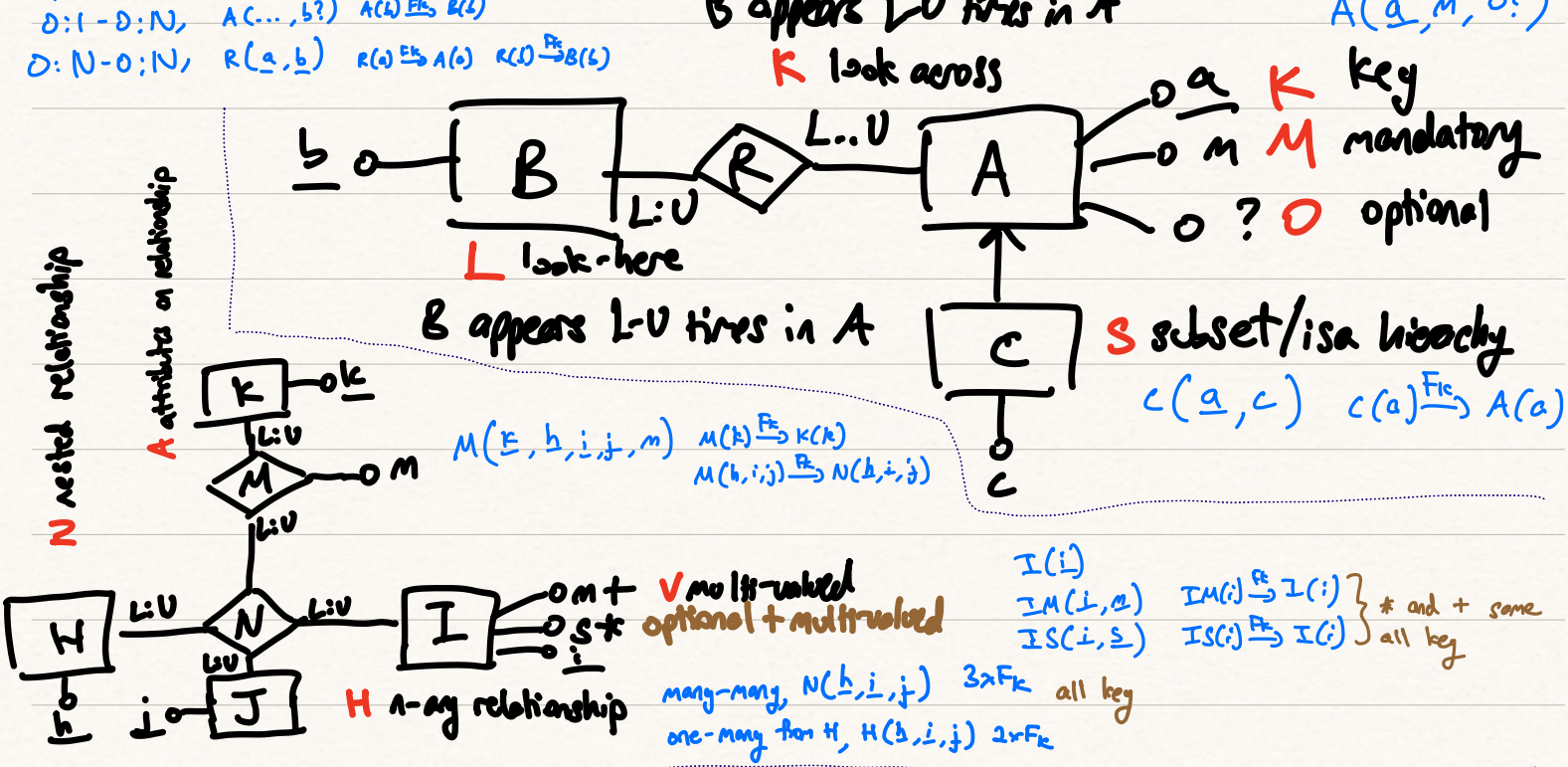
$\pi$: use subset of body vars in head
$\sigma$: use variable more than once/with data value
x: name two preds in rule body

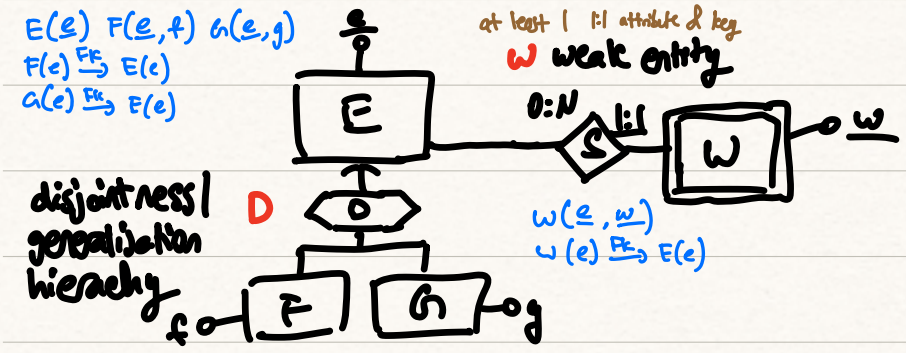
$\bowtie$: name two preds in rule body, with same name/compare attribute
U: more than one rule def for intentional pred
—: negation on pred being subtracted

---

# ER Modelling

1:1 - 0:N, A(..., b) A(b) FK→ B(b)
0:1 - 0:N, A(..., b?) A(b) FK→ B(b)
0:N - 0:N, R(a, b) R(a) FK→ A(a) R(b) FK→ B(b)




## Keys & FDs

Generate minimal keys: 1. Generate minimal cover
2. Find keys where $K^+ = Attr(R)$

Super-key X of R: X determines all other attributes of R

(non-unique) minimal-key X of R: not possible to remove attributes to form another super-key.

closure $S^+$: all FDs that can be inferred from S. $S \equiv T$ if $S^+ = T^+$

minimal cover $S_c$: $S_c^+ = S^+$ and not possible to remove another FD

Generate minimal cover:
1. rewrite S to have single attribute on RHS of each FD
2. For each FD $X \to A$, if $(X - B) \to B$, remove B from X
3. For each FD $X \to A$, compute $X^+$ without $X \to A$. If $A \subseteq X^+$, remove it

## Normalisation

1st Normal Form (1NF) Every attribute depends on key. (key$^+$ gives all attributes)

Prime Attributes Attributes A of R that are part of a minimal candidate key X of R

3rd Normal Form (3NF) For every non trivial FD $X \to A$ (in minimal cover!), either X super-key or A prime

Boyce-Codd Normal Form (BCNF) For every non trivial FD $X \to A$, X is a super key.

**Lossless-join decomposition** Decompose R into $R_1, \dots, R_n$ such that $Attr(R_1) \cup \dots \cup Attr(R_n) = Attr(R)$

**FD-preserving decomposition** Lossless decomposition preserve FDs if $S^+ = (S_a \cup S_b)^+$ may need to add more tables on top of lossless

**Generating 3/BCNF:** ① Find FD $(X \to A) \subseteq S$ which violates NF ($X$ not superkey, $A$ non-prime 3NF; $X$ not superkey BCNF) also possible to preserve FDs

② Decompose R into $R_a(Attr(R) - A)$ and $R_b(XA)$ $(X \subseteq R_a \ \& \ (X \to A) \subseteq R_b)$

③ Project S onto new relations and repeat from ①

---

# Serialisability & Recoverability

**Serialisability** Concurrent execution same end result as some serial execution of those transaction

**Recoverability** No transaction commits depending on uncommitted data

① **Lost Update** not serialisable, recoverable — $r_1[o] \prec w_2[o] \prec w_1[o]$ removing update leads to no change

② **Inconsistent Analysis** not serialisable, recoverable — $r_1[o_a] \prec w_2[o_a]$, $w_2[o_b] \prec r_1[o_b]$ reading halfway through update

③ **Dirty Read** serialisable, may not recoverable — $w_1[o] \prec r_2[o] \prec e_1$ read uncommitted data

④ **Dirty Write** may not serialisable, may not recoverable — $w_1[o] \prec w_2[o] \prec e_1$ overwrite uncommitted data

**Serialisability conflicts** occur between $(r_x[o]$ and $w_y[o])$ or $(w_x[o]$ and $w_y[o])$ in H

H is conflict serialisable (CSR) if the serialisation graph of H is acyclic. e.g. lost update ① ② inconsistent analysis, cycle

**Recoverable (RC)** if no transactions commit before another transaction it read from commits

**Avoids Cascading Aborts (ACA)** if no reads from non-committed transaction (no dirty read) ST ⊂ ACA ⊂ RC

**Strict (ST)** if no read from non-committed transaction (no dirty read) + no write over non-committed transaction (no dirty write.)

---

# Concurrency

## Two-phase Locking (2PL)

① growing phase + shrinking phase → only one peak / cannot unlock then lock

② refuse $rl_i[o]$ if $wl_j[o]$ is held

③ refuse $wl_i[o]$ if $rl_j[o]$ or $wl_j[o]$ is held

} delay transaction, put on waits-for-graph (WFG)

If WFG has no cycle → no deadlock

If WFG has cycle → deadlock → rollback (abort) one of the transactions