A **query** is a view with a distinguished **answer**/n relation:

answer(Emp,Mgr) ← employee(Emp, Salary, DeptNo), dept(DeptNo,Mgr)

answer(E, M) ∈ employee(E, –, D), dept(D, M). ↻ same thing

… D_1 … D_2 , $D_1 = D_2$.

11/12

- HW #4 ⇝ Files (instructions ⇝ hand-in)
- Midterm solutions posted ⇝ results by Thurs.
- Today: Datalog (~ HW #4)
  - discussion (more on Datalog, Triggers)
- Thursday: Storage structures, …, Indexing … Query Processing

---

$Q_1(X, U) \leftarrow R(X, Y), S(Y, Z, U)$ // $\bowtie, \pi$

$Q_1 = \pi_{A,E}(R \bowtie_{B=C} S)$

same ↻

$Q_2(X) \leftarrow R(X, X)$ // σ, π

$Q_2(X) \leftarrow R(X, Y), X = Y$ // σ, π

$Q_2 = \pi_A(\sigma_{A=B}(R))$

SQL: select A
from R
where A = B;

$R(A, B), S(C, D, E)$ rel. schema

σ
π
$\bowtie_C$
$-$ (minus)
$\cup$
ρ

Head ⟵ Body

$A(X) :- B(X,Y), C(Y).$ (⋈)

$A = \Pi_{B.1}\left(B \bowtie_{B.2=C.1} C\right)$

lhs ⟵ rhs, "premisse"

"conclusion"

RAlgebra: $A = B \cup C$: $B(U,V), C(W,Z)$

Datalog $A(X,Y) :- B(X,Y)$
$A(X,Y) :- C(X,Y)$

$A(X,Y) :- B(X,Y)$
; $C(X,Y)$.

RAlgebra: $A = B \cap C$

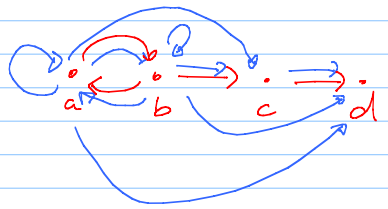
$A(X,Y) :- B(X,Y), C(X,Y)$

// $A(X_1,Y_2) :- B(X_1,Y_1), C(X_2,Y_2),$
$X_1 = X_2, Y_1 = Y_2.$

RAlgebra: $A = B \cup C$

Datalog $A(X,Y) :- B(X,Y)$
~~$A(X,Y) :- C(X,Y)$~~
$A(P,Q) :- C(P,Q).$

} correct, but bad style!



edge

a, b
b, a
b, c
c, d

↓C

a, b
b, a
b, c
c, d
b, d
a, a
b, b
a, c
a, d

```
tc(X,Y) :- edge(X,Y).   // (1)
tc(X,Y) :- edge(X,Z), tc(Z,Y).   // (2)
```

```
A(X,Y) :- B(X,Y), not C(X,Y)
```

RA: $A = B \setminus C$ // $A = B - C$

```
A(X,Y) :- B(X,Y), not D(Y).
```

DATALOG: Examples for Relational Operations

Relational operations have concise representations! Examples:

$\sigma_{X=a, Y \neq X}(p)$

```
sel(X,Y) :- p(X,Y), X=a, not X=Y.   % SELECT some tuples from p(X,Y)

proj(X) :- p(X,Y).                  % PROJECT on the first argument

join(X,Y,Z) :- p(X,Y), q(Y,Z).      % JOIN p(A,B), q(C,D) s.t. B=C

prod(X,Y) :- p(X), q(Y).            % PRODUCT of p(X) and q(Y)

intersect(X) :- p(X), q(X).         % INTERSECTION of p(X), q(X)

diff(X) :- p(X), not q(X).          % SET-DIFFERENCE: p(X) \ q(X)

union(X) :- p(X).                   % UNION of p(X), ...
union(X) :- q(X).                   % ... and q(X)
```

$\Pi_X(p)$

$p \bowtie q$ $\quad p(A,B)\ q(B,C)$

$p \times q$

Rules have a "logical reading" (i.e., rules are formulas):

$$\forall X\ (\ \texttt{diff}(X) \leftarrow \texttt{p}(X) \wedge \neg \texttt{q}(X)\ ).$$
$$\forall X\ (\ \texttt{union}(X) \leftarrow \texttt{p}(X) \vee \texttt{q}(X)\ ).$$

Safety

$A(X,Y) :- B(X).$ ↯ not safe

$A(X,Y,Z) :- B(X,Y), \neg C(Y,Z)$ ↯ not safe

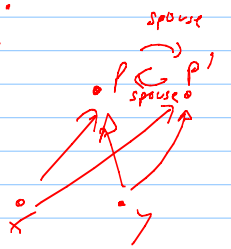
$Q(Y) :- B(X), \neg C(Y).$

Safety of rules:

Every variable in a (safety) rule must occur positively in the body!

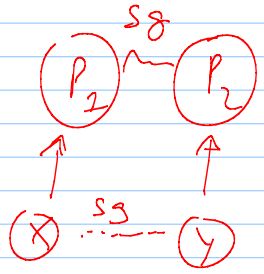
Proposition: rule r safe ⇒ r is range-restricted ⇒ finite answer

Same generation:



$sg(X,Y) :- \ldots$

if X and Y share a parent.



if P_1, P_2 are in SG, and X, Y are children of P_1, P_2, resp.

⇒ X, Y are in SG

Relational Division in Datalog

```
rents(Sailor, Boat)
sailor(Sailor)
boat(Boat)
```

```
rents(jane, green).
rents(jane, blue).
rents(bob, red).
rents(tom, red).
rents(jane, red)
```

Who rents all the boats?

rents ÷ boat

```
not_all(S) :- boat(B),
              sailor(S),
              not rents(S,B).
```

```
all_boats(S) :-
              sailor(S),
              not not_all(S).
```