- Derivation: $r \cap s = r - (r - s)$ $= s - (s - r)$
- Example: given the relations r and s

r

r.A	r.B
α	1
α	2
β	1

s

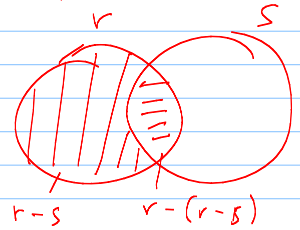
s.A	s.B
α	2
β	3

$r \cap s$

A	B
α	2

$= \Pi_{r.A, r.B}\left(\sigma_{r.A = s.A \wedge r.B = s.B}(r \times s)\right)$

$= r \bowtie s$



$r \times s$

r.A	r.B	s.A	s.B

} 6 rows

$\downarrow \sigma \ldots$

r.A	r.B	s.A	s.B
α	2	α	2

$\downarrow \Pi_{r.A, r.B}$

r.A	r.B
α	2

---

r

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

t_r

s

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	τ

t_s

=

=

for $t_r \in r$

for $t_s \in s$

if t_r and t_s "agree" on the shared columns

then output $(t_r . t_s)$

natural join

$r \bowtie s$

r.B = s.B

r.D = s.D

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

- Example: Given the relations $R(A, B, C, D)$ and $S(B, D, E)$
  - Join can be applied because $R \cap S \neq \emptyset$
  - the result schema is (A, B, C, D, E)
  - and the result of $r \bowtie s$ is defined as

$$\underbrace{\pi_{r.A, r.B, r.C, r.D, s.E}}_{\text{projection on 5 col}}(\underbrace{\sigma_{r.B = s.B \wedge r.D = s.D}}_{\text{"join"}}(\underbrace{r \times s}_{\text{7 col}}))$$

rename SC

r

A	B	C
4	5	6
7	8	9

s

C	D
6	8
10	12

$r \bowtie s$

A	B	C	D
4	5	6	8

$r \bowtie_{C=SC} (\underbrace{\rho_{S(SC,D)}(s)}_{\text{renaming to avoid name clash with A}})$

A	B	C	SC	D
4	5	6	6	8

$r \underset{r.c = s.c}{\bowtie} s$

---

shared attributes

(students)

r

A	B	C	D	E
~~α~~	~~a~~	~~α~~	~~a~~	~~1~~
α	a	γ	a	1
α	a	γ	b	1
~~β~~	~~a~~	~~γ~~	~~a~~	~~1~~
~~β~~	~~a~~	~~γ~~	~~b~~	~~3~~
γ	a	γ	a	1
γ	a	γ	b	1
~~γ~~	~~a~~	~~β~~	~~b~~	~~1~~

γ a γ c 5

(db classes)

s

D	E
a	1
b	1

$\forall$-query

which students (A, B, C) have taken <u>all</u> the classes (D, E)?

$r \div s$

A	B	C
α	a	γ
γ	a	γ

BOOKS(DocId, Title, Publisher, Year)

STUDENTS(StId, StName, Major, Age)

AUTHORS(AName, Address)

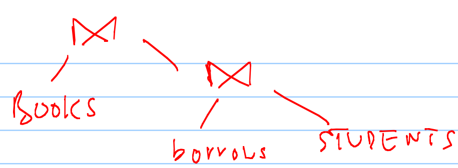
borrows(DocId, StId, Date)

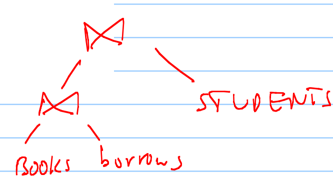
has-written(DocId, AName)

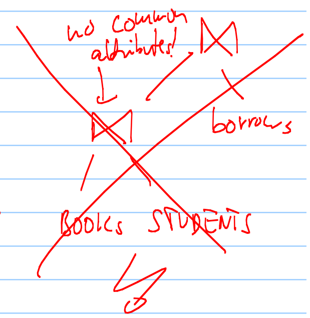
describes(DocId, Keyword)

2. *List each student with the books s/he has borrowed.*

   BOOKS $\bowtie$ (borrows $\bowtie$ STUDENTS)







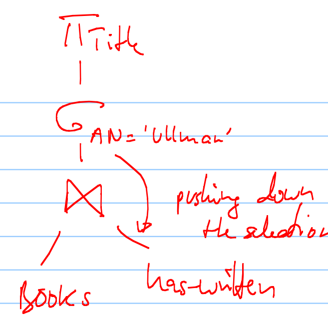
---

3. *List the title of books written by the author 'Ullman'.*

   $\pi_{\text{Title}}(\sigma_{\text{AName='Ullman'}}(\text{BOOKS} \bowtie \text{has-written}))$

   or

   $\pi_{\text{Title}}(\text{BOOKS} \bowtie \sigma_{\text{AName='Ullman'}}(\text{has-written}))$



5. *Which books have* <u>*both*</u> *and!* *keywords 'database' and 'programming'?*

BOOKS $\bowtie$ $(\pi_{\text{DocId}}(\sigma_{\text{Keyword='database'}}(\text{Descriptions})) \cap \pi_{\text{DocId}}(\sigma_{\text{Keyword='programming'}}(\text{Descriptions})))$

Books $\bowtie$ $\sigma_{\text{Keyw='db'} \wedge \text{Keyw='PL'}}$(Descriptions)

↑ never true

$\sigma_{\text{Kw='db'} \vee \text{Kw='PL'}}(\ldots)$

too many results

- A typical SQL query has the form

  select $A_1, A_2, \ldots, A_n$ → * keep everything (all columns)

  from $r_1, r_2, \ldots, r_k$ ~ $r_1 \times r_2 \times \ldots \times r_k$

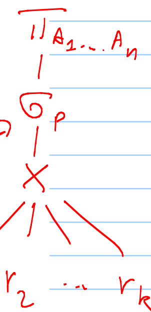
  [**where** P]

  – A_is represent attributes

  – r_is represent relations

  – P is a predicate

select * from where



- This query is equivalent to the relational algebra expression

$$\underbrace{\pi_{A_1,A_2,\ldots,A_n}}_{\text{SELECT}}(\underbrace{\sigma_P}_{\text{where}}(\underbrace{r_1 \times r_2 \times \ldots \times r_k}_{\text{From}}))$$

SQL

special case

select *

From r; $\hat{=}$ r

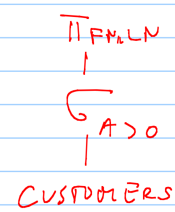
List the first and last name of customers having a negative account. (balance)

select FName, LName

from CUSTOMERS

where Account < 0;

$\hat{=}$



Which suppliers (SName) offer a MegaPC or a TinyMac? ← and

select SName **from** offers ← and ? No!

where Prodname = 'MegaPC' **or** Prodname = 'TinyMac';

$\Pi_{SN}(\sigma_{PN='M' \vee PN='T'}(\text{offers}))$

$\hat{=}$ . . . **where** Prodname **in** ('MegaPC','TinyMac')

10/17

Find all pizzas eaten by at least one female over the age of 20.

Person(name, age, gender) // name is a key
Frequents(name, pizzeria) // [name,pizzeria] is a key
Eats(name, pizza) // [name,pizza] is a key
Serves(pizzeria, pizza, price) // [pizzeria,pizza] is a key

eats(N, P)
person(N, A, G)

$$\pi_P \left(Eats \bowtie \sigma_{A>20,\, G=female}(person) \right)$$

---

offers

SN	PN
s_1	TinyMac
s_2	MegaPC
s_2	TinyMac
..	..

suppliers offering TM or MPC

$$\pi_{SN}\ \sigma_{PN='TM' \vee 'MPC'}\ offers$$

$$\pi\left(\sigma\ offers \cup \sigma\ offers\right)$$

→ ... and ..

$$\pi_{O1.SN}\ \sigma_{O1.SN=O2.SN \wedge O1.PN='TM' \wedge O2.PN='MPC'} \left(\rho_{O1(O1.SN,\, O1.PN)}\ offers \times \rho_{O2(O2.SN,\, O2.PN)}\ offers \right)$$

$$\pi_{SN}\ \sigma_{PN='TM'}\ offers \ \cap\ \pi_{SN}\ \sigma_{PN='MPC'}\ offers$$

$$\pi_{SN}\ \sigma_{PN1='TM' \wedge PN2='MPC'\ [\vee PN1='MPC' \wedge PN2='TM']} \left(\rho_{O1(SN,\, PN1,\, Pr_1)}\ offers \bowtie \rho_{O2(SN,\, PN,\, Pr_2)}\ offers \right)$$

offers [SN, PN, Price]

Division revisited

$$R(A,B,C) \div S(B) \rightsquigarrow T(A,C)$$

$$R(A,B,C) \div \underbrace{\{(a_1,c_1),(a_2,c_2)\}}_{\rho_{\ldots(A,C)}\{\ldots\}}(A,C) \rightsquigarrow U(B)$$

borders

C1	C2	...
US	MX	
MX	US	
US	CA	
CA	US	

$\pi_{C1}(\text{borders}) = \{US, MX, CA\}$

```
select C1
From borders;
```

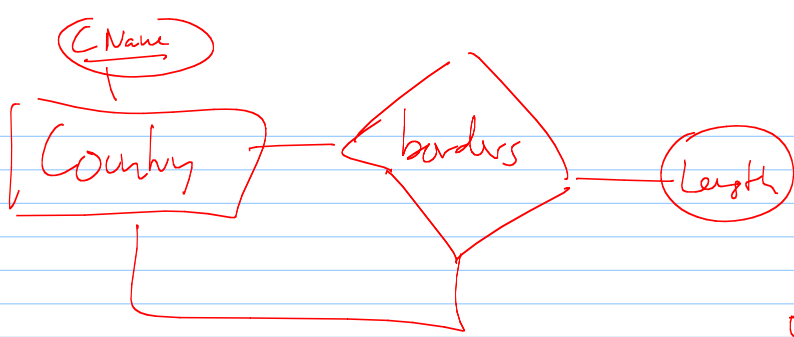
⇝ US
MX
US
MX

```
select distinct C1
From borders;
```

⇝ US
MX
CA

SQL has bag (duplicate) semantics for many operations

Exception: union, intersection



borders

C1	C2
US	MX
MX	US
~~Australia~~	~~NULL~~

```
select
from borders B1, borders B2, borders B3
where B1.C1 = B2.C1
  and B2.C1 = B3.C1
  and B1.C2 ≠ B2.C2
  and B2.C2 ≠ B3.C2
  and B3.C2 ≠ B1.C2 .
```