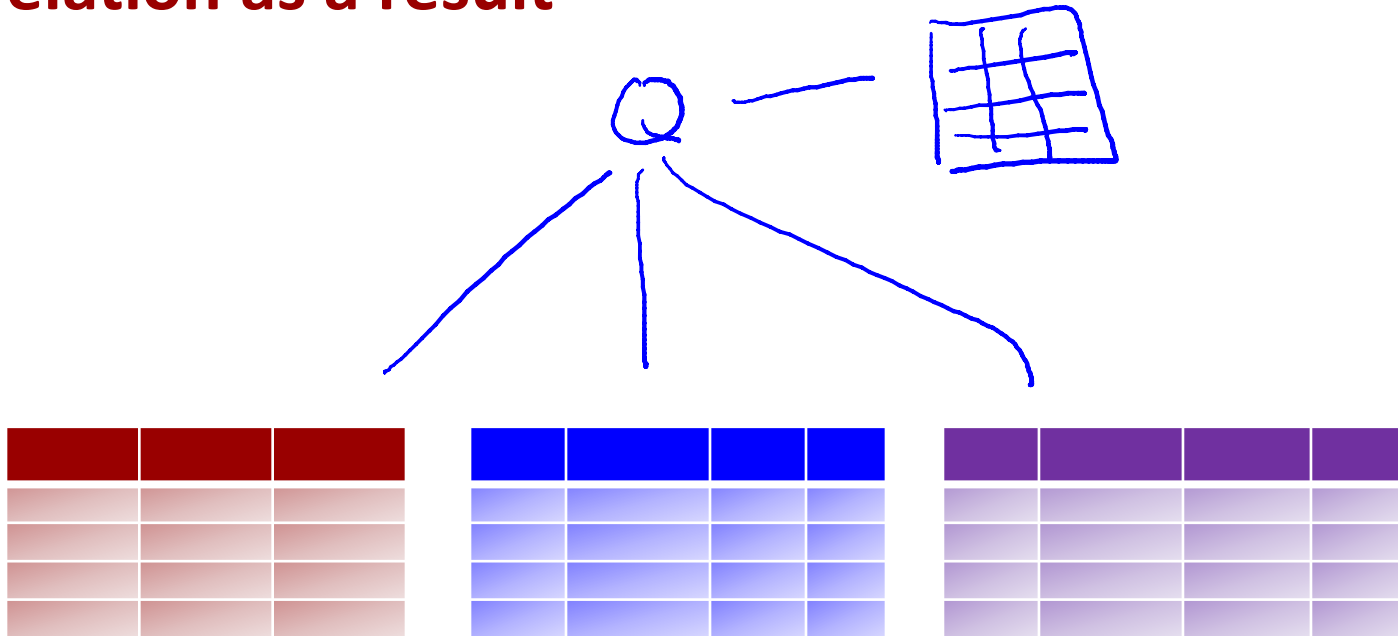


Relational Databases

Relational Algebra (1)

Select, project, join

Query (expression) on set of relations produces relation as a result



Examples: simple college admissions database

College(cName, state, enrollment) ←

Student(sID, sName, GPA, sizeHS) ←

Apply(sID, cName, major, decision) ←

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Simplest query: relation name

Use operators to filter, slice, combine

Student — 

↓

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Select operator: picks certain rows

Students with GPA > 3.7

$\sigma_{GPA > 3.7}$ Student



σ_{cond} Rel.

Students with GPA > 3.7 and HS < 1000

$\sigma_{GPA > 3.7 \wedge HS < 1000}$ Student

Applications to Stanford CS major

$\sigma_{cName = 'stanford' \wedge major = 'cs'}$ Apply

college

cName	state	enr

Student

sID	sName	GPA	HS

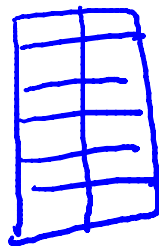
Apply

sID	cName	major	dec

Project operator: picks certain columns

ID and decision of all applications

$\pi_{sID, dec}$ Apply



π_{A_1, \dots, A_n} Rel

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

To pick both rows and columns...

ID and name of students with GPA > 3.7

$$\pi_{sID, sName}(\sigma_{GPA > 3.7} Student)$$

$$\sigma_{cond}(Expr)$$

$$\pi_{A_1, \dots, A_n}(Expr)$$

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Duplicates

List of application majors and decisions

$\Pi_{\text{major, dec}} \text{ Apply}$

SQL: Multisets, bags

R.A.: Sets

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Cross-product: combine two relations
(a.k.a. **Cartesian product**)

Student \times Apply

college

cName	state	enr

Student.sID
5 tuples

sID	sName	GPA	HS

Apply.sID
A tuples

sID	cName	major	dec

*S + A
attributes*

*S \times A
tuples*

Cross-product: combine two relations (a.k.a. Cartesian product)

Names and GPAs of students with $HS > 1000$ who applied to CS and were rejected

$$\pi_{\langle \text{Name}, \text{GPA} \rangle} \left(\sigma_{\text{student.sID} = \text{Apply.sID} \wedge HS > 1000 \wedge \text{major} = 'cs' \wedge \text{dec} = 'R'} (\text{Student} \times \text{Apply}) \right)$$

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Natural Join



- Enforce equality on all attributes with same name ←
- Eliminate one copy of duplicate attributes ←

college

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Natural Join

Names and GPAs of students with HS>1000 who applied to CS at college with enr>20,000 and were rejected

$\Pi_{sName, GPA}$

$\left(\sigma_{HS > 1000 \wedge major = 'cs'} (\underline{Student} \bowtie (\underline{Apply} \bowtie College)) \right)$
 $\wedge dec = 'R' \wedge enr > 20,000$

college		
cName	state	enr

Student			
sID	sName	GPA	HS

Apply			
sID	cName	major	dec

Natural Join

$$Exp_1 \bowtie Exp_2 \equiv$$

$$\Pi_{\text{Schema}(E_1) \cup \text{Schema}(E_2)} \left($$

$$\sigma_{E_1.A_1 = E_2.A_1 \wedge E_1.A_2 = E_2.A_2 \wedge \dots} (Exp_1 \times Exp_2)$$

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Theta Join

$$Exp_1 \bowtie_{\theta}^{condition} Exp_2 \equiv \sigma_{\theta} (Exp_1 \times Exp_2)$$

- Basic operation implemented in DBMS
- Term “join” often means theta join

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Query (expression) on set of relations produces relation as a result

- Simplest query: relation name
- Use operators to filter, slice, combine
- Operators so far: select, project, cross-product, natural join, theta join