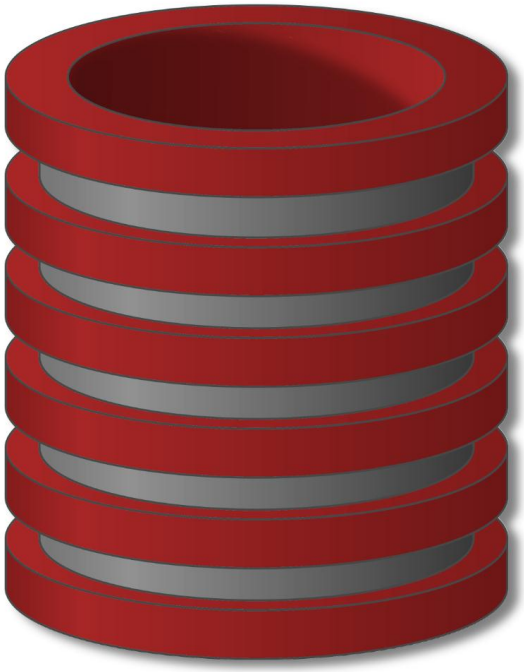
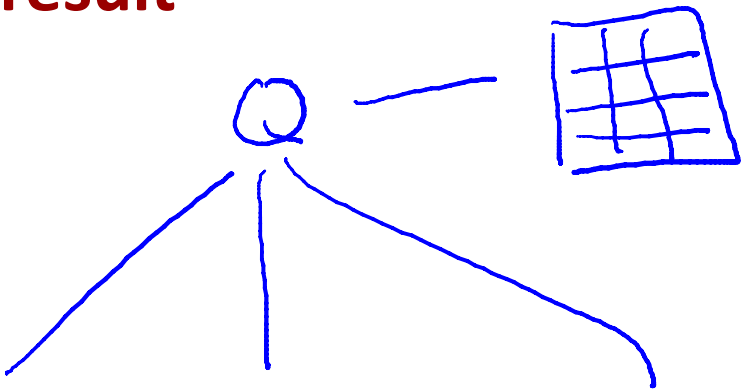


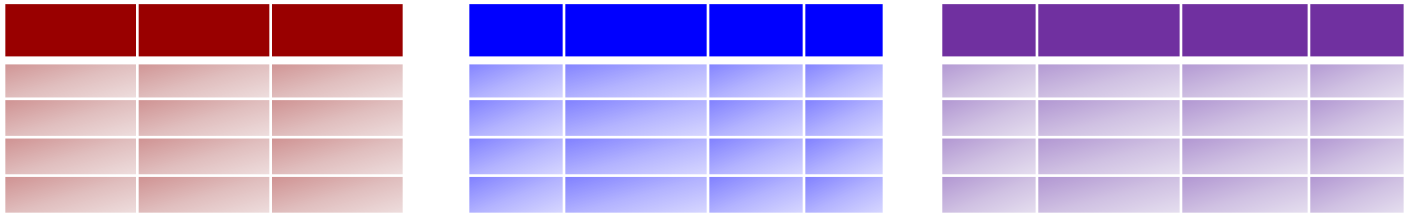
Relational Databases

Relational Algebra (1)
Select, project, join

Jennifer Widom

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

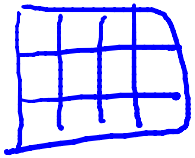
sID	cName	major	dec

Relational Algebra (1)

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

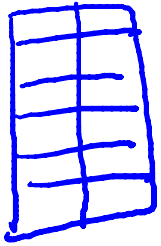
Apply

sID	cName	major	dec

Project operator: picks certain columns

ID and decision of all applications

$\pi_{sID,\,dec}$ Apply



$\pi_{A_1,\ldots,A_n}$ 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

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

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

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

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Duplicates

List of application majors and decisions

$\Pi_{major,\ dec}$ 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 × Apply

Student.sID

S tuples

Apply.sID

A tuples

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

S × 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

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

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

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

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Natural Join

$$Exp_1 \bowtie Exp_2 \equiv$$

$$\pi_{schema(E_1) \cup schema(E_2)} \Big(\sigma_{E_1.A_1 = E_2.A_1 \wedge E_1.A_2 = E_2.A_2 \wedge \ldots} \big(Exp_1 \times Exp_2 \big) \Big)$$

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

Theta Join

condition

$$Exp_1 \bowtie_\theta Exp_2 \equiv \sigma_\theta (Exp_1 \times Exp_2)$$

- 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