Relational Algebra

DBI - Databases and Interfaces
Dr Matthew Pike & Prof Linlin Shen

Overview of next 3-4 lectures

In the next 4 lectures we will see how to translate:

English <-> Relational Algebra <->SQL queries

English: “Find all universities with > 20000 students”

Relational Algebra: $\pi_{uName}(\sigma_{Enrollment > 20000}(University))$

SQL: SELECT uName FROM University WHERE University.Enrollment>20000

This Lecture

- Relational algebra - Operators
    - Projection, Selection
    - Product, Join
    - Union, Intersection, Difference
    - Rename
    - Examples

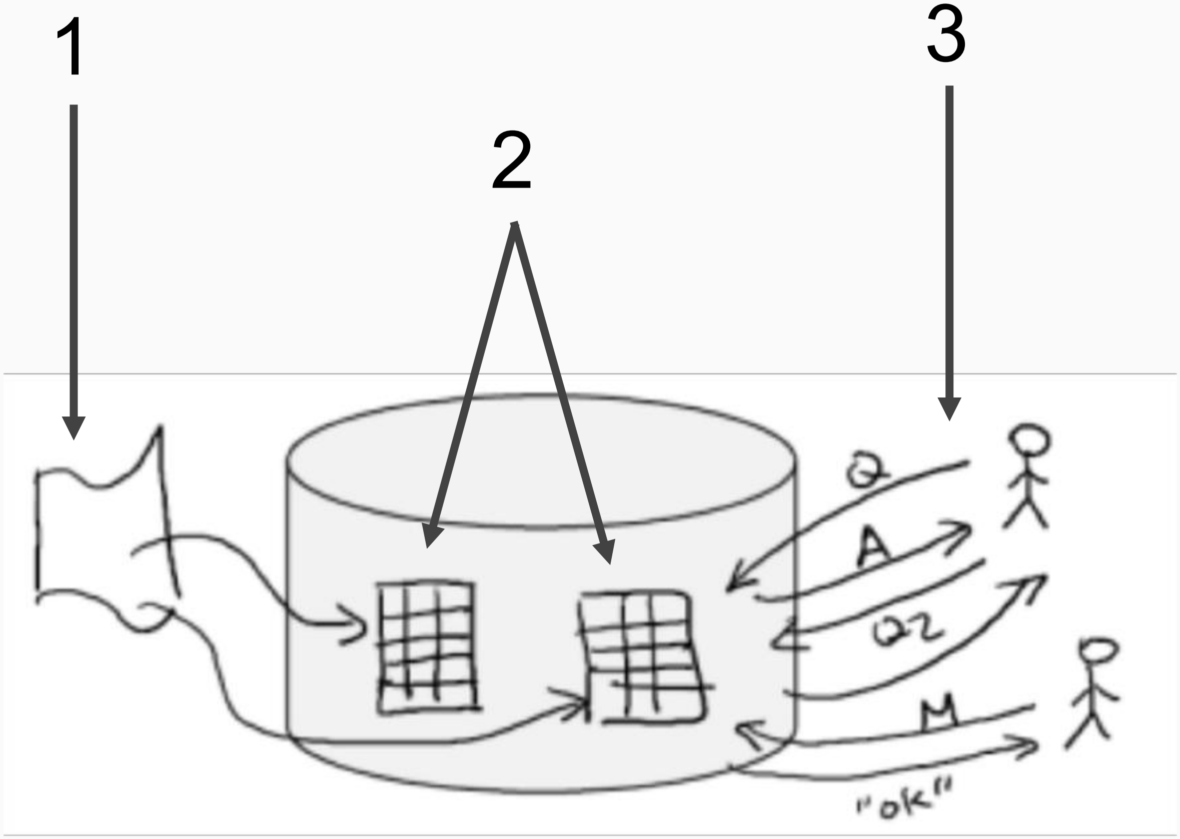
- Chapters 4, 5 of the DB Book

Relational Databases

Create and Use Relational Databases

Basic Steps:

1. Design Schema
2. Insert Data
3. Execute Queries and Modifications



Querying Relational Databases

University		
uName	**County**	**Enrollment**
NOTT	Nott/shire	18000
CAM	Cam/shire	22000
UCL	Great/Lon	20000

- Example with 3 Relations
  - “All universities with > 20000 students”
  - “All engineering departments with < 2000 applicants”
  - “Uni in Nott/shire with highest average GPA”

Student			
SID	**sName**	**GPA**	**HS**
0135	John	18.5	100
0025	Mary	19.3	1000
0423	Mary	17.5	300

Apply			
SID	**uName**	**Subj**	**Dec**
0135	CAM	CS	‘A’
0135	NOTT	CS	‘A’
0423	NOTT	ENG	‘R’

Querying Relational Databases

- Queries are expressed in high level language
- Query language in DML
  - (Data Manipulation Language: querying and data modification)
- Complicated queries expressed in a compact way
  Declarative (vs. Procedural):
  - no need to specify how (the algorithm)
- Some queries are easy to pose
- Some queries easy to execute

Closures

- When you ask a query over a relation you get back a relation
- “When you get back the same type of object that you query, that's known as closure of the language.”

- This is called :
  - **Closure**

University		
uName	**County**	**Enrollment**
NOTT	Nott/shire	18000
CAM	Cam/shire	22000
UCL	Great/Lon	20000

E.g. “Find all universities with >= 20000 students”

uName	**County**	**Enrollment**
CAM	Cam/shire	22000
UCL	Great/Lon	20000

Composition

- The result can be used as input to another query
- The ability to run a query over the result of your previous query.
- This is called:
  - **Composition**

Result of previous query

uName	County	Enrollment
CAM	Cam/shire	22000
UCL	Great/Lon	20000

E.g. “Find all universities in Cambridgeshire from” (“Find all universities with > 20000 students”) “

uName	County	Enrollment
CAM	Cam/shire	22000

Relational Algebra VS SQL

- 2 Query Languages
  - Relational Algebra is the formal language
    - Provides the theoretical foundations for SQL
  - SQL is the real language
    - What runs an actual deployed database application
- Example
  - “Find all universities with > 20000 students”
  - RA
    - $\pi_{uName}(\sigma_{Enrollment > 20000}(University))$
  - SQL
    - SELECT uName FROM University WHERE University.Enrollment>20000

Relational Algebra Operator Types

Relational Algebra Operators

- Operators in Relational Algebra allow us to filter, slice and combine relations
- We will use our University/Student/Apply example:
  - Underlined are keys for each relation
  - Question:
    - What does it mean that SID, uName, Subj is a key for Apply?

University		
uName	**County**	**Enrollment**
NOTT	Nott/shire	18000
CAM	Cam/shire	22000
UCL	Great/Lon	20000

Student			
SID	**sName**	**GPA**	**HS**
0135	John	18.5	100
0025	Mary	19.3	1000
0423	Mary	17.5	300

Apply			
SID	**uName**	**Subj**	**Dec**
0135	CAM	CS	‘A’
0135	NOTT	CS	‘A’
0423	NOTT	ENG	‘R’

Relational Algebra Operators: Filter

- Operators in Relational Algebra allow us to filter, slice and combine relations
  - **Filter** means row removal

Student			
SID	**sName**	**GPA**	**HS**
0135	John	18.5	100
~~0025~~	~~Mary~~	~~19.3~~	~~1000~~
0423	Mary	17.5	300

Relational Algebra Operators: Slice

- Operators in Relational Algebra allow us to filter, slice and combine relations
  - **Slice** means column removal

Student

<u>**SID**</u>	**sName**	**GPA**	**HS**
0135	John	18.5	100
0025	Mary	19.3	1000
0423	Mary	17.5	300

Relational Algebra Operators: Combine

- Operators in Relational Algebra allow us to filter, slice and combine relations
  - **Combine** means combine rows or columns

Student X Apply

S.SID	sName	GPA	HS	A.SID	uName	Subj	Dec
0135	John	18.5	100	0135	CAM	CS	‘A’
0135	John	18.5	100	0135	NOTT	CS	‘A’
0135	John	18.5	100	0423	NOTT	ENG	‘R’
0025	Mary	19.3	1000	0135	CAM	CS	‘A’
0025	Mary	19.3	1000	0135	NOTT	CS	‘A’
0025	Mary	19.3	1000	0423	NOTT	ENG	‘R’
0423	Mary	17.5	300	0135	CAM	CS	‘A’
0423	Mary	17.5	300	0135	NOTT	CS	‘A’
0423	Mary	17.5	300	0423	NOTT	ENG	‘R’

Relational Algebra Operators

Select Operator

The Select Operator picks certain rows (filtering)

We use Sigma (σ) as the symbol to represent the select operator.

E.g. 1: "(Find) Students with GPA > 19"

$$\sigma_{GPA > 19}(Student)$$

Returns subset of the student table with rows such that GPA > 19

<u>SID</u>	sName	GPA	HS
0025	Mary	19.3	1000

E.g. 2: "Students with GPA > 19 and HS<1000"

$$\sigma_{GPA > 19 \text{ and } HS < 1000}(Student)$$

E.g. 3: "Applications to Notts with subject CS"

$$\sigma_{uName = 'Notts' \text{ and } Subj = 'CS'}(Apply)$$

Select Operator

- The **Select** operator picks certain rows (filtering)
- General form

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

- E is an Expression of Relational Algebra
- E could be a Relation
- OR E could be the result of applying an Operator (not necessarily only the Select) to a Relation

E.g. $\sigma_{GPA > 19}(\sigma_{HS > 100}(Student))$

Project Operator

The **Project** Operator picks certain attributes (slicing)

We use PI (π) to represent the projection operation.

E.g. 1: "(Get) the IDs and decisions from all applications"

$$\pi_{SID,Dec}(Apply)$$

SID	Dec
0135	'A'
0135	'A'
0423	'R'

There's no condition in the **Project** Operator

General Form:

$$\pi_{A1,A2,\ldots,An}(E)$$

E is an Expression of Relational Algebra

A1, A2, …, An are the attributes to be kept

Select + Project

- The **Select** and **Project** Operators can be naturally combined
- E.g.:
  - "(Get) the IDs and names of students with GPA > 19"

Step 1:: SID, sName (GPA> 19(Student))

Step 2:: $\pi_{SID,\ sName}(\sigma_{GPA>\ 19}(\textbf{Student}))$

- You can compose as much as you like **select, project, select, select, project,….**

Duplicates

A frequent result of applying Project is the creation of duplicates

E.g.: “List the subjects and decisions of all applications”

$$\pi_{\text{Subj, Dec}}(\text{Apply})$$

<u>**Subj**</u>	**Dec**
CS	‘A’
CS	‘A’
ENG	‘R’
CS	‘A’
CS	‘A’
ENG	‘R’

VS

<u>**Subj**</u>	**Dec**
CS	‘A’
ENG	‘R’

- In Relational Algebra all duplicates **are** eliminated.
  - **NOT** THE CASE for SQL
- Relational Algebra is based on Sets. SQL is based on multi-Sets

Cross Product Operator

- Also known as Cartesian Product (operator x)
- It is applied between 2 or more relations The schema of the result is the union of the schema of the two relations
- The contents of the result are all combination of tuples from the 2 relations

E.g.: **Student** x **Apply**

Student

SID	sName	GPA	HS
0135	John	18.5	100
0025	Mary	19.3	1000
0423	Mary	17.5	300

Apply

SID	uName	Subj	Dec
0135	CAM	CS	‘A’
0135	NOTT	CS	‘A’
0423	NOTT	ENG	‘R’

Student X Apply

S.SID	sName	GPA	HS	A.SID	uName	Subj	Dec
0135	John	18.5	100	0135	CAM	CS	‘A’
0135	John	18.5	100	0135	NOTT	CS	‘A’
0135	John	18.5	100	0423	NOTT	ENG	‘R’
0025	Mary	19.3	1000	0135	CAM	CS	‘A’
0025	Mary	19.3	1000	0135	NOTT	CS	‘A’
0025	Mary	19.3	1000	0423	NOTT	ENG	‘R’
0423	Mary	17.5	300	0135	CAM	CS	‘A’
0423	Mary	17.5	300	0135	NOTT	CS	‘A’
0423	Mary	17.5	300	0423	NOTT	ENG	‘R’

Cross Product Operator

- Student x Apply
    - If S tuples from Student and A tuples from Apply, in total S x A tuples (3 x 3 = 9 in this example)
    - If attributes share the same name, they should be renamed
        - e.g. S.SID and A.Apply
    - Notice that some rows make little sense!
        - For example S.SID = 0135 is combined with A.SID = 0423
    - E.g. 2 “Names and GPAs of students with HS>1000 who applied to CS and were rejected

$$\pi_{GPA,sName}(\sigma_{S.SID=A.SID \text{ and } HS>1000 \text{ and } subj=\text{'CS'} \text{ and } dec=\text{'Rej'}}\ \mathbf{(St\ x\ Ap)})$$

Natural Join Operator

- **Student** $\bowtie$ **Apply** (bowtie)
  - Same as Cross Product but enforces equality on all attributes with the same name (S.SID and A.SID in our case)
  - Automatically sets values equal when attribute names are the same
  - Gets rid of multiple copies of the attributes with the same name (there will be only one common SID attribute in the result)

Student $\bowtie$ **Apply**						
SID	**sName**	**GPA**	**HS**	**uName**	**Subj**	**Dec**
0135	John	18.5	100	CAM	CS	‘A’
0135	John	18.5	100	NOTT	CS	‘A’
0423	Mary	17.5	300	NOTT	ENG	‘R’

Natural Join Operator

- E.g. 1 "Names and GPAs of students with HS>1000 who applied to CS and were rejected"
    - $\pi_{GPA,sName}(\sigma_{HS>1000 \text{ and } subj='CS' \text{ and } dec='Rej'}(\textbf{Student} \bowtie \textbf{Apply}))$
- E.g. 2 "Names and GPAs of students with HS>1000 who applied to CS at Universities with enrolment > 20000 and were rejected"
    - $\pi_{GPA,sName}(\sigma_{HS>1000 \text{ and } subj='CS' \text{ and } Enrollment>20000 \text{ and } dec='Rej'}(\textbf{Student} \bowtie (\textbf{Apply} \bowtie \textbf{Uni})))$
- Natural join does not add expressive power to Relational Algebra, just facilitates the writing of complex queries

Theta Join Operator

- Similar to Cartesian Product and Natural Join
- Can be implemented via Cartesian Product and Select.
- The Theta Join operator is defined as
    - Student $\bowtie_{\theta}$ Apply = σ_{θ} (Student × Apply)
- The result of this operation consists of all combinations of tuples in Student and Apply that satisfy condition θ
- A theta join allows for arbitrary comparison relationships (such as ≥).
- Theta join does not add expressive power to Relational Algebra, just facilitates the writing of complex queries

A Summary

61

+50

A *theta join* allows for arbitrary comparison relationships (such as ≥).

An *equijoin* is a theta join using the equality operator.

A *natural join* is an equijoin on attributes that have the same name in each relationship.

Additionally, a natural join removes the duplicate columns involved in the equality comparison so only 1 of each compared column remains; in rough relational algebraic terms:

$$\bowtie \; = \pi_{R,S-\mathbf{a}_s} \circ \bowtie_{\mathbf{a}_R=\mathbf{a}_S}$$

share edit flag

edited Oct 24 '14 at 19:56

answered Oct 24 '11 at 0:04

outis

46.2k ● 8 ● 76 ● 123

http://stackoverflow.com/questions/7870155/difference-between-a-theta-join-equijoin-and-natural-join

Union Operator

Union operator combines information between 2 relations vertically (cross product or join combines information horizontally)

- Only relations with same schemas can be combined using the Union operator (not the exactly the case here!)
- **Duplicates are always eliminated!**

E.g. 1 “List all Student and University names”

π_{sName} (**Student**) U π_{uName} (**University**)

Name
John
Mary
CAM
NOTT
UCL

Difference Operator

Better illustrated with example:

- E.g. 1 “IDs of all students who didn’t apply anywhere”

$$\pi_{\text{SID}}\ (\textbf{Student}) - \pi_{\text{SID}}\ (\textbf{Apply})$$

- E.g. 2 “IDs and Names of all students who didn’t apply anywhere”

$$\pi_{\text{sName,SID}}((\pi_{\text{SID}}(\textbf{Student}) - \pi_{\text{SID}}\ (\textbf{Apply})) \bowtie \text{Student})$$

  - (this is called join back)

Intersection Operator

Better illustrated with example!

- E.g. 1 “Names that are both a University name and Student name”

$$\pi_{\text{sName}} (\textbf{Student}) \cap \pi_{\text{uName}} (\textbf{Apply})$$

- Intersection does not add expressive power
  - E1 ∩ E2 = E1- (E - E2)
  - E1 ∩ E2 = E1 ⋈ E2
    - (match up all columns that are equal and eliminate duplicates from the columns.)

Rename Operator

- The rename operator has 3 forms. The first one is the most general
  - $\rho R(A1, A2, \ldots An)(E)$.
    - This should be read as: "Evaluate E, and get a relation as a result. Then call the result relation R with attributes A1,…,An." From now on, we can use this schema to describe the result of E.
  - $\rho R(E)$.
    - "Use the same attribute names but change the relation name to
  - $\rho(A1, A2, \ldots An)(E)$.
    - "Use the same relation name but change the attribute names to A1,…,An."

Rename Operator

- 2 main uses
  - Unifies schemas for the Union, Difference and Intersection operators
  - E.g. 1 “List all Student and University names”

$$(\rho C(name)(\pi\ sName(\textbf{Student})) \cup (\rho C(name)(\pi\ uName\ (\textbf{University}))$$

  - Helps to disambiguation in self joins
  - E.g. 2 “Pairs of Universities in same County”

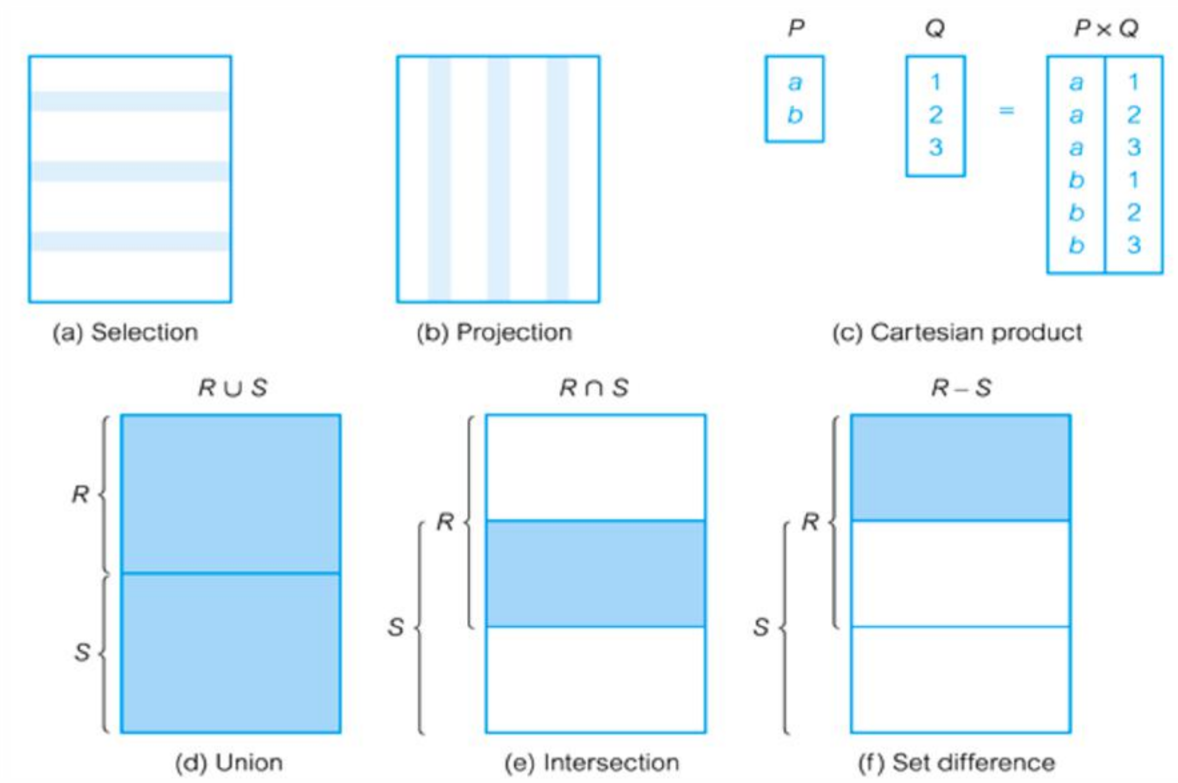
University		
<u>uName</u>	**County**	**Enrollment**
NOTT	Nott/shire	18000
CAM	Cam/shire	22000
UCL	Great/Lon	20000

Rename Operator

E.g. 2 “Pairs of Universities in same County”

- Step 1
  - $\sigma_{c1=c2}$ (ρU1(n1, c1, e1)(University) x ρU2(n2, c2, e2)(University))
  - or
  - ρU1(n1, c, e1)(University) ⋈ ρU2(n2, c, e2)(University)
- Step 2
  - $\sigma_{n1 \neq n2}$ (ρU1(n1, c, e1)(University) ⋈ ρU2(n2, c, e2)(University))
- Step 3
  - $\sigma_{n1>n2}$ (ρU1(n1, c, e1)(University) ⋈ ρU2(n2, c, e2)(University))

Relational Algebra Operations Visualised



(a) Selection (b) Projection (c) Cartesian product

(d) Union (e) Intersection (f) Set difference

Question

Assume that we are given relations R(A,C) and S(B,C,D):

R	
A	**C**
3	3
6	4
2	3
3	5
7	1

S		
B	**C**	**D**
5	1	6
1	5	8
4	3	9

Compute the natural join of R and S. Which of the following tuples is in the result? Assume each tuple has schema (A,B,C,D).

1) (5, 1, 6, 4)
2) (6, 4, 3, 9)
3) (3, 3, 5, 8)
4) (2, 4, 3, 9)

Questions?