

Database Systems and Web (15B11CI312)

Database Systems and Web

Lecture 17: Relational Algebra

Contents to be covered

- Compound Operators
- Joins
- Division
- Sql vs Relational algebra

Compound Operator: Join

Joins are compound operators involving cross product, selection, and (sometimes) projection.

Most common type of join is a “natural join” (often just called “join”). $R \bowtie S$ conceptually is:

- Compute $R \times S$
- Select rows where attributes that **appear in both relations** have equal values
- Project all unique attributes and one copy of each of the common ones.

Natural Join Example

R1

**S
1**

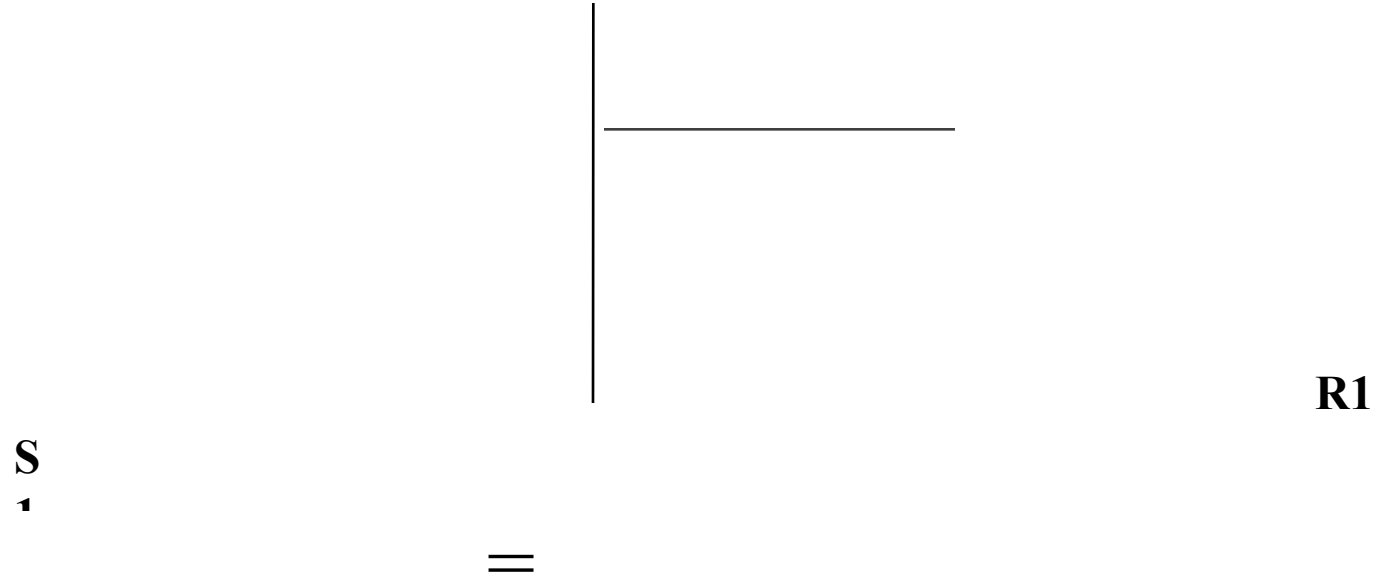
R1 \bowtie **S1**

=

Other Types of Joins

- Condition Join (or “theta-join”):
- *Result schema* same as that of cross-product.
- May have fewer tuples than cross-product.
- Equi-Join: Special case: condition c contains only conjunction of *equalities*.
- Self Join

“Theta” Join Example



Outer Join

An extension of the join operation that **avoids loss of information**.

Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result of the join.

Uses *null* values:

- *null* signifies that the value is unknown or does not exist
- All comparisons involving *null* are (roughly speaking) **false** by definition.

Outer Join – Example

Relation *loan*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Relation *borrower*

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

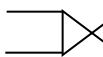
<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

Relation *loan*

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

Relation *borrower*

Left Outer Join

loan  *Borrower*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	<i>null</i>

Source: Fundamentals of database systems / Ramez Elmasri, Shamkant B. Navathe.—6th ed, Pearson Publications

Right Outer Join

loan ⋈_{right} *borrower*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	<i>null</i>	<i>null</i>	Hayes

Full Outer Join

Loan ⋈_{full} *borrower*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	<i>null</i>
L-155	<i>null</i>	<i>null</i>	Hayes

Compound Operator: Division

Goal: Produce the tuples in one relation, R that match *all* tuples in another relation, S.

For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S.

Useful for expressing “**for all**” queries like:
Find s.ids of sailors who have reserved all boats.

Division cont.

Takes two relations, one binary and one unary, and returns a relation consisting of all values of one attribute of the binary relation that match (in the other attribute) all values in the unary relation.

Examples of Division A/B

B
 1

B
 2

B
 3

A

$A/B1$

$A/B2$

$A/B3$

Division - Example

- List the Ids of students who have passed all courses that were taught in odd sem 2018

- *Numerator:*

- *StudId* and *CrsCode* for every course passed by every student:

$$\text{Course} \quad \pi_{\text{StudId}, \text{CrsCode}} (\sigma_{\text{Grade} \neq 'F'} (\text{Transcript}))$$

- *Denominator:* ←

- *CrsCode* of all courses taught in spring 2000

$$\text{Spring} \quad \pi_{\text{CrsCode}} (\sigma_{\text{Semester} = 'S2018'} (\text{Teaching}))$$

- Result is *numerator* / *denominator*

Result Course ÷ Spring

←

Example Queries

Find all customers who have an account at all branches located in Brooklyn city.

$$\Pi_{customer-name, branch-name} (depositor \quad account) \quad \bowtie \quad \div \Pi_{branch-name} (\sigma_{branch-city = \text{“Brooklyn”}} (branch))$$

Modification of the Database

The content of the database may be modified using the following operations:

- Deletion
- Insertion
- Updating

All these operations are expressed using the assignment operator.

Deletion

A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database.

Can delete only whole tuples; cannot delete values on only particular attributes

A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where r is a relation and E is a relational algebra query.

Deletion Examples

Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch-name = "Perryridge"}(account)$$

Delete all loan records with amount in the range of 0 to 50

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$$

Insertion

To insert data into a relation, we either:

- specify a tuple to be inserted
- write a query whose result is a set of tuples to be inserted

in relational algebra, an insertion is expressed by:

$$r \leftarrow r \cup E$$

where r is a relation and E is a relational algebra expression.

The insertion of a single tuple is expressed by letting E be a constant relation containing one tuple.

Insertion Examples

Insert information in the database specifying that *Smith* has \$1200 in account A-973 at the Perryridge branch.

$$\begin{aligned} \text{account} &\leftarrow \text{account} \cup \{(\text{"Perryridge"}, \text{A-973}, 1200)\} \\ \text{depositor} &\leftarrow \text{depositor} \cup \{(\text{"Smith"}, \text{A-973})\} \end{aligned}$$

Updating

A mechanism to change a value in a tuple without changing *all* values in the tuple

Use the generalized projection operator to do this task

$$r \leftarrow \Pi_{F1, F2, \dots, FI} (r)$$

Update Examples

Make interest payments by increasing all balances by 5 percent.

$$account \leftarrow \Pi_{AN, BN, BAL * 1.05}(account)$$

where AN , BN and BAL stand for *account-number*, *branch-name* and *balance*, respectively.

Pay all accounts with balances over \$10,000, 6 percent interest and pay all others 5 percent

$$account \leftarrow \begin{array}{l} \Pi_{AN, BN, BAL * 1.06}(\sigma_{BAL > 10000}(account)) \\ \cup \Pi_{AN, BN, BAL * 1.05}(\sigma_{BAL \leq 10000}(account)) \end{array}$$

SQL VS RELATIONAL ALGEBRA

Questions:

Q1. **FACULTY(name, dpt, salary)**

CHAIR(dpt, name)

- Find the salaries of department chairs using RA and SQL.

RA:

SQL: `SELECT FACULTY.dpt, FACULTY.salary
FROM FACULTY, CHAIR
WHERE FACULTY.name = CHAIR.name AND
FACULTY.dpt = CHAIR.dpt`

Reserves

Sailors

Find names of sailors who've reserved boat id 103

Solution 1:

Solution 2:

Find names of sailors who've reserved a red boat

Information about boat color only available in Boats; so need an extra join:

works(person name, company name, salary);

lives(person name, street, city);

located in(company name, city); managers(person name, manager name);

- Find the names of the persons who work for company 'FBC'.

RA:

SQL: Select person_name From works

Where company_name = 'FBC'

works(person name, company name, salary);

lives(person name, street, city);

located in(company name, city);

managers(person name, manager name);

- List the names of the persons who work for company 'FBC' along with the cities they live in.

RA:

SQL: Select lives.person_name, city
From works, lives
Where company_name = 'FBC' and
works.person_name = lives.person_name

works(person-name, company name, salary);

lives(person-name, street, city);

located in(company-name, city);

managers(person-name, manager name);

- Find the names of the persons who live and work in the same city.

RA:

SQL:

Select person_name

From works, lives, locatedin

Where works.person-name =lives.person-name

and works.company name=located_in.company-name

and located_in.city = lives.city

works(person-name, company name, salary);

lives(person-name, street, city);

located in(company-name, city);

managers(person-name, manager name);

- Find the persons whose salaries are more than the salary of everybody who work for company 'SBC'.

RA:

SQL:

```
Select person_name  
From works  
Where salary > all (Select salary  
From works  
Where Company_name = 'SBC')
```


References

These Slides were prepared using following resources:

Books:

- A First Course in Database Systems, by J. Ullman and J. Widom
- Fundamentals of Database Systems, by R. Elmasri and S. Navathe