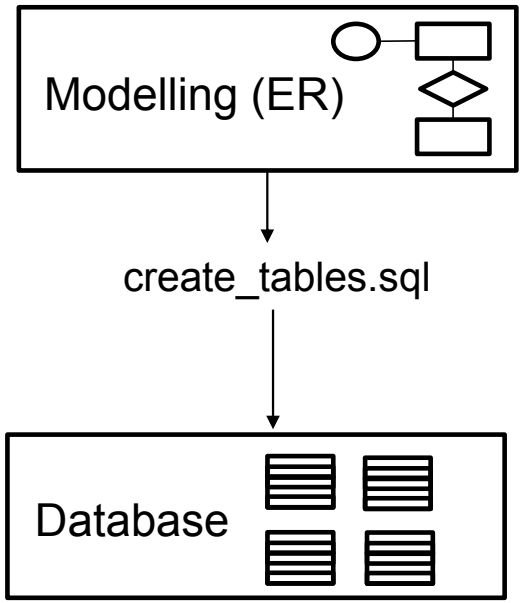
INFO20003 Database Systems

Dr Renata Borovica-Gajic

Lecture 07
Relational Algebra

Week 4

What we have done so far



SQL:

- Language for data manipulation
- Allow to create/delete tables, add/update/remove data, etc

Introduced next time

Relational algebra:

- The theory behind SQL
- Makes sure that SQL produces correct answers
- Inputs/outputs are relations

relational algebra is closed Today

How do we manipulate with relations?

Relational Algebra: 5 Basic Operations

1. **Selection** (σ): Selects a subset of *rows* from relation (horizontal filtering).
2. **Projection** (π): Retains only wanted *columns* from relation (vertical filtering).
3. **Cross-product** (x): Allows us to combine two relations.
4. **Set-difference** (–): Tuples in one relation, but not in the other.
5. **Union** ($\cup$): Tuples in one relation and/or in the other.

creat one bigger relation out of these two

Each operation returns a relation, operations can be composed

Coverage : Relational Algebra

- Selection & Projection
- Union, Set Difference & Intersection
- Cross product & Joins
- Examples

Readings: Chapter 4, Ramakrishnan & Gehrke, Database Systems

Example Instances

associative entity

Reserves (R1)

sid	bid	day
22	101	10/10/96
58	103	11/12/96

→ descriptive attribute

many-to-many relationship

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Sailors 1 (S1)

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Sailors 2 (S2)

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Relational Algebra

- Selection & Projection — operation of a single table
- Union, Set Difference & Intersection
- Cross product & Joins
- Examples

Readings: Chapter 4, Ramakrishnan & Gehrke, Database Systems

Projection

- Retains only attributes that are in the *projection list*
- *Schema* of result:
  - Only the fields in the projection list, with the same names that they had in the input relation
- Projection operator has to *eliminate duplicates*
  - How do they arise? Why remove them?
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it

Projection Examples

1. *Find ages of sailors :* $\pi_{age}(S2)$
2. *Find names and rating of sailors :* $\pi_{sname,rating}(S2)$

projection attribute list
$\pi_{age}(S2)$
input

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

$\pi_{sname,rating}(S2)$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{sname,rating}(S2)$

age
35.0
55.5

Removed duplicates

$\pi_{age}(S2)$

Selection (σ)

- Selects rows that satisfy *selection condition*
- Result is a relation. *Schema* of the result is same as that of the input relation.
- Do we need to do duplicate elimination? no, don't have duplicate in the original table
- **Example:**

  Find sailors whose rating is above 8

by taking subset, can't have duplicate.

sid	sname	rating	age
28	yuppy	9	35.0
~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~44~~	~~guppy~~	~~5~~	~~35.0~~
58	rusty	10	35.0

$$\sigma_{rating>8}(S2)$$ (condition)

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

$$\sigma_{rating>8}(S2)$$

Conditions

- Conditions are standard arithmetic expressions

  >, <, >=, <=, =, !=

- Conditions are combined with AND/OR clauses

  And: $\wedge$

  Or: $\vee$

- **Example**:

Find sailors whose rating is above 8 and who are younger than 50

$$\sigma_{rating>8 \wedge age<50}\ (S2)$$

Selection & Projection

- Operations can be combined
- Select rows that satisfy *selection condition* & retain only *certain attributes (columns)*
- **Example:**
  Find names and rating of sailors whose rating is above 8

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

sname	rating
yuppy	9
rusty	10

start from innermost bracket.

$\sigma_{rating>8}(S2)$.

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

Relational Algebra

- Selection & Projection
- Union, Set Difference & Intersection
- Cross product & Joins
- Examples

Readings: Chapter 4, Ramakrishnan & Gehrke, Database Systems

Union and Set-Difference

- **Union**: Combines both relations together
- **Set-difference:** Retains rows of one relation that do not appear in the other relation

- These operations take two input relations, which must be *union-compatible*:
  - Same number of fields
  - Corresponding fields have the same type

Union

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

Duplicates are removed

S2 ∪ S1 -

when there comes to relation they are set (order between relations doesn't matter)

Set Difference

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

appear in S1 but not S2.

Set Difference

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

$S2 - S1$

Set-difference is not symmetrical

Compound Operator: Intersection

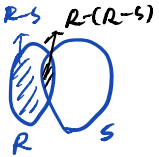
- In addition to the 5 basic operators, there are several additional "Compound Operators"
  - –These add no computational power to the language, but are useful shorthands
  - –Can be expressed solely with the basic operations

- **Intersection** retains rows that appear in *both* relations
- Intersection takes two input relations, which must be *union-compatible*

# of attributes
attributes with same type

- Q: How to express it using basic operators?

$$R \cap S = R - (R - S)$$



Intersection

Example:

Find sailors who appear in both relations S1 and S2

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Relational Algebra

- Selection & Projection
- Union, Set Difference & Intersection
- Cross product & Joins
- Examples

Readings: Chapter 4, Ramakrishnan & Gehrke, Database Systems

Cross Product

- **Cross product** combines two relations:
  - Each row of one input is merged with each row from another input
  - Output is a new relation with all attributes of *both* inputs
  - X is used to denote cross-product
- Example: S1 x R1
  - Each row of S1 paired with each row of R1
- Question: How many rows are in the result?
  - A: card(S1)*card(R1)

cardinality
# of record

Cross Product Example

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

S1 X R1 =

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Cross Product: Conflicting names

- *Result schema* has one field per field of S1 and R1, with field names "inherited" if possible.
  - –May have *a naming conflict*, i.e. both S1 and R1 have a field with the same name (e.g. sid).
  - –In this case, can use the *renaming operator*:

$$\rho\,(C(1 \rightarrow sid1, 5 \rightarrow sid2),\ S1 \times R1)$$

renaming operator

Result relation name

C

sid1	sname	rating	age	sid2	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

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 a cross product that matches rows where attributes that appear in both relations have equal values (and we omit duplicate attributes).

- To obtain cross product a DBMS must:
  1. Compute R X S
  2. Select rows where attributes that appear in both relations have equal values
  3. Project all unique attributes and one copy of each of the common ones.

Natural Join Example

Example:

Find all sailors (from relation S1) who have reserved a boat

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

S1 ⋈ R1 =

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

Natural Join Example

1

S1 X R1 =

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Natural Join Example

1

S1 X R1 =

2 σ

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35.0	58	103	11/12/96

Natural Join Example

1

S1 X R1 =

2 σ

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35.0	58	103	11/12/96

π 3

S1 ⋈ R1=

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

Other Types of Joins

- **Condition Join (or theta-join)** is a cross product with a condition.

$$R \bowtie_c S = \sigma_c (R \times S)$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

  - Result schema is the same as that of cross-product

- **Equi-Join** is a special case of condition join, where condition *c* contains only *equalities* (e.g. S1.sid = R1.sid)
  - Is this then a natural join? What is different?

implicit have projection (keep only one attribute when duplication)

look for equality of attribute in both table

equi-join is flexible, need equality (eg. S1.sid=R1.bid)

Relational Algebra

- Selection & Projection
- Union, Set Difference & Intersection
- Cross product & Joins
- Examples

Readings: Chapter 4, Ramakrishnan & Gehrke, Database Systems

Let's try it...

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Question

Find names of sailors who have reserved boat #103

$\pi_{sname}(\sigma_{bid=103}(S \times R))$ $\pi_{sname}(\sigma_{bid=103}(Sailors \bowtie Reserves))$

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Solution 1: ✓ $\pi_{sname}((\sigma_{bid=103}\text{Reserves}) \bowtie Sailors)$

Solution 2: ✓ $\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))$

Question

$\rho(S1(1 \to sid1, 2 \to sname1, 3 \to rating1, 4 \to age1), sailors)$
$\rho(S2(1 \to sid2, 2 \to sname2, 3 \to rating2, 4 \to age2), sailors).$

$\pi_{sname1, sname2}(S1 \bowtie age1 > age2 \wedge rating1 < rating\ S2)$

- Find all pairs of sailors in which the older sailor has a lower rating

$\rho(S1(1 \to \ldots), sailor1)$

$\rho(S0(1 \to \ldots), sailor2)$

$\sigma(age1 > age2 \wedge rating1 < rating2\ (\rho(C(1 \to sid1, 2 \to sname1, 3 \to rating1, 4 \to age1, 5 \to sid2, 6 \to sname2, 7 \to rating2, 8 \to age2), sailor \times sailor)))$

What's Examinable

- Relational Algebra Operations: Selection, Projection, Union, Set, Difference, Intersection, **JOINS**…
- Draw different queries with Relational Algebra operations

Next Lecture

- Introducing SQL