

# Databases

Lecture 7

Relational Algebra

- query languages in the relational model
  - relational algebra and calculus - formal query languages with a significant influence on SQL
    - relational algebra
      - queries are specified in an operational manner
    - relational calculus
      - queries describe the desired answer, without specifying how it will be computed (declarative)
  - not expected to be Turing complete
  - not intended for complex calculations
  - provide efficient access to large datasets
  - allow optimizations

- relational algebra
  - used by DBMSs to represent query execution plans
  - a relational algebra query:
    - is built using a collection of operators
    - describes a step-by-step procedure for computing the result set
    - is evaluated on the input relations' instances
    - produces an instance of the output relation
  - every operation returns a relation, so operators can be composed; the algebra is closed
  - the result of an algebra expression is a relation, and a relation is a set of tuples
- relational algebra on bags (multisets) - duplicates are not eliminated

## Conditions

- conditions that can be used in several algebraic operators
- similar to the SELECT filter conditions

### 1. *attribute\_name relational\_operator value*

- *value* - attribute name, expression

### 2. *attribute\_name IS [NOT] IN single\_column\_relation*

- a relation with one column can be considered a set
- the condition tests whether a value belongs to a set

### 3. *relation {IS [NOT] IN | = | <>} relation*

- the relations in the condition must be union-compatible

## Conditions

4. *(condition)*

*NOT condition*

*condition<sub>1</sub> AND condition<sub>2</sub>*

*condition<sub>1</sub> OR condition<sub>2</sub>,*

where *condition*, *condition<sub>1</sub>*, *condition<sub>2</sub>* are conditions of type 1-4.

## Operators in the Algebra

- equivalent SELECT statements can be specified for the relational algebra expressions
- *selection*
  - notation:  $\sigma_C(R)$
  - resulting relation:
    - schema:  $R$ 's schema
    - tuples: records in  $R$  that satisfy condition  $C$
  - equivalent SELECT statement

```
SELECT *  
FROM R  
WHERE C
```

- *projection*
  - notation:  $\pi_{\alpha}(R)$
  - resulting relation:
    - schema: attributes in  $\alpha$
    - tuples: every record in  $R$  is projected on  $\alpha$
  - $\alpha$  can be extended to a set of expressions, specifying the columns of the relation being computed
  - equivalent SELECT statement

```
SELECT DISTINCT  $\alpha$ 
FROM R
```

```
SELECT  $\alpha$ 
FROM R                -- algebra on bags
```

- *cross-product*
  - notation:  $R_1 \times R_2$
  - resulting relation:
    - schema: the attributes of  $R_1$  followed by the attributes of  $R_2$
    - tuples: every tuple  $r_1$  in  $R_1$  is concatenated with every tuple  $r_2$  in  $R_2$
- equivalent SELECT statement

```
SELECT *  
FROM R1 CROSS JOIN R2
```



- *union, set-difference, intersection*
  - notation:  $R_1 \cup R_2$ ,  $R_1 - R_2$ ,  $R_1 \cap R_2$
  - $R_1$  and  $R_2$  must be union-compatible:
    - same number of columns
    - corresponding columns, taken in order from left to right, have the same domains
  - equivalent SELECT statements

SELECT *	SELECT *	SELECT *
FROM R1	FROM R1	FROM R1
UNION	EXCEPT	INTERSECT
SELECT *	SELECT *	SELECT *
FROM R2	FROM R2	FROM R2

-- algebra on bags: SELECT statements that don't eliminate duplicates (e.g., UNION ALL)

- join operators
  - *condition join* (or *theta join*)
    - notation:  $R_1 \otimes_{\Theta} R_2$
    - result: the records in the cross-product of  $R_1$  and  $R_2$  that satisfy a certain condition
  - definition  $\Rightarrow R_1 \otimes_{\Theta} R_2 = \sigma_{\Theta}(R_1 \times R_2)$
  - equivalent SELECT statement
 

```
SELECT *
FROM R1 INNER JOIN R2 ON  $\Theta$ 
```

- join operators
  - *natural join*
    - notation:  $R_1 * R_2$
    - resulting relation:
      - schema: the union of the attributes of the two relations (attributes with the same name in  $R_1$  and  $R_2$  appear once in the result)
      - tuples: obtained from tuples  $\langle r_1, r_2 \rangle$ , where  $r_1$  in  $R_1$ ,  $r_2$  in  $R_2$ , and  $r_1$  and  $r_2$  agree on the common attributes of  $R_1$  and  $R_2$
  - let  $R_1[\alpha]$ ,  $R_2[\beta]$ ,  $\alpha \cap \beta = \{A_1, A_2, \dots, A_m\}$ ; then:
 
$$R_1 * R_2 = \pi_{\alpha \cup \beta} (R_1 \otimes_{R_1.A_1=R_2.A_1 \text{ AND } \dots \text{ AND } R_1.A_m=R_2.A_m} R_2)$$
  - equivalent SELECT statement
 

```
SELECT *
FROM R1 NATURAL JOIN R2
```

- join operators
  - *left outer join*
    - notation (in these notes):  $R_1 \bowtie_C R_2$
    - resulting relation:
      - schema: the attributes of  $R_1$  followed by the attributes of  $R_2$
      - tuples: tuples from the condition join  $R_1 \bowtie_C R_2$  + the tuples in  $R_1$  that were not used in  $R_1 \bowtie_C R_2$  combined with the *null* value for the attributes of  $R_2$
  - equivalent SELECT statement

```
SELECT *
FROM R1 LEFT OUTER JOIN R2 ON C
```

- join operators
  - *right outer join*
    - notation:  $R_1 \bowtie_C R_2$
    - resulting relation:
      - schema: the attributes of  $R_1$  followed by the attributes of  $R_2$
      - tuples: tuples from the condition join  $R_1 \Join_C R_2$  + the tuples in  $R_2$  that were not used in  $R_1 \Join_C R_2$  combined with the *null* value for the attributes of  $R_1$
  - equivalent SELECT statement

```
SELECT *
FROM R1 RIGHT OUTER JOIN R2 ON C
```

- join operators
  - *full outer join*
    - notation:  $R_1 \bowtie_c R_2$
    - resulting relation:
      - schema: the attributes of  $R_1$  followed by the attributes of  $R_2$
      - tuples:
        - tuples from the condition join  $R_1 \otimes_c R_2$  +
        - the tuples in  $R_1$  that were not used in  $R_1 \otimes_c R_2$  combined with the *null* value for the attributes of  $R_2$  +
        - the tuples in  $R_2$  that were not used in  $R_1 \otimes_c R_2$  combined with the *null* value for the attributes of  $R_1$
  - equivalent SELECT statement
 

```
SELECT *
FROM R1 FULL OUTER JOIN R2 ON C
```

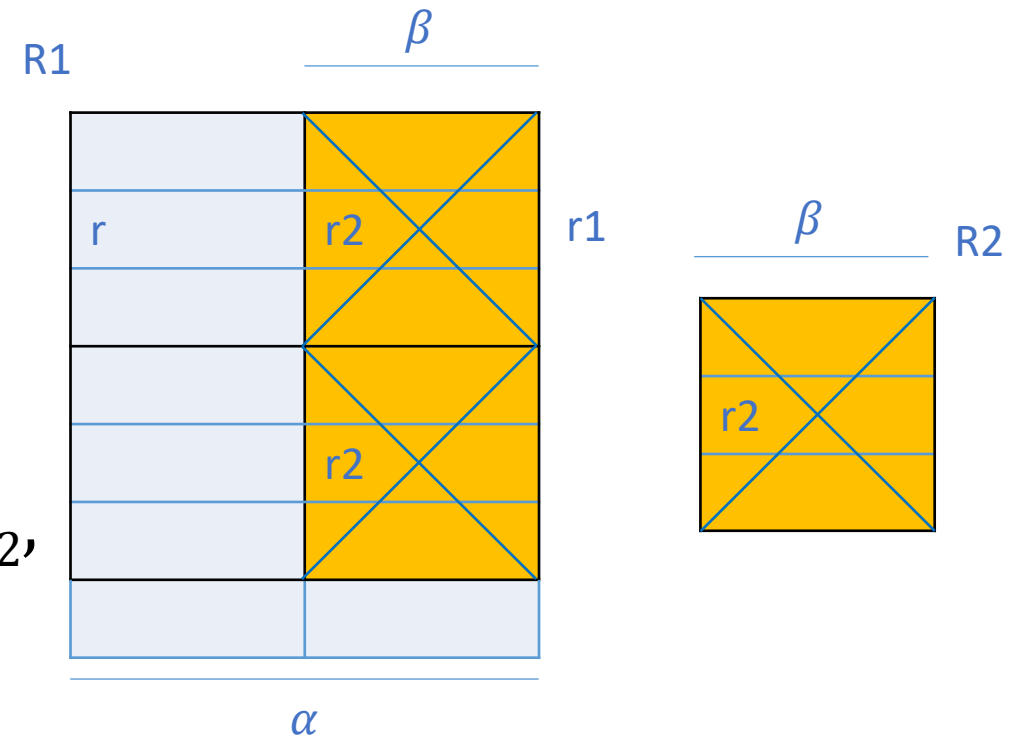
- join operators
  - *left semi join*
    - notation:  $R_1 \triangleright R_2$
    - resulting relation:
      - schema:  $R_1$ 's schema
      - tuples: the tuples in  $R_1$  that are used in the natural join  $R_1 * R_2$

- join operators
  - *right semi join*
    - notation:  $R_1 \Join R_2$
    - resulting relation:
      - schema:  $R_2$ 's schema
      - tuples: the tuples in  $R_2$  that are used in the natural join  $R_1 * R_2$



- *division*

- notation:  $R_1 \div R_2$
- $R_1[\alpha], R_2[\beta], \beta \subset \alpha$
- resulting relation:
  - schema:  $\alpha - \beta$
  - tuples: a record  $r \in R_1 \div R_2$  iff  $\forall r_2 \in R_2, \exists r_1 \in R_1$  such that:
    - $\pi_{\alpha-\beta}(r_1) = r$
    - $\pi_{\beta}(r_1) = r_2$
  - i.e., a record  $r$  belongs to the result if in  $R_1$   $r$  is concatenated with every record in  $R_2$



- see lecture examples (at the board) with algebra queries:
  - selection
  - projection
  - division
  - selection, projection
  - natural join, selection, projection
  - different algebra expressions producing the same result (optimization - reducing the size of intermediate relations)

## An Independent Subset of Operators

- independent set of operators M:
  - eliminating any operator  $op$  from M: there will be a relation that can be obtained using M's operators, but cannot be obtained with the operators in  $M - \{op\}$
- for the previously described query language, with operators:  
 $\{\sigma, \pi, \times, \cup, -, \cap, \otimes, *, \ltimes, \rtimes, \bowtie, \triangleright, \triangleleft, \div\}$   
an independent set of operators is  $\{\sigma, \pi, \times, \cup, -\}$
- the other operators are obtained as follows (some expressions have already been introduced):
  - $R_1 \cap R_2 = R_1 - (R_1 - R_2)$
  - $R_1 \otimes_C R_2 = \sigma_C(R_1 \times R_2)$

- the other operators are obtained as follows (some expressions have already been introduced):

- $R_1[\alpha], R_2[\beta], \alpha \cap \beta = \{A_1, A_2, \dots, A_m\}$ , then:

$$R_1 * R_2 = \pi_{\alpha \cup \beta}(R_1 \otimes_{R_1.A_1=R_2.A_1 \text{ AND } \dots \text{ AND } R_1.A_m=R_2.A_m} R_2)$$

- $R_1[\alpha], R_2[\beta], R_3[\beta] = \{(null, \dots, null)\}, R_4[\alpha] = \{(null, \dots, null)\}$

$$R_1 \bowtie_C R_2 = (R_1 \otimes_C R_2) \cup (R_1 - \pi_\alpha(R_1 \otimes_C R_2)) \times R_3$$

$$R_1 \bowtie_C R_2 = (R_1 \otimes_C R_2) \cup R_4 \times (R_2 - \pi_\beta(R_1 \otimes_C R_2))$$

$$R_1 \bowtie_C R_2 = (R_1 \bowtie_C R_2) \cup (R_1 \bowtie_C R_2)$$

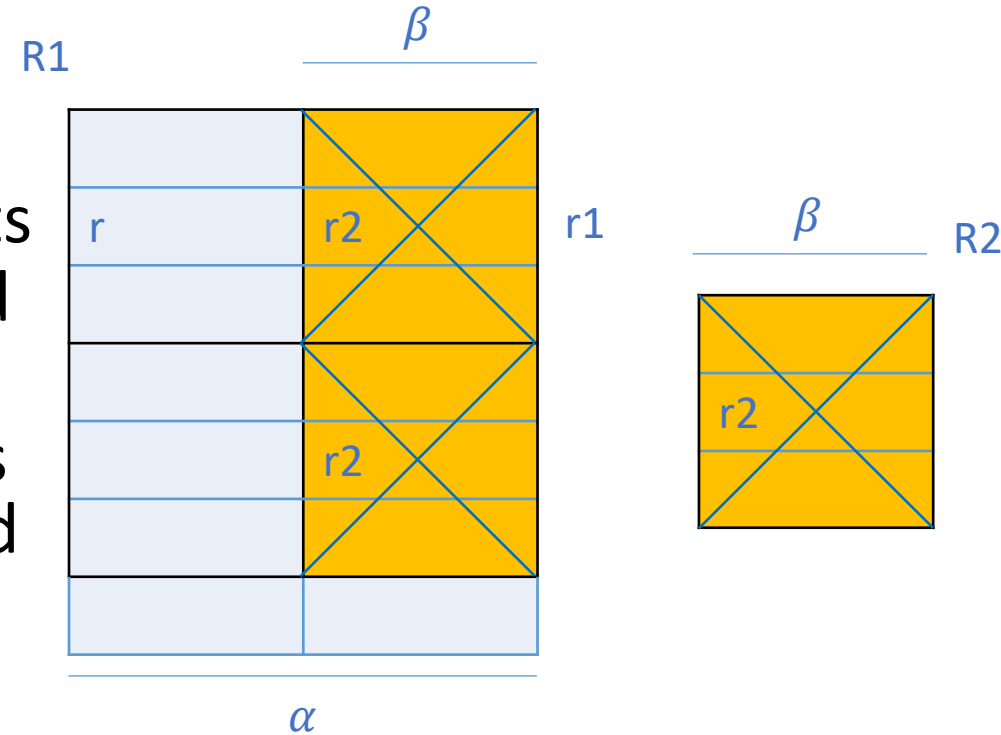
- $R_1[\alpha], R_2[\beta]$

$$R_1 \triangleright R_2 = \pi_\alpha(R_1 * R_2)$$

$$R_1 \triangleleft R_2 = \pi_\beta(R_1 * R_2)$$

- the other operators are obtained as follows (some expressions have already been introduced):
  - if  $R_1[\alpha]$ ,  $R_2[\beta]$ ,  $\beta \subset \alpha$ , then  $r \in R_1 \div R_2$  iff  $\forall r_2 \in R_2, \exists r_1 \in R_1$  such that:  
 $\pi_{\alpha-\beta}(r_1) = r$  and  $\pi_\beta(r_1) = r_2$   
 $\Rightarrow r$  is in  $\pi_{\alpha-\beta}(R_1)$ , but not all the elements in  $\pi_{\alpha-\beta}(R_1)$  are in the result
  - $(\pi_{\alpha-\beta}(R_1)) \times R_2$  contains all the elements with one part in  $\pi_{\alpha-\beta}(R_1)$  and the second part in  $R_2$
  - to obtain values that are disqualified,  $R_1$  is subtracted from the obtained relation, and the result is projected on  $\alpha - \beta$
  - the final expression:

$$R_1 \div R_2 = \pi_{\alpha-\beta}(R_1) - \pi_{\alpha-\beta}((\pi_{\alpha-\beta}(R_1)) \times R_2 - R_1)$$



- the *renaming* operator

$$\rho(R'(A_1 \rightarrow A_1', A_2 \rightarrow A_2', A_3 \rightarrow A_3'), E)$$

- E - relational algebra expression
- the result, relation R', has the same tuples as the result of E
- attributes A<sub>1</sub>, A<sub>2</sub>, and A<sub>3</sub> are renamed to A<sub>1</sub>', A<sub>2</sub>', and A<sub>3</sub>', respectively

\* the next examples use the statements below:

- assignment

$R[\text{list}] := \text{expression}$

- the expression's result (a relation) is assigned to a variable ( $R[\text{list}]$ ), specifying the name of the relation [and the names of its columns]

- eliminating duplicates from a relation

$\delta(R)$

- sorting records in a relation

$S_{\{\text{list}\}}(R)$

- grouping

$Y_{\{\text{list1}\} \text{ group by } \{\text{list2}\}}(R)$

- $R$ 's records are grouped by the columns in *list2*
- *list1* (that can contain aggregate functions) is evaluated for each group of records

students [id, name, sgroup, gpa, dob]

groups [id, year, program]

schedule [day, starthour, endhour, activtype, room, sgroup,  
facultym\_id]

faculty\_members [id, name]

**1. The names of students in a given group:**

$$R := \pi_{\{name\}} \left( \sigma_{sgroup='222'}(students) \right)$$

SELECT name

FROM students

WHERE sgroup='222'



## 2. The students in a given program (alphabetical list, by groups):

$$G := \pi_{\{id\}} \left( \sigma_{program='IG'}(groups) \right)$$
$$R := S_{\{sgroup, name\}} \left( \sigma_{sgroup \text{ is in } G}(students) \right)$$

```
SELECT *
FROM students
WHERE sgroup IN
    (SELECT id
     FROM groups
     WHERE program='IG')
ORDER BY sgroup, name
```

students [id, name, sgroup, gpa, dob]  
groups [id, year, program]  
schedule [day, starthour, endhour, activtype, room,  
sgroup, facultym\_id]  
faculty\_members [id, name]

3. The number of students in every group of a given program:

$$ST := \sigma_{sgroup \text{ is in } \left( \pi_{\{id\}} \left( \sigma_{program='IG'}(groups) \right) \right)}(students)$$

$$NR := \gamma_{\{sgroup, count(*)\}} \text{ group by } \{sgroup\} (ST)$$

```
SELECT sgroup, COUNT(*)
FROM (SELECT *
      FROM students
      WHERE sgroup IN
        (SELECT id
         FROM groups
         WHERE program='IG')
      ) t
GROUP BY sgroup
```

students [id, name, sgroup, gpa, dob]  
groups [id, year, program]  
schedule [day, starthour, endhour, activtype, room,  
sgroup, facultym\_id]  
faculty\_members [id, name]

4. A student's schedule (the student is given by name):

$$T := \sigma_{sgroup \text{ is in } \left( \pi_{\{sgroup\}} \left( \sigma_{name='Ionescu M. Razvan'}(students) \right) \right)}(schedule)$$

5. The number of hours per week for every group:

$$F(no, sgroup) := \pi_{\{endhour - starthour, sgroup\}}(schedule)$$
$$NoHours(sgroup, nohours) := \gamma_{\{sgroup, sum(no)\}} \text{ group by } \{sgroup\}(F)$$

students [id, name, sgroup, gpa, dob]

groups [id, year, program]

schedule [day, starthour, endhour, activtype, room, sgroup, facultym\_id]

faculty\_members [id, name]

6. The faculty members (their names) who teach a given student:

$$A := (\sigma_{name='Ionescu M. Razvan'}(students)) \otimes_{students.sgroup=schedule.sgroup} schedule$$
$$B := \pi_{\{faculty\_id\}}(A)$$
$$C := faculty\_members \otimes_{faculty\_members.id=B.facultym\_id} B$$
$$D := \pi_{\{name\}}(C)$$

students [id, name, sgroup, gpa, dob]

groups [id, year, program]

schedule [day, starthour, endhour, activtype, room, sgroup, facultym\_id]

faculty\_members [id, name]

7. The faculty members with no teaching assignments (i.e., not on the schedule):

$$C := \pi_{\{name\}}(faculty\_members) - \pi_{\{name\}}(schedule \otimes_{schedule.facultym\_id=faculty\_members.id} faculty\_members)$$

\* Is there a problem if two different faculty members have the same name?

students [id, name, sgroup, gpa, dob]

groups [id, year, program]

schedule [day, starthour, endhour, activtype, room, sgroup, facultym\_id]

faculty\_members [id, name]

8. Students with school activities on every day of the week (all days with school activities considered):

$$A := \delta \left( \pi_{\{day\}}(schedule) \right)$$

$$B := students \otimes_{students.sgroup=schedule.sgroup} schedule$$

$$C := \delta \left( \pi_{\{name, day\}}(B) \right)$$

$$D := C \div A$$

\* Is there a problem if two different students have the same name?

students [id, name, sgroup, gpa, dob]

groups [id, year, program]

schedule [day, starthour, endhour, activtype, room, sgroup, facultym\_id]

faculty\_members [id, name]

# References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra02] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (3rd Edition), McGraw-Hill, 2002
- [Da03] DATE, C.J., An Introduction to Database Systems (8<sup>th</sup> Edition), Addison-Wesley, 2003
- [Ga09] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book (2nd Edition), Pearson Education, 2009
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2<sup>nd</sup> Edition), Prentice Hall, 1996
- [Ra02S] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, Slides for the 3<sup>rd</sup> Edition,  
<http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html>
- [Si11] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts (6th Edition), McGraw-Hill, 2011
- [Si19S] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts, Slides for the 7th Edition, <http://codex.cs.yale.edu/avi/db-book/>
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems,  
<http://infolab.stanford.edu/~ullman/fcdb.html>