Lecture 7

Relational Algebra

Query Language for the Relational Model

- for handling and retrieving the data from a database
- the Relational Model offers support for simple and powerful query languages
  - formal and based on logic
  - allow the optimization
- Query Language NOT THE SAME WITH Programming Language
  - are not used for complex calculus
  - offers a simple and efficient way to access big sets of data
  - not expected to be Turing complete
- 2 query languages with a significant influence on SQL
  - ***Relational Algebra*** – operational and useful for the execution plan (of the queries)
  - ***Relational Calculus*** – allows the users to describe *what* and *not how* to get what they want (non-operational, declarative)

Relational Algebra

- used in the DBMSs to represent the query execution plans
- the query is applied to the instance of the relation and the result of the query represent such an instance of the relation
  - the structure of a relation from a query is fixed
  - the structure of the result that is returned by a query is fixed and determined by the definitions of the structured used in the query language
- each operation returns a relation
- the operators used can be composed
- the result returned by an algebra expression is a relation / table and this relation has a set or records / tuples
- in the relational algebra for the multisets, the duplicates are not eliminated
- the relational algebra query:
  - is organized by using a collection of operators
  - describes each step that is performed in computing the result
  - is evaluated on the input instances of the relations considered
  - return an output instance relation

Relational Algebra

Conditions that can be used with the Relational Algebra

- all the conditions that are used for the algebraic operators
- are similar with the conditions used in the SELECT
  - attribute_name relational_operator value_expression
  - attribute_name IS [NOT] IN column_relation
    - this condition tests if a value belongs to a set of values
  - relation IS [NOT] IN | = | <> relation
    - the relations must be compatible
  - condition
  - NOT condition
  - condition1 AND | OR condition2

Relational Algebra

Operators from Relational Algebra

- The SELECT statements can be also specified by using the expressions from relational algebra

Basic operations

- ***projection*** (π) – eliminates all the un-wished attributes from the relation
- ***selection*** (σ) – selects a subset of tuples from the relation
- ***cross product*** (X) – allows the combination of two relations
- ***set-difference*** (-) – selects the tuples that are in a relation and not in the other relation
- ***union*** (∪) – selects the tuples that are in both of the relations

Additional operations – very useful

- intersection
- join
- division
- rename

Each operation returns a relation / table, and so, the operations can be composed (algebra is *closed)*

Relational Algebra

Projection (π) – eliminates all the un-wished attributes from the relation

- Notation: $\pi_\alpha(R)$
- α can have a set of expressions that specify the name of the columns of the relation R considered (also, can be computed)
- R – the relation considered
- Result set: a relation with all the attributes that are mentioned in α
  - schema: attributes in α
  - tuples: each record from R that is projected on α
- equivalent with

SELECT DISTINCT α **FROM R**	**not (SELECT α FROM R)**

- The Relational Algebra operates on SETS, so there will NOT BE DUPLICATES

Relational Algebra

Projection (π) – example:

$$\pi_{StudentId,\,Grade}(Exam) \iff SELECT\ \boldsymbol{DISTINCT}\ StudentId, Grade\ FROM\ Exam$$

$\pi_{StudentId,\,Grade}$ (

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

) =

StudentId	Grade
4	9
5	10
4	10
6	7
7	9
7	6
7	10

Relational Algebra

Selection (σ) – selects a subset of tuples from the relation, tuples that satisfy a specified condition

- Notation: $\sigma_C(R)$
- C is the condition, called *selective predicate*, that has to be satisfied by the returned tuples from the relation R
- R – the relation considered
- Result set: a relation with all the attributes from the relation R that satisfy the condition C
  - schema: *R* schema
  - tuples: the records from R that satisfy condition C
- equivalent with

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

- condition C has the form: *Term Operator Term*, where *Term* is an attribute or a constant, and *Operator* is a logical operator (<, <=, >, >=, =, !=, …)
- $C1 \wedge \mathrm{C2}$, $C1 \vee \mathrm{C2}$, $\neg C1$ are conditions that contain the operators ∧ (*AND*), ∨ (*OR*), ¬ (*NOT*), and $C1, \mathrm{C2}$ are also conditions

Relational Algebra

Selection (σ)– example:

$$\sigma_{Grade>7}(Exam) \Leftrightarrow SELECT\ DISTINCT * \ FROM\ Exam\ WHERE\ Grade > 7$$

$\sigma_{Grade>7}$ (

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

) =

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
7	22	9
7	11	10

Relational Algebra

Projection (π) and ***Selection*** (σ)

Projection - $\pi_{attribute1, attribute2, \ldots}(Relation)$

$SELECT\ DISTINCT$ attribute1, attribute2, …

$FROM\ Relation$

$WHERE\ Condition$

Selection - $\sigma_{Condition}(Relation)$

Relational Algebra

Projection (π) and ***Selection*** (σ) - composition

$\pi_{StudentId,\, Grade}(\sigma_{Grade>7}(Exam))$

$\Leftrightarrow SELECT\ DISTINCT\ \text{StudentId}, \text{Grade}\ FROM\ Exam\ WHERE\ Grade > 7$

$\pi_{StudentId,\, Grade}(\sigma_{Grade>7}($

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

$)) =$

StudentId	Grade
4	9
5	10
4	10
7	9
7	10

What about $\sigma_{Grade>7}(\pi_{StudentId,\, Grade}(Exam))$? Are equivalent? Is important the order of π and σ?

- σ before π only when it is a subquery

Relational Algebra

Cross product (X) – allows the combination of two relations

- Notation: $R_1 \times R_2$
- 2 relations are combined
- $R_1, \ R_2$ – the relations considered
- Result set: a relation with all the attributes that are in R_1 and R_2, combined
  - schema: the attributes from R_1 followed by the attributes from R_2
  - tuples: every tuple $r_1 \in R_1$ is concatenated with every tuple $r_2 \in R_2$
- equivalent with

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

or

```
SELECT *
FROM R1, R2
```

Relational Algebra

Cross product (X) example: Student[StudentId, Age] and Course[CourseId, NoOfCredits]

$$Student \times Course \Leftrightarrow SELECT\ DISTINCT * \ FROM\ Student\ CROSS\ JOIN\ Course$$

Student

StudentId	Age
4	19
5	19
6	21
7	20

Course

CourseId	NoOfCredits
11	6
12	5

$Student \times Course$

StudentId	Age	CourseId	NoOfCredits
4	19	11	6
4	19	12	5
5	19	11	6
5	19	12	5
6	21	11	6
6	21	12	5
7	20	11	6
7	20	12	5

Relational Algebra

Union, Intersection, Set-Difference

- Notation: $R_1 \cup R_2$, $R_1 \cap R_2$, $R_1 - R_2$
- R_1 and R_2 must be union-compatible
  - same number of columns
  - compatible = the columns has to correspond from left to right and have compatible data types / domain
- equivalent with

$R_1 \cup R_2$	$R_1 \cap R_2$	$R_1 - R_2$
SELECT DISTINCT * FROM R_1 UNION SELECT DISTINCT * FROM R_2	SELECT DISTINCT * FROM R_1 INTERSECT SELECT DISTINCT * FROM R_2	SELECT DISTINCT * FROM R_1 EXCEPT SELECT DISTINCT * FROM R_2

- UNION ALL – does not eliminate the duplicates

Relational Algebra

Union, Intersection, Set-Difference example: Student[StudentId, Age] and Course[CourseId, NoOfCredits]

$$Student \cup Course \Leftrightarrow SELECT\ DISTINCT * \ FROM\ Student\ UNION\ SELECT\ DISTINCT * \ FROM Course$$

$$Student \cap Course \Leftrightarrow SELECT\ DISTINCT * \ FROM\ Student\ INTERSECR\ SELECT\ DISTINCT * \ FROM Course$$

$$Student - Course \Leftrightarrow SELECT\ DISTINCT * \ FROM\ Student\ EXCEPT\ SELECT\ DISTINCT * \ FROM Course$$

Student

StudentId	Age
4	19
5	19
6	21
7	20

Course

CourseId	NoOfCredits
11	6
12	5

$Student \cup Course$

StudentId	Age
4	19
5	19
6	21
7	20
11	6
12	5

$Student \cap Course$

StudentId	Age

$Student - Course$

StudentId	Age
4	19
5	19
6	21
7	20

Relational Algebra

Union, Intersection, Set-Difference

Indentify the tuples from R_1 that are not in R_2

$$R_1 \cap R_2 = \big((R_1 \cup R_2) - (R_1 - R_2)\big) - (R_2 - R_1)$$

Identify the tuples from R_1 AND R_2

Indentify the tuples from R_2 that are not in R_1

Identify the tuples from R_1 OR R_2

Relational Algebra

Join Operator $(\otimes_\theta)$ – or $\boldsymbol{\theta}$ ***(theta) join***

- Notation: $R_1 \otimes_\theta R_2$
- 2 relations are combined with respect to condition θ
- $R_1,\ R_2$ – the relations considered
- Result set: the records from the cross-product of R_1 and R_2 that satisfy the condition θ
- $\boldsymbol{R_1 \otimes_\theta R_2 = \sigma_\theta (R_1 \times R_2)}$
- equivalent with

```
SELECT DISTINCT *
FROM R1 INNER JOIN R2 ON θ
```

Example: Student $\otimes_{Student.StudentId=Exam.StudentId} Exam \Leftrightarrow$

SELECT DISTINCT *	SELECT DISTINCT *
FROM Student, Exam	FROM Student INNER JOIN Exam
WHERE Student.StudentId = Exam.StudentId	ON Student.StudentId = Exam.StudentId

Relational Algebra

Equi Join ($\otimes_{\boldsymbol{E(\theta)}}$)

- Notation: $R_1 \otimes_{E(\theta)} R_2$
- 2 relations are combined with respect to the composed condition $\mathrm{E}(\theta)$
- $R_1,\ R_2$ – the relations considered
- Result set: the records from R_1 and R_2 that satisfy the composed condition $\mathrm{E}(\theta)$: only equalities between the attributes that are in R_1 and R_2 and project only one of the redundant attributes (because are equal)

Relational Algebra

Equi Join **($\otimes_{E(\theta)}$)** – example

$$\text{Student } \otimes_{E(Student.StudentId=Exam.StudentId)} Exam$$

Student

StudentId	Age
4	19
5	19
6	21
7	20

Exam

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

$\text{Student } \otimes_{E(Student.StudentId=Exam.StudentId)} Exam$

StudentId	Age	CourseId	Grade
4	19	11	9
5	19	11	10
4	19	21	10
4	19	22	9
6	21	21	7
7	20	22	9
7	20	21	6
7	20	11	10

Relational Algebra

Natural Join ($*$)

- Notation: $R_1 * R_2$
- 2 relations are combined with respect to the attributes that have the *same name* and display just one of the redundant attributes
- Result set: a relation with all the attributes that are in R_1 and R_2, combined
  - schema: the union of the attributes of the two relations: attributes with the same name in R_1 and R_2 that appear just once in the result set
  - tuples: obtained from the tuples (r_1, r_2), where $r_1 \in R_1$ and $r_2 \in R_2$ and r_1, r_2 agree on the common attributes of R_1 and R_2
- Let $R_1[\alpha], R_2[\beta], \alpha \cap \beta = \{A_1, A_2, \dots, A_n\}$. Then

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

- equivalent with

SELECT DISTINCT *
FROM R_1 NATURAL JOIN R_2

Relational Algebra

Natural Join (∗) – example

Student

StudentId	Age
4	19
5	19
6	21
7	20

Exam

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

$Student * Exam$

StudentId	Age	CourseId	Grade
4	19	11	9
5	19	11	10
4	19	21	10
4	19	22	9
6	21	21	7
7	20	22	9
7	20	21	6
7	20	11	10

Relational Algebra

Left Outer Join ($\ltimes_C$)

- Notation: $R_1 \ltimes_C R_2$
- 2 relations are combined with respect to the attributes that have the *same name* and display just one of the redundant attributes + the attributes from R_1 that have no correspondent in R_2 (*null* value)
- Result set:
  - schema: a relation with the attributes from R_1 followed by the attributes from R_2
  - tuples: tuples from the condition join $R_1 \otimes_C R_2$ + tuples from R_1 that are not used in $R_1 \otimes_C R_2$ combined with the *null* value for the attributes of R_2
- equivalent with

SELECT DISTINCT *
FROM R_1 LEFT OUTER JOIN R_2 ON C

Relational Algebra
Left Outer Join – example

Student

StudentId	Age
4	19
5	19
6	21
7	20
8	21

Exam

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

$$\text{Student} \ltimes_{\text{Student.StudentId}=\text{Exam.StudentId}} \text{Exam}$$

StudentId	Age	CourseId	Grade
4	19	11	9
5	19	11	10
4	19	21	10
4	19	22	9
6	21	21	7
7	20	22	9
7	20	21	6
7	20	11	10
8	21	**NULL**	**NULL**

Relational Algebra

Right Outer Join ($\rtimes_C$)

- Notation: $R_1 \rtimes_C R_2$
- 2 relations are combined with respect to the attributes that have the *same name* and display just one of the redundant attributes + the attributes from R_2 that have no correspondent in R_1 (*null* value)
- Result set:
  - schema: a relation with the attributes from R_1 followed by the attributes from R_2
  - tuples: tuples from the condition join $R_1 \otimes_C R_2$ + tuples from R_2 that are not used in $R_1 \otimes_C R_2$ combined with the *null* value for the attributes of R_1
- equivalent with

SELECT DISTINCT *
FROM R_1 RIGHT OUTER JOIN R_2 ON C

Relational Algebra

Right Outer Join – example

Exam

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

Course

CourseId	NoOfCredits
11	5
21	6
22	5
23	4

$\text{Exam} \rtimes_{\text{Exam.CourseId}=\text{Course.CourseId}} \text{Course}$

StudentId	CourseId	Grade	NoOfCredits
4	11	9	5
5	11	10	5
4	21	10	6
4	22	9	5
6	21	7	6
7	22	9	5
7	21	6	6
7	11	10	5
NULL	23	**NULL**	4

Relational Algebra

Full Outer Join ($⟗_C$)

- Notation: $R_1 ⟗_C R_2$
- 2 relations are combined with respect to the attributes that have the *same name* and display just one of the redundant attributes + the attributes from R_1 that have no correspondent in R_2 (*null* value) + the attributes from R_2 that have no correspondent in R_1 (*null* value)
- Result set:
  - schema: a relation with the attributes from R_1 followed by the attributes from R_2
  - tuples: tuples from the condition join $R_1 \otimes_C R_2$ + tuples from R_1 that are not used in $R_1 \otimes_C R_2$ combined with the *null* value for the attributes of R_2 + tuples from R_2 that are not used in $R_1 \otimes_C R_2$ combined with the *null* value for the attributes of R_1
- equivalent with

SELECT DISTINCT *
FROM R_1 FULL OUTER JOIN R_2 ON C

Relational Algebra

Full Outer Join – example

Student

StudentId	Age
4	19
5	19
6	21
7	20
8	21

Exam

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

Course

CourseId	NoOfCredits
11	5
21	6
22	5
23	4

$\text{Student} \bowtie_{\text{Student.StudentId}=\text{Exam.StudentId}} \text{Exam} \bowtie_{Exam.CourseId=Course.CourseId} Course$

StudentId	Age	CourseId	Grade
4	19	11	9
5	19	11	10
4	19	21	10
4	19	22	9
6	21	21	7
7	20	22	9
7	20	21	6
7	20	11	10
8	21	**NULL**	**NULL**
NULL	**NULL**	23	**NULL**

Relational Algebra

Left Semi Join (⊳)

- Notation: $R_1 \rhd R_2$
- 2 relations are combined with respect to the attributes that have the *same name* and display just the attributes from R_1 that have no correspondent in R_2 (*null* value)
- Result set:
  - schema: a relation with the attributes from R_1
  - tuples: tuples from R_1 that are not used in the condition join $R_1 \otimes_C R_2$ (R_1 combined with the *null* value for the attributes of R_2)

Student

StudentId	Age
4	19
5	19
6	21
7	20
8	21

Exam

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

Student ⊳ Exam

StudentId	Age	CourseId	Grade
8	21	**NULL**	**NULL**

Relational Algebra

Right Semi Join (⊲)

- Notation: $R_1 \lhd R_2$
- 2 relations are combined with respect to the attributes that have the *same name* and display just the attributes from R_2 that have no correspondent in R_1 (*null* value)
- Result set:
  - schema: a relation with the attributes from R_2
  - tuples: tuples from R_2 that are not used in the condition join $R_1 \otimes_C R_2$ (R_2 combined with the *null* value for the attributes of R_1)

Exam

StudentId	CourseId	Grade
4	11	9
5	11	10
4	21	10
4	22	9
6	21	7
7	22	9
7	21	6
7	11	10

Course

CourseId	NoOfCredits
11	5
21	6
22	5
23	4

Exam ⊲ Course

StudentId	CourseId	Grade	NoOfCredits
NULL	23	**NULL**	4

Relational Algebra

Division (÷)

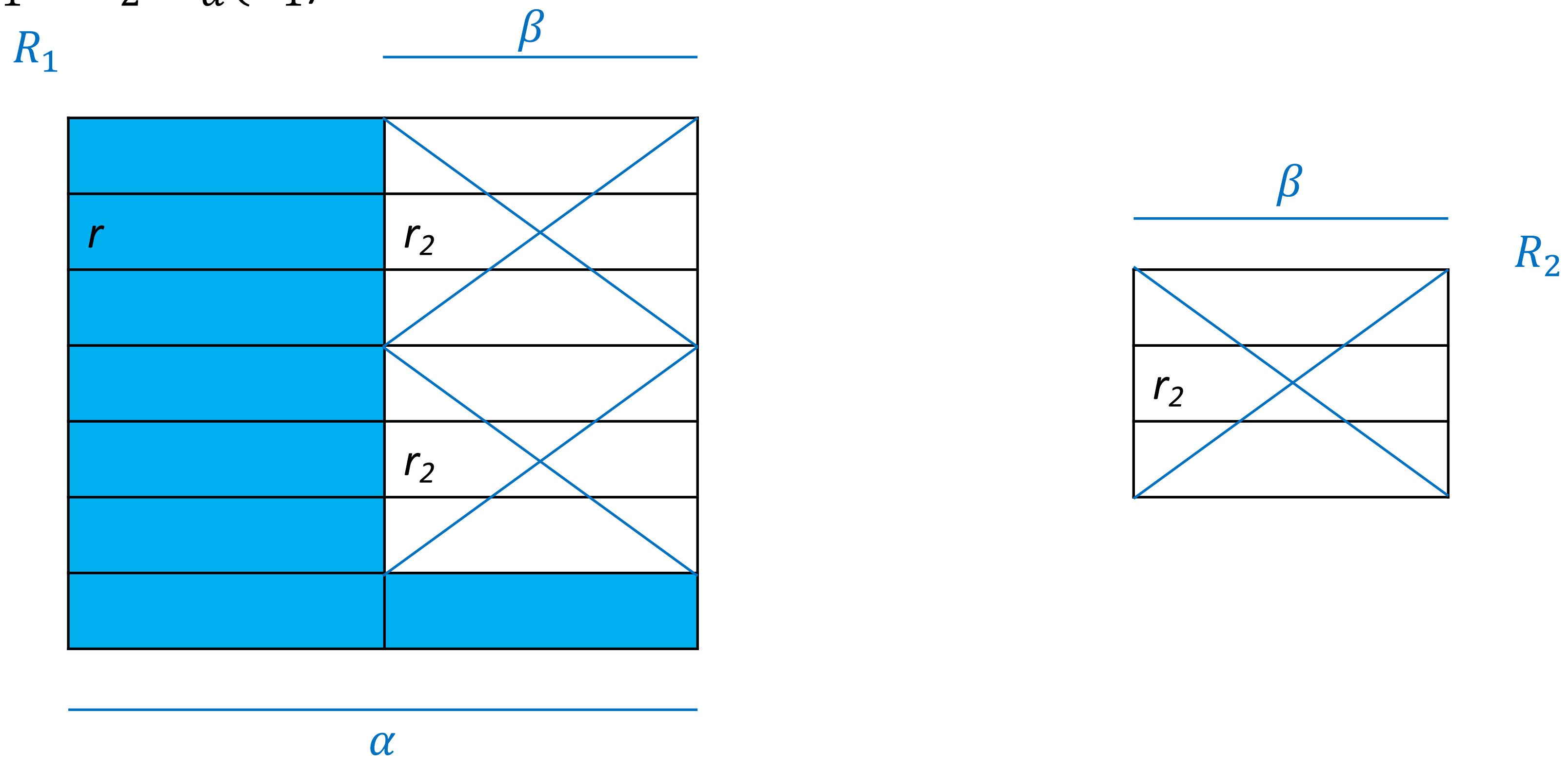
- Notation: $R_1 \div R_2$
- It is not a base operator, but it can be useful, because can simplify the query
- 2 relations are combined
- $R_1,\ R_2$ – the relations considered
- $R_1[\alpha],\ R_2[\beta], \beta \subset \alpha$, where α, β are attributes of R_1 and β is an attribute of R_2
- $R_1 \div R_2 = \{\alpha \mid \exists(\alpha, \beta) \in R_1\ \forall \beta \in R_2\}$, i.e. $R_1 \div R_2$ contains all the tuples α such that for each tuple $\beta \in R_2$, exists a $\alpha\beta$ tuple from R_1 (or, if the set of β values associated with a value α from R_1 contains all the values $\beta \in R_2$ then α will be returned as a result for $R_1 \div R_2$)
- Result set: a relation with the attributes that are in $R_1 \div R_2$ combined
  - schema: $\alpha - \beta$
  - tuples: a record $r \in R_1 \div R_2$ if $\forall r_2 \in R_2, \exists r_1 \in R_1$ 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

Relational Algebra

Division (÷)

- Generalization: α and β can represent any set of attributes; β is the set of attributes from R_1 and $\alpha \cup \beta$ represent the attributes of R_1
- Division operator is not essential; it is just a shortcut (like join, that is used more often)
- For $R_1 \div R_2$ will be determined the α values that are not connected with some β values from R_2

(the α value is not connected if attaching to it a β value from R_2 is obtained a tuple $\alpha\beta$ that is not in R_1) - not connected α values: $R_1 \div R_2$=$\pi_\alpha(R_1)$



Relational Algebra

Rename

- Notation: $\rho(R'(A_1 \to A'_1, A_2 \to A'_2), R)$ or $\rho_{R'(A'_1, A'_2)}(R)$
- The new relation R' has the same instance as R, and it's structure contains the attribute A'_i instead of the attribute A_i
- Used when the attributes and relations have the same name (e.g. *join* between 1 table, *join* for 2 tables on the same attributes); one must be renamed

Example: $\rho(Student2(StudentId \to SId, StudentName \to SName),\ Student) \Leftrightarrow$

$$SELECT\ StudentId\ as\ SId, StudentName\ as\ SName\ FROM\ Student, Student2$$

Student

StudentId	StudentName	Age
4	Maria	19
5	Dan	19
6	Florin	21
7	Lara	20

Student2

SId	SName	Age
4	Maria	19
5	Dan	19
6	Florin	21
7	Lara	20

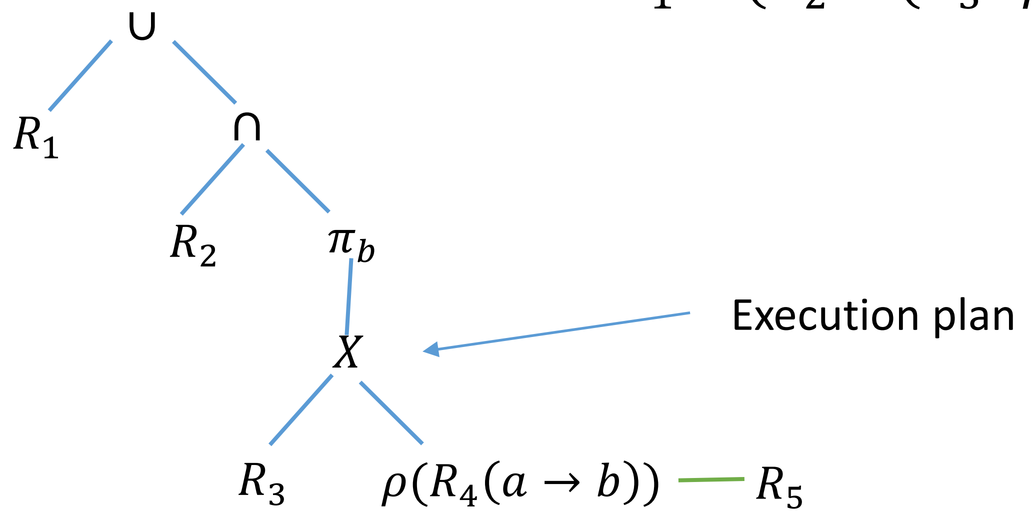
Relational Algebra

Assignment (←)

- Notation: $Temporary\_variable \leftarrow \pi_X(R_1 \times R_2)$ or R[list]:=expression
- Offers a simple way in which the complex queries can be treat it
- The result is saved in a temporary variable
- The result of the expression from the right side of ← is assigned to the attribute that is in the left side of the operator ←
- The variables can be used, after, in other expressions (e.g. result ← Temporary_variable - R_2)

Complex expressions

$$R_1 \cup (R_2 \cap (R_3 X \rho(R_4(a \rightarrow b), R_5)))$$



Relational Algebra

Example: Display the name of the students that have grades on the course 11.

○ Solution 1

$$\pi_{StudentName}\left(\left(\sigma_{CourseId=11}(Exam)\right) * Student\right)$$

○ Solution 2

$$\rho\left(Temp1, \sigma_{CourseId=11}(Exam)\right)$$
$$\rho(Temp2, Temp1 * Student)$$
$$\pi_{StudentName}(\mathrm{Temp2})$$

○ Solution 3

$$\pi_{StudentName}\left(\sigma_{CourseId=11}(Exam * Student)\right)$$

Relational Algebra

Example: Display the name of the students that have grades on the courses with 6 credits.

- the number of credits is in the table *Course*, so, a natural join will be added.

- Solution 1

$$\pi_{StudentName}\left(\left(\sigma_{NoOfCredits=6}(Course)\right) * Exam * Student\right)$$

- Solution 2

$$\pi_{StudentName}\left(\pi_{StudentId}\left(\pi_{CourseId}\left(\sigma_{NoOfCredits=6}(Course)\right) * Exam\right) * Students\right)$$

- Solution 2 is more efficient than Solution 1
- The query optimization model allows the transformation of the Solution 1 in Solution 2

Relational Algebra

Example: Display the name of the students that have grades on the courses with 5 or 6 credits.

- Are identified the courses with 5 or 6 credits and then are displayed the students that have grades on these courses.
- Solution

$$\rho\ (\ TempCourse, \left(\sigma_{NoOfCredits=5\ \vee\ NoOfCredits=6}(Course)\right)\ )$$
$$\pi_{StudentName}(TempCourse * Exam * Student)$$

- *TempCourse* can be defined also as a union.

Example: Display the name of the students that have grades on the courses with 5 AND 6 credits.

- First are identified the students that have grades in a 5 credit course; then are identified the students that have grades in a 6 credits course; then are intersected (StudentId is used)
- Solution

$$\rho\ (\ Temp5, \left(\sigma_{NoOfCredits=5}\ (Course) * Exam\right)\ )$$
$$\rho\ (\ Temp6, \left(\sigma_{NoOfCredits=6}\ (Course) * Exam\right)\ )$$
$$\pi_{StudentName}((Temp5 \cap Temp6) * Student)$$

Relational Algebra

Example: Display the name of the students that have grades on all the courses

- The *division* is used; must be prepared the relation structures before use the division operator
- Solution

$$\rho\left(TempStudentId, \pi_{StudentId,\, CourseId}(Exam)\right) \div \pi_{CourseId}(Course)$$

$$\pi_{StudentName}(TempStudentId * Student)$$

Other notations and operations: let R be a relation

- **Eliminating duplicates from a relation: $\boldsymbol{\delta(R)}$**
- **Sorting records in a relation: $\boldsymbol{S_{\{list\_attributes\}}\ (R)}$**
- **Grouping: $\boldsymbol{\gamma_{\{list1\_attributes\}\ GROUP\ BY\ \{list2\_attributes\}}\ (R)}$**
  - *list1_attributes* (can contain aggregate functions) is evaluated for each group of records
  - the records from *R* are grouped by the columns in *list2_attributes*

Relational Algebra

Example: Consider the following relations
Group [<u>GroupId</u>, Year, Specialization]
Student [<u>StudentId</u>, StudentName, Age, SGroup]
Professor [<u>ProfessorId</u>, ProfessorName, Dob]
Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

- Display the name of the students from the group *821*.

$$R := \pi_{StudentName}\left(\sigma_{SGroup=821}(Student)\right)$$

equivalent with

$$SELECT\ StudentName\ FROM\ Student\ WHERE\ SGroup = 821$$

Relational Algebra

Example: Consider the following relations

Group [<u>GroupId</u>, Year, Specialization]

Student [<u>StudentId</u>, StudentName, Age, SGroup]

Professor [<u>ProfessorId</u>, ProfessorName, Dob]

Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

- Display for each student group from *Computer Science,* the students alphabetically.

$$A := \pi_{GroupId}\left(\sigma_{Specialization='Computer\ Science'}(Group)\right)$$

$$R := S_{SGroup,\,StudentName}\left(\sigma_{SGroup\ IN\ A}(Student)\right)$$

equivalent with

$$SELECT\ *\ FROM\ Student\ WHERE\ SGroup\ IN$$
$$(SELECT\ GroupId\ FROM\ Group$$
$$WHERE\ Specialization = 'Computer\ Science')$$
$$ORDER\ BY\ SGroup, StundentName$$

Relational Algebra

Example: Consider the following relations

Group [<u>GroupId</u>, Year, Specialization]

Student [<u>StudentId</u>, StudentName, Age, SGroup]

Professor [<u>ProfessorId</u>, ProfessorName, Dob]

Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

○ Display the number of students from every group for the specialization *Computer Science*.

$$A := \sigma_{SGroup\ IN\ \left(\pi_{GroupId}\left(\sigma_{Specialization='Computer\ Science'}(Group)\right)\right)}(Student)$$

$$R := \gamma_{\{SGroup,\ COUNT(*)\}\ GROUP\ BY\ \{SGroup\}}(A)$$

equivalent with

$$SELECT\ SGroup, COUNT(*)\ FROM$$
$$(\ SELECT\ \ *\ FROM\ Student\ WHERE\ SGroup\ IN$$
$$(SELECT\ GroupId\ FROM\ Group$$
$$WHERE\ Specialization =' \ Computer\ Science')\ )\ B$$
$$GROUP\ BY\ SGroup$$

Relational Algebra

Example: Consider the following relations

Group [<u>GroupId</u>, Year, Specialization]

Student [<u>StudentId</u>, StudentName, Age, SGroup]

Professor [<u>ProfessorId</u>, ProfessorName, Dob]

Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

- Display the program for the student *Popescu*.

$$R := \sigma_{SGroup\ IN\ \left(\pi_{GroupId}\left(\sigma_{StudentName='Popescu'}(Student)\right)\right)}(Program)$$

equivalent with

$$SELECT\ *\ FROM\ Program\ WHERE\ SGroup\ IN$$
$$(SELECT\ GroupId\ FROM\ Student$$
$$WHERE\ StudentName =' Popescu')$$

Relational Algebra

Example: Consider the following relations

Group [<u>GroupId</u>, Year, Specialization]

Student [<u>StudentId</u>, StudentName, Age, SGroup]

Professor [<u>ProfessorId</u>, ProfessorName, Dob]

Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

- Display the number of hours per week for each group.

$$A(Number, SGroup) := \pi_{EndHour-StartHour,\, SGroup}(Program)$$

$$R(SGroup, NoOfHours) \coloneqq \gamma_{\{SGroup,\, SUM(Number)\}\ GROUP\ BY\ \{SGroup\}}(A)$$

equivalent with

$$SELECT\ SGroup, \quad SUM(EndHour - StartHour)$$
$$FROM\ Program$$
$$GROUP\ BY\ SGroup$$

Relational Algebra

Example: Consider the following relations

Group [<u>GroupId</u>, Year, Specialization]

Student [<u>StudentId</u>, StudentName, Age, SGroup]

Professor [<u>ProfessorId</u>, ProfessorName, Dob]

Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

- Display the name of the professors that have been teach the student with the name *Popescu*.

$$A := (\sigma_{StudentName='Popescu'}(Student)) \otimes_{Student.SGroup=Program.SGroup} Program$$

$$B := \pi_{ProfessorId}(A)$$

$$C := Professor \otimes_{Professor.ProfessorId=B.ProfessorId} B$$

$$R := \pi_{ProfessorName}\ (A)$$

equivalent with $SELECT\ Student.StudentName$

$$FROM\ Student\ INNER\ JOIN\ Program\ ON\ Student.SGroup = Program.SGroup$$

$$INNER\ JOIN\ Professor\ ON\ Professor.ProfessorId = Program.ProfessorId$$

$$WHERE\ Student.StudentName =' \ Popescu'$$

Relational Algebra

Example: Consider the following relations

Group [<u>GroupId</u>, Year, Specialization]

Student [<u>StudentId</u>, StudentName, Age, SGroup]

Professor [<u>ProfessorId</u>, ProfessorName, Dob]

Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

- Display the name of the professors that have not teach (with no program).

$$R \coloneqq \pi_{ProfesssorName}(Professor)\ -$$
$$-\pi_{ProfesssorName}(\,Professor \otimes_{Professor.ProfessorId=Program.ProfessorId} \text{Program})$$

equivalent with

$$SELECT\ P1.ProfessorName\ FROM\ Professor\ P1\ EXCEPT$$
$$SELECT\ P2.ProfessorName$$
$$FROM\ Professor\ P2\ INNER\ JOIN\ Program\ P\ ON\ P2.Professorid = P.ProfessorId$$

- Is it ok if there are professors with the same name? (ProfessorId, ProfessorName)

Relational Algebra

Example: Consider the following relations

Group [<u>GroupId</u>, Year, Specialization]

Student [<u>StudentId</u>, StudentName, Age, SGroup]

Professor [<u>ProfessorId</u>, ProfessorName, Dob]

Program [<u>Class, Day, StartHour, EndHour</u>, ProgramType, ProfessorId, SGroup]

- Display the students that have program in every day of the week (all the days in which are programs).

$$A := \delta\left(\pi_{Day}(Program)\right)$$

$$B := Student \otimes_{Student.SGroup=Program.SGroup} Program$$

$$C := \delta\left(\pi_{StudentName,Day}(B)\right)$$

- Is it ok if there are students with the same name? (StudentId, StudentName, Day)

References:

- C.J. Date, *An Introduction to Database Systems (8th Edition)*, Addison-Wesley, 2003.
- H. Garcia-Molina, J. Ullman, J. Widom, *Database Systems: The Complete Book, Prentice Hall Press*, 2008.
- G. Hansen, J. Hansen, *Database Management And Design (2nd Edition),* Prentice Hall, 1996.
- R. Ramakrishnan, J. Gehrke, *Database Management Systems*, McGraw- Hill, 2007. http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- R. Ramakrishnan, J. Gehrke, *Database Management Systems (2nd Edition),* McGraw-Hill, 2000.
- A. Silberschatz, H. Korth, S. Sudarshan, *Database System Concepts*, McGraw-Hill, 2010. http://codex.cs.yale.edu/avi/db-book/
- L. Țâmbulea, *Curs Baze de date*, Facultatea de Matematică și Informatică, UBB, 2013-2014.
- J. Ullman, J. Widom, *A First Course in Database Systems*, http://infolab.stanford.edu/~ullman/fcdb.html