Relational algebra

Relational Algebra

- Procedural language
- Six basic operators
 - select: σ
 - project: ∏
 - union: ∪
 - set difference: –
 - Cartesian product: x
 - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

Select Operation – Example

Relation r

A	В	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$$\sigma_{A=B^{\wedge}D>5}(r)$$

A	В	C	D
α	α	1	7
β	β	23	10

Select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as:

$$\sigma_p(\mathbf{r}) = \{t \mid t \in r \text{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by : \land (**and**), \lor (**or**), \neg (**not**)

Each **term** is one of:

<a href="<a href="<a

Example of selection:

$$\sigma_{dept_name="Physics"}(instructor)$$

Project Operation – Example

Relation *r*:

Project Operation

Notation:

$$\prod_{A_1,A_2,\ldots,A_k}(r)$$

where A_1 , A_2 are attribute names and r is a relation name.

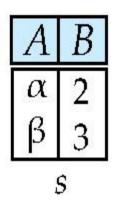
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the dept_name attribute of instructor

$$\prod_{ID, name, salary}$$
 (instructor)

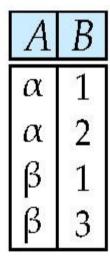
Union Operation – Example

Relations r, s:

A	В
α	1
α	2
β	1
[5] 1º	<u> </u>



 $r \cup s$:



Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.
 - 1. *r*, *s* must have the *same* **arity** (same number of attributes)
 - 2. The attribute domains must be **compatible** (example: 2^{nd} column of r deals with the same type of values as does the 2^{nd} column of s)
- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{course_id}$$
 ($\sigma_{semester="Fall"\ \Lambda\ year=2009}$ (section)) \cup $\Pi_{course\ id}$ ($\sigma_{semester="Spring"\ \Lambda\ year=2010}$ (section))

Set difference of two relations

Relations *r*, *s*:

\boldsymbol{A}	В
α	1
α	2
β	1

2
3

r - s:

Set Difference Operation

- Notation r s
- Defined as:

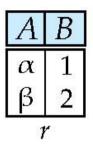
$$r-s = \{t \mid t \in r \text{ and } t \notin s\}$$

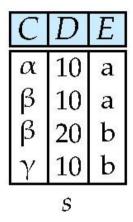
- Set differences must be taken between compatible relations.
 - r and s must have the same arity
 - attribute domains of *r* and *s* must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\prod_{course_id} (\sigma_{semester="Fall" \land year=2009}(section)) - \prod_{course_id} (\sigma_{semester="Spring" \land year=2010}(section))$$

Cartesian-Product Operation – Example

Relations *r*, *s*:





rxs:

A	В	C	D	Ε
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-Product Operation

- Notation *r* x s
- Defined as:

$$r \times s = \{t \mid q \mid t \in r \text{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of r(R) and s(S) are not disjoint, then renaming must be used.

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

 rxs

A	В	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

\boldsymbol{A}	В	C	D	E
α	1	α	10	a
β	2	β	10	a
ß	2	β	20	b

•
$$\sigma_{A=C}(r \times s)$$

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_x(E)$$

returns the expression *E* under the name *X*

If a relational-algebra expression E has arity n, then

$$\rho_{x(A_1,A_2,...,A_n)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, ..., A_n$.

Example Query

- Find the largest salary in the university
 - Step 1: find instructor salaries that are less than some other instructor salary (i.e. not maximum)
 - using a copy of instructor under a new name d

```
\Pi_{instructor.salary} (\sigma_{instructor.salary < d,salary} (instructor \times \rho_d (instructor)))
```

Step 2: Find the largest salary

```
\Pi_{salary} (instructor) –
\Pi_{instructor.salary} (\sigma_{instructor.salary} < \sigma_{d,salary} (instructor x \rho_d (instructor)))
```

Example Queries

 Find the names of all instructors in the Physics department, along with the course_id of all courses they have taught

```
Query 1 \Pi_{instructor.ID,course\_id}(\sigma_{dept\_name="Physics"}(\sigma_{instructor.ID=teaches.ID}(instructor x teaches)))
Query 2 \Pi_{instructor.ID,course\_id}(\sigma_{instructor.ID=teaches.ID}(\sigma_{instructor.ID=teaches.ID}(\sigma_{dept\_name="Physics"}(instructor) x teaches))
```

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
 - A relation in the database
 - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
 - $-E_1 \cup E_2$
 - $-E_1-E_2$
 - $-E_1 \times E_2$
 - $-\sigma_{p}(E_{1})$, P is a predicate on attributes in E_{1}
 - $-\prod_{s}(E_{1})$, S is a list consisting of some of the attributes in E_{1}
 - $-\rho_{x}(E_{1})$, x is the new name for the result of E_{1}

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

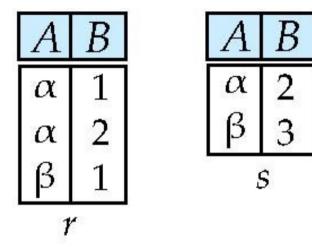
- Set intersection
- Natural join
- Assignment
- Outer join

Set-Intersection Operation

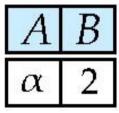
- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{t \mid t \in r \text{ and } t \in s\}$
- Assume:
 - r, s have the same arity
 - attributes of *r* and *s* are compatible
- Note: $r \cap s = r (r s)$

Set-Intersection Operation – Example

• Relation *r*, *s*:



• r ∩ s



Natural-Join Operation

- Let r and s be relations on schemas R and S respectively.
 Then, r ⋈ s is a relation on schema R ∪ S obtained as follows:
 - Consider each pair of tuples t_r from r and t_s from s.
 - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
 - t has the same value as t_r on r
 - t has the same value as t_s on s
- Example:

```
R = (A, B, C, D)

S = (E, B, D)

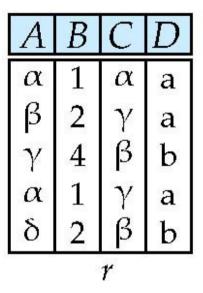
- Result schema = (A, B, C, D, E)

- r \bowtie s is defined as:

\prod_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \land r.D = s.D} (r \times s))
```

Natural Join Example

Relations r, s:



a	α
a	β
a	γ
b	δ
b	3
	a a b

 $r \bowtie s$

A	В	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Natural Join and Theta Join

- Find the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach
 - $-\prod_{\textit{name, title}} (\sigma_{\textit{dept_name="Comp. Sci."}} (\textit{instructor} \bowtie \textit{teaches} \bowtie \textit{course}))$
- Natural join is associative
 - (instructor ⋈ teaches) ⋈ course is equivalent to instructor ⋈ (teaches ⋈ course)
- Natural join is commutative
 - instruct ⋈ teaches is equivalent to teaches ⋈ instructor
- The **theta join** operation $r \bowtie_{\theta} s$ is defined as
 - $-r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$

Assignment Operation

 The assignment operation (←) provides a convenient way to express complex queries.

Write query as a sequential program consisting of

- a series of assignments
- followed by an expression whose value is displayed as a result of the query.

Assignment must always be made to a temporary relation variable.

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form 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.
 - We shall study precise meaning of comparisons with nulls later

Outer Join – Example

Relation instructor1

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

• Relation teaches1

ID	course_id
10101 12121	CS-101 FIN-201
76766	BIO-101

Outer Join – Example

Join

instructor ⋈ *teaches*

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201

Left Outer Join

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	<i>null</i>

Outer Join – Example

Right Outer Join

instructor \times teaches

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	null	null	BIO-101

Full Outer Join

instructor \to teaches

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	<i>null</i>
76766	null	null	BIO-101

Outer Join using Joins

- Outer join can be expressed using basic operations
 - e.g. r ⇒ s can be written as

```
(r \bowtie s) \cup (r - \prod_{R}(r \bowtie s) \times \{(null, ..., null)\}
```

Null Values

- It is possible for tuples to have a null value, denoted by null, for some of their attributes
- null signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null*.
- Aggregate functions simply ignore null values (as in SQL)
- For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

Null Values

- Comparisons with null values return the special truth value: unknown
 - If *false* was used instead of *unknown*, then A < 5 would not be equivalent to A >= 5
- Three-valued logic using the truth value unknown:

```
- OR: (unknown or true) = true,
(unknown or false) = unknown
(unknown or unknown) = unknown
```

- AND: (true and unknown) = unknown, (false and unknown) = false, (unknown and unknown) = unknown
- NOT: (not unknown) = unknown
- In SQL "P is unknown" evaluates to true if predicate P evaluates to unknown
- Result of select predicate is treated as false if it evaluates to unknown

Division Operator

Given relations r(R) and s(S), such that S ⊂ R, r ÷ s is the largest relation t(R-S) such that
 t x s ⊆ r

- E.g. let $r(ID, course_id) = \prod_{ID, course_id} (takes)$ and $s(course_id) = \prod_{course_id} (\sigma_{dept_name="Biology"}(course)$ then $r \div s$ gives us students who have taken all courses in the Biology department
- Can write $r \div s$ as

```
temp1 \leftarrow \prod_{R-S} (r)
temp2 \leftarrow \prod_{R-S} ((temp1 \times s) - \prod_{R-S,S} (r))
temp2 \leftarrow temp1 - temp2
```

The result to the right of the \leftarrow is assigned to the relation variable on the left of the \leftarrow .

May use variable in subsequent expressions.

Extended Relational-Algebra-Operations

- Generalized Projection
- Aggregate Functions

Generalized Projection

 Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\prod_{F_1},_{F_2},\ldots,_{F_n}(E)$$

- E is any relational-algebra expression
- Each of F_1 , F_2 , ..., F_n are are arithmetic expressions involving constants and attributes in the schema of E.
- Given relation instructor(ID, name, dept_name, salary)
 where salary is annual salary, get the same information but
 with monthly salary

 $\prod_{\text{ID, name, dept_name, salary/12}}$ (instructor)

Aggregate Functions and Operations

 Aggregation function takes a collection of values and returns a single value as a result.

avg: average value

min: minimum value

max: maximum value

sum: sum of values

count: number of values

Aggregate operation in relational algebra

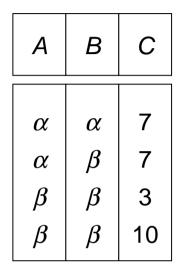
$$_{G_1,G_2,...,G_n}$$
 $_{G_1,G_1,...,F_n(A_n)}(E)$

E is any relational-algebra expression

- $-G_1, G_2 \dots, G_n$ is a list of attributes on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name
- Note: Some books/articles use γ instead of $\mathcal G$ (Calligraphic G)

Aggregate Operation – Example

Relation r:



$$\mathcal{G}_{\mathrm{sum(c)}}(\mathbf{r})$$

sum(c)

27

Aggregate Operation – Example

• Find the average salary in each department $_{dept_name}G$ $_{avg(salary)}$ (instructor)

ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	avg_salary
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregate Functions (Cont.)

- Result of aggregation does not have a name
 - Can use rename operation to give it a name
 - For convenience, we permit renaming as part of aggregate operation

 $dept_name G avg(salary) as avg_sal (instructor)$

Modification of the Database

- The content of the database may be modified using the following operations:
 - Deletion
 - Insertion
 - Updating
- All these operations can be expressed using the assignment operator

Multiset Relational Algebra

- Pure relational algebra removes all duplicates
 - e.g. after projection
- Multiset relational algebra retains duplicates, to match SQL semantics
 - SQL duplicate retention was initially for efficiency, but is now a feature
- Multiset relational algebra defined as follows
 - selection: has as many duplicates of a tuple as in the input, if the tuple satisfies the selection
 - projection: one tuple per input tuple, even if it is a duplicate
 - cross product: If there are m copies of t1 in r, and n copies of t2 in s, there are m x n copies of t1.t2 in r x s
 - Other operators similarly defined
 - E.g. union: m + n copies, intersection: min(m, n) copies difference: min(0, m n) copies

SQL and Relational Algebra

select A1, A2, ... An from r1, r2, ..., rm where P

is equivalent to the following expression in multiset relational algebra

$$\prod_{A_1,...A_n} (\sigma_P (r1 \times r2 \times ... \times rm))$$

select A1, A2, sum(A3) from r1, r2, ..., rm
 where P group by A1, A2

is equivalent to the following expression in multiset relational algebra

$$_{A1, A2}G_{sum(A3)}$$
 ($\sigma_P(r1 \times r2 \times ... \times rm)$))

SQL and Relational Algebra

 More generally, the non-aggregated attributes in the select clause may be a subset of the group by attributes, in which case the equivalence is as follows:

```
select A1, sum(A3)
from r1, r2, ..., rm
where P
group by A1, A2
```

is equivalent to the following expression in multiset relational algebra

```
\prod_{A1,sumA3} (A_{1,A2} \quad sum(A G_{as sumA3} (\sigma_P (r1 \times r2 \times ... \times rm)))
```