Relational Model II

Additional Operations

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

- Set intersection
- Natural inner join
- Assignment
- Outer join

Set-Intersection Operation

- Defined as:

  $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$

- Assumes r and s are compatible

- Can be reduced to basic relational algebra as follows:
  - $r \cap s = r - (r - s)$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	2

$r \cap s$

Natural Join

- Notation: r ⋈ s
- Let *r* and *s* be relations on schemas *R* and *S* respectively.
  Then, r ⋈ s is a relation on schema $R \cup S$ 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 attributes from *R*
    - *t* has the same value as t_s on attributes from *S*

Natural Join Example

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

A	B	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

r ⋈ s

Natural Join of instructor and teaches

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

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

A Query with Natural 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
  - $\Pi_{name,\ title}(\sigma_{dept\_name=\text{“Comp. Sci.”}}(instructor \bowtie teaches \bowtie course))$

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

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

name	*title*
Brandt	Game Design
Brandt	Image Processing
Katz	Image Processing
Katz	Intro. to Computer Science
Srinivasan	Intro. to Computer Science
Srinivasan	Robotics
Srinivasan	Database System Concepts

Natural Join Expressed in Basic Relational Algebra

- If we have two relations r(R) and s(S) with

  $R = (A, B)$ and

  $S = (B, C)$

  then the natural join $r \bowtie s$ is equal to:

$$\rho_{(A,B,C)}(\Pi_{A,\ r.B,\ C}\ (\sigma_{r.B\ =\ s.B}\ (r\ \times\ s))) .$$

Natural Join Properties

- Natural join is associative:
  - $(instructor \bowtie teaches) \bowtie course = instructor \bowtie (teaches \bowtie course)$
- Natural join is commutative:
  - $instruct \bowtie teaches = teaches \bowtie instructor$
  - … if we only refer to attributes by name (instead of position)

Assignment Operation

- Using the assignment operation ($\leftarrow$) we can write complex queries as a sequential program consisting of
  - a series of assignments
  - followed by an expression whose value is displayed as a result of the query.

- Example

$$\Pi_{instructor.ID,course\_id}(\sigma_{dept\_name=\text{“Physics”}}(\sigma_{instructor.ID=teaches.ID}(instructor \times teaches)))$$

expressed using assignments:

$r1 \leftarrow instructor \times teaches$

$r2 \leftarrow \sigma_{instructor.ID=teaches.ID}(r1)$

$r3 \leftarrow \sigma_{dept\_name=\text{“Physics”}}(r2)$

$\Pi_{instructor.ID,course\_id}(r3)$

Outer Join: Motivation

- Find instructors from the physics department together with the courses they teach (use natural join)
- This query "loses" information (which information?)

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

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Outer Join

- The outer join is an extension of the join operation that avoids loss of information
- It 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
- It uses *null* values (unknown or does not exist) to fill the missing attributes
- Notation
  - Left Outer Join: r ⟕ s
  - Right Outer Join: r ⟖ s
  - Full Outer Join: r ⟗ s

Left Outer Join of *instructor* and *teaches*

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

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
33456	Gold	Physics	87000	*null*	*null*	*null*	*null*
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
58583	Califieri	History	62000	*null*	*null*	*null*	*null*
76543	Singh	Finance	80000	*null*	*null*	*null*	*null*
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Outer Join Expressed in Basic Relational Algebra

- Outer join can be expressed using basic operations:
  - $r \mathbin{⟗} s = (r \bowtie s) \cup (r - \prod_R(r \bowtie s)) \times \mathit{Null}$

    where *Null* is the constant relation with attributes $S - R$ and only one tuple: *(null,…,null).*

Extended Relational Algebra

We now define yet more additional operations that DO add power to the relational algebra.

- Generalized Projection
- Aggregation

Generalized Projection

- Like the projection operation but allows arithmetic functions to be used in the projection list:

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

  - where F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of E.

- Example: monthly salary

$$\Pi_{ID,\ name,\ dept\_name,\ salary/12}\ (instructor)$$

Aggregation

- An **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

- An **aggregate operation** in relational algebra

$$_{G_1,G_2,\ldots,G_m}\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(E)$$

  - G_1, G_2 …, G_m 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

Aggregate Operation – Example

r:

A	B	C
a	a	7
a	b	7
b	b	3
b	b	10

- By default, the name of the result of an aggregation is just the aggregation operation
- We often use a rename operation as part of the expression to give it a better name:

$\mathcal{G}_{\mathbf{sum(C)}}(r)$:

sum(C)
27

$\mathcal{G}_{\textbf{\textit{sum}}(C)\ \textbf{\textit{as}}\ c\_sum}(r)$:

c_sum
27

Aggregate Operation – Example

- Find the average salary in each department

$$_{dept\_name}\mathcal{G}_{\textbf{avg}(salary)\ as\ avg\_salary}\ (instructor)$$

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

Null Values

- The special value ***null*** is a member of every domain. It either signifies a value
  1. that we don't know, or
  2. that does not exist
- We will see that null values are problematic, so we avoid them if possible

- How databases treat null values:
  - The result of any arithmetic expression involving *null* is *null.*
  - Aggregate functions simply ignore *null* values
  - How about comparisons?
    - What is the result of "salary > 50000" if salary is *null?*

Null Values

- Comparisons with null values return the special truth value: *unknown*
- 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*
- The result of the select predicate is treated as *false* if it evaluates to *unknown*

TRANSLATING SQL INTO RELATIONAL ALGEBRA

Multiset Relational Algebra

- Relational algebra works on sets, which have no duplicates
  - e.g. after projection they are removed
- SQL works on **multisets**, which do have duplicates
- Example: {1,2,2} and {1,2} are two different multisets. There are two duplicates of 2 in the first multiset
- We want to translate SQL into relational algebra, so we first need to define a version of relational algebra on multisets.
- Why do you think SQL made the design decision to keep duplicates?

Multiset Relational Algebra

- Multiset relational algebra is defined just like usual relational algebra with duplicates treated as follows:
  - Selection: output has as many duplicates as the input
  - Projection: one tuple per input tuple, even if it is a duplicate
  - Cartesian product: If there are m copies of t_1 in r, and n copies of t_2 in s, there are $m \cdot n$ duplicates of t_1 t_2 in r x s
  - Set union: output has $m + n$ copies
  - Set difference: $\max(0, m - n)$ copies
    - $\{a, a, b\} - \{a, b, b\} = \{a\}$
  - Aggregation: treats duplicates as separate elements
    - $\mathbf{avg}(\{1,1,2,3\}) = 1.75$

SQL

- Example: Find the building of the instructor “Einstein”

  select *building*
  from *instructor*, *department*
  where *instructor.dept_name* = *department.dept_name* **and**
  name = ‘Einstein’

- More generally, an SQL query has the form:

  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  where P

  - A_i is an attribute
  - r_i is a relation
  - P is a predicate.

Translating SQL to Relational Algebra

- This query:

  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  where P

  translates to:

$$\Pi_{A_1,A_2,\ldots,A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

Translating SQL to Relational Algebra (with Aggregation)

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

translates to

$$_{A1, A2}\mathcal{G}_{\textbf{sum}(A3)}\,(\sigma_{P}\,(r_1 \times r2 \times .. \times rm)))$$