Chapter 2: The Relational Model

Outline

Structure of Relational Databases

Database Schema

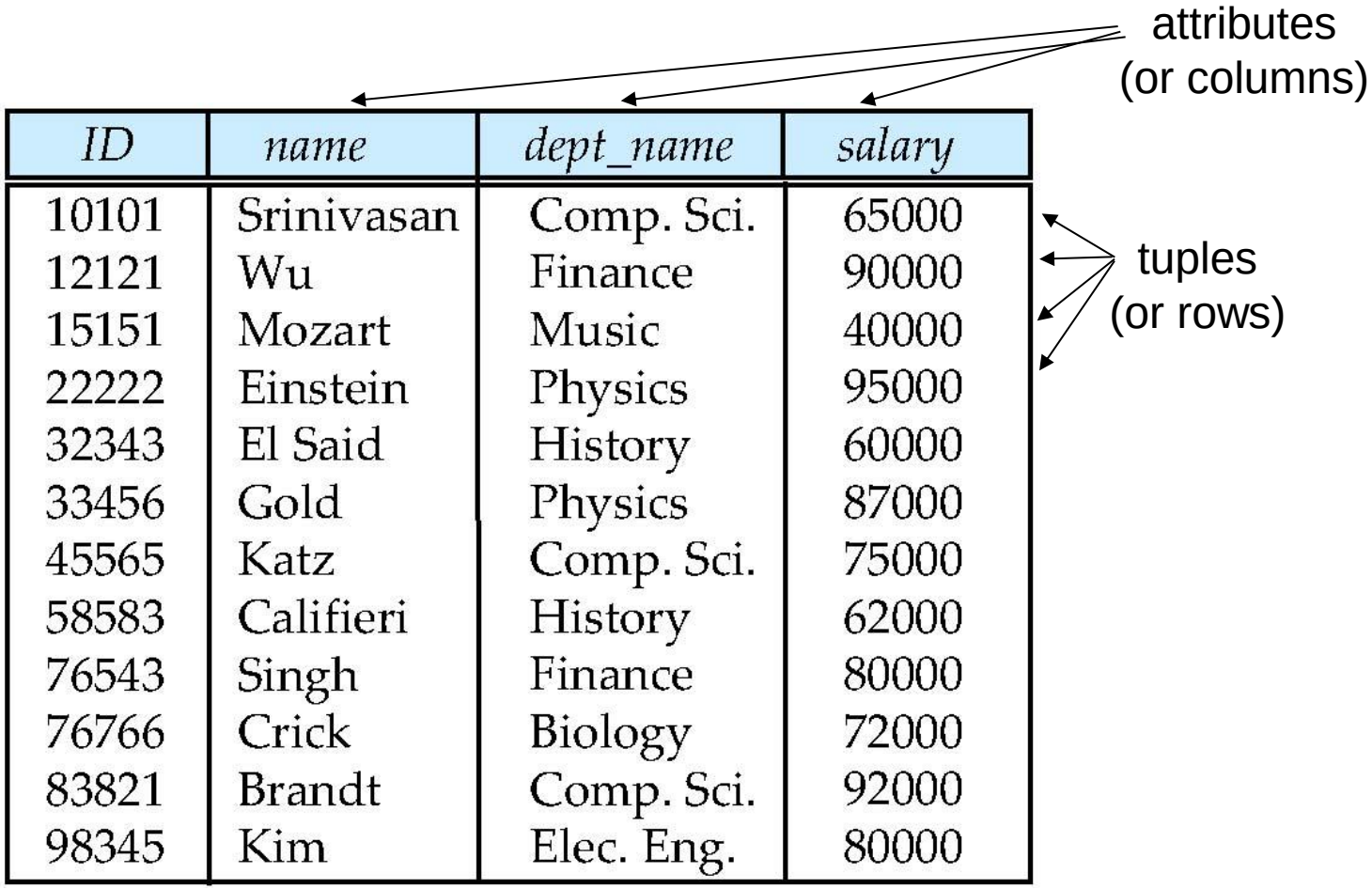
Keys

Schema Diagrams

Relational Query Languages

The Relational Algebra

Example of a Relation



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

Basic Structure

Formally, given sets D_1, D_2, …. D_n a **relation** r is a subset of D_1 x D_2 x … x D_n.
Thus, a relation is a set of n-tuples (a_1, a_2, …, a_n) where each $a_i \in D_i$

Example: If

name = {Wu, Mozart, Gold, Singh, …} /* set of all instructor names */

dept_name = {Music, Physics, Finance, …} /* set of all department names*/

salary = {40000, 80000, 87000,90000 …} /* set of all salary */

Then r = { (Wu, Finance, 90000),
(Mozart, Music, 40000),
(Gold, Physics, 87000),
(Singh, Finance, 80000) }

is a relation over

name x dept_name x salary

Relation Schema and Instance

$A_1, A_2, \ldots, A_n$ are *attributes*

$R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

Example:

instructor = (*ID*, *name*, *dept_name*, *salary*)

A relation instance *r* defined over schema *R* is denoted by *r* (*R*).

The current values a relation are specified by a table

An element ***t*** of relation ***r*** is called a *tuple* and is represented by a *row* in a table

Attributes

The set of allowed values for each **attribute** is called the **domain（域）**of the attribute

Attribute values are (normally) required to be **atomic（原子的）**; that is, indivisible

The special value **null（空值）**is a member of every domain

The null value causes complications in the definition of many operations

Relations are Unordered

Order of tuples is irrelevant (tuples may be stored in an arbitrary order)

Example: *instructor* relation with unordered tuples

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

Database Schema

Database schema -- is the logical structure of the database.

Database instance -- is a snapshot of the data in the database at a given instant in time.

Example:

schema: *instructor* (*ID, name, dept_name, salary*)

Instance:

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

Keys

Let K $\subseteq$ R

K is a **superkey(超健)** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*

Example: {*ID*} and {ID,name} are both superkeys of *instructor.*

Superkey *K* is a **candidate key(候选健)** if *K* is minimal

Example: {*ID*} is a candidate key for *Instructor*

One of the candidate keys is selected to be the **primary key（主键）**.

Keys

Foreign key（外键） constraint from attribute(s) A of relation r1 to the primary key B of relation r2 states that on any database instance, the value of A for each tuple in r1 must also be the value of B for some tuple in r2.

Referencing relation : teaching

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

Referenced relation: instructor

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

Referential integrity(参照完整性) constraint requires that the values appearing in specified attribute(s) A of any tuples in the referencing relation r1 also appear in specified attribute(s) B of at least one tuple in the referenced relation r2.

Referencing relation : section

course_id	*sec-id*	*semester*	*year*	*time-slot_id*
CS-101	1	spring	2022	t1
CS-101	2	spring	2022	t1
BIO-101	1	summer	2021	t2

Referenced relation: time-slot

time-slot_id	*day*	*start-time*	*end-time*
t1	Tue	13:15	15:40
t1	Fri	8:00	9:35
t2	Wed	13:15	15:40
t2	Wed	18:30	20:05

Database

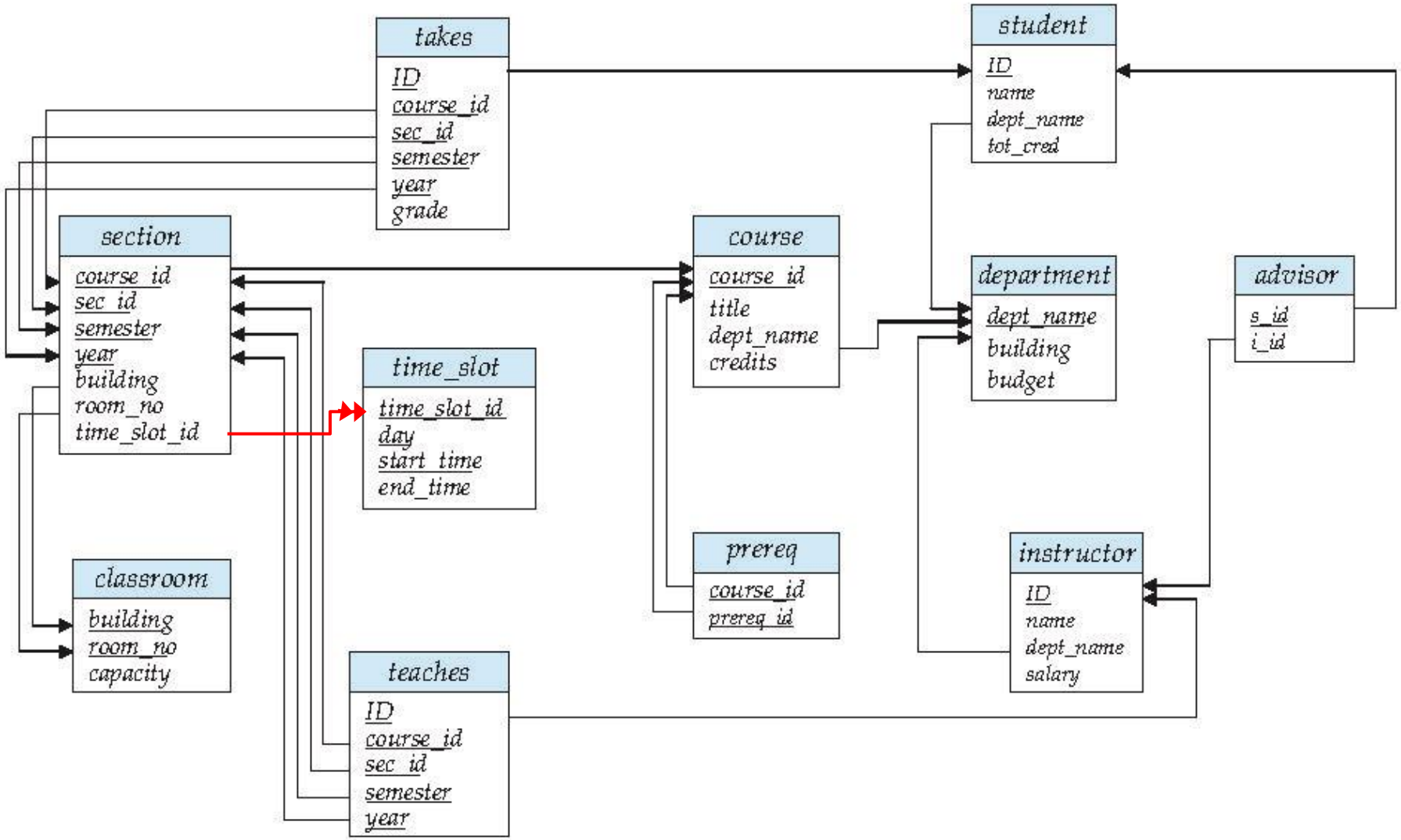
A database consists of multiple relations

Information about an enterprise is broken up into parts

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

Figure 2.9 Schema of the university database.

Schema Diagram for University Database



Relational Query Languages

Procedural vs.non-procedural, or declarative

“Pure” languages:

- Relational algebra(关系代数)
- Tuple relational calculus（元组关系演算）
- Domain relational calculus（域关系演算）

The above 3 pure languages are equivalent in computing power

We will concentrate on relational algebra

- Not Turing-machine equivalent
- Consists of 6 basic operations

Relational Algebra

A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.

Six basic operators

- select: σ
- project: Π
- union: $\cup$
- set difference: –
- Cartesian product(笛卡尔积): x
- rename: ρ

Select Operation – Example

Relation r

A	B	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	B	C	D
α	α	1	7
β	β	23	10

Select Operation

The **select** operation selects tuples that satisfy a given predicate.

Notation: $\sigma_p(r)$

p is called the **selection predicate**

Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where *p* is a formula in propositional calculus consisting of **terms** connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each **term** is one of:

<attribute> *op* <attribute>

<attribute> *op* <constant>

where *op* is one of: $=, \neq, >, \geq, <, \leq$

Example of selection:

$$\sigma_{dept\_name=\text{“}Physics\text{”}}(instructor)$$

$$\sigma_{salary > 90000}(instructor)$$

$$\sigma_{dept\_name=\text{“}Physics\text{”} \wedge salary > 90000}(instructor)$$

Project Operation – Example

Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

The **project** operation is a unary operation that returns its argument relation, with certain attributes left out.

Notation: $\Pi_{A_1, A_2, \ldots, A_k}(r)$

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*

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

Union Operation – Example

Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∪ s:

A	B
α	1
α	2
β	1
β	3

Union Operation

The **union** operation allows us to combine two relations

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** (元数)(same number of attributes)
2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2009}(section)) \cup \Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2010}(section))$$

Set difference – Example

Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r – s:

A	B
α	1
β	1

Set Difference Operation

The **set-difference** operation allows us to find tuples that are in one relation but are not in another.

Notation $r - s$

Defined as:

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

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

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2009}(section)) -$$
$$\Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2010}(section))$$

Cartesian-Product Operation – Example

Relations *r, s*:

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

A	*B*	*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

Cartesian-Product Operation

The **Cartesian-product operation** (denoted by **X**) allows us to combine information from any two relations.

Notation r x s

Defined as:

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

Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \varnothing$).

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: $\sigma_{A=C}(r \times s)$

r x s

A	*B*	*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

$\sigma_{A=C}(r \times s)$

A	*B*	*C*	*D*	*E*
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

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

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

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

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

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

Query 2

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

Example Queries

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

Query

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

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

- $\Pi_{salary} (instructor) - \Pi_{instructor.salary} (\sigma_{instructor.salary < d,salary} (instructor \times \rho_{d} (instructor)))$

Example Query

Find the largest salary in the university

$$\Pi_{salary}(instructor) - \Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_{d}(instructor)))$$

ID	*salary*
10101 12121 15151	70000 80000 90000

instrtuctor.ID	*instructor.salary*	*d.ID*	*d.salary*
10101	70000	10101	70000
10101	70000	12121	80000
10101	70000	15151	90000
12121	80000	10101	70000
12121	80000	12121	80000
12121	80000	15151	90000
15151	90000	10101	70000
15151	90000	12121	80000
15151	90000	15151	90000

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: $r \cap s$
- Natural join: $r \bowtie s$
- Semijoin: $\ltimes_\theta$
- Assignment: $\leftarrow$
- Outer join : $r \mathbin{⟕} s$, $r \mathbin{⟖} s$, $r \mathbin{⟗} s$
- Division Operator: $r \div s$

Set-Intersection Operation – Example

Relation *r, s*:

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

$r \cap s$

A	*B*
α	2

Set-Intersection Operation

The set-intersection operation allows us to find tuples that are in both the input relations.

Notation: $r \cap s$

Defined as:

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

Assume:

r, s have the *same arity*

attributes of r and s are compatible

Note: $r \cap s = r - (r - s)$

Natural Join – Example

Relations r, s:

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

r ⋈ 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

Natural-Join Operation

Notation: $r \bowtie s$

Let r and s be relations on schemas R and S respectively.
Then, $r \bowtie s$ 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 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)

$r \bowtie s =$

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

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

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

Natural join is associative

$(instructor \bowtie teaches) \bowtie course$ is equivalent to
$instructor \bowtie (teaches \bowtie course)$

Natural join is commutative

$instructor \bowtie teaches$ is equivalent to
$teaches \bowtie instructor$

The **theta join** operation $r \bowtie_\theta s$ is defined as

$$r \bowtie_\theta s = \sigma_\theta (r \times s)$$

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

instructor

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

teaches

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

Join

instructor ⋈ *teaches*

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

Left Outer Join

instructor ⟕ *teaches*

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

Outer Join – Example

instructor

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

teaches

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

Right Outer Join

instructor ⟖ *teaches*

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

Full Outer Join

instructor ⟗ *teaches*

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

Outer Join using Joins

Outer join can be expressed using basic operations

r $\overset{\supset}{\bowtie}$ s can be written as

$(r \bowtie s) \cup (r - \prod_R(r \bowtie s) \times \{(null, \ldots, null)\}$

r $\overset{}{\bowtie}\subset$ s can be written as

$(r \bowtie s) \cup \{(null, \ldots, null)\} \times (s - \prod_S(r \bowtie s)$

r $\overset{\supset}{\bowtie}\subset$ s can be written as

$(r \bowtie s) \cup$

$(r - \prod_R(r \bowtie s) \times \{(null, \ldots, null)\} \cup$

$\{(null, \ldots, null)\} \times (s - \prod_S(r \bowtie s)$

Semijoin（半连接） Operation

Notation: $r \ltimes_\theta s$

Is a subset of r, in which every tuple r_i matches at least one truple s_i in s under the condition θ.

$r \ltimes_\theta s = \Pi_R (r \bowtie_\theta s)$

Semijoin examples:

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

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

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

B	D	E
1	a	α
1	a	γ
2	b	δ

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.

Division Operation – Example

Relations *r, s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

r

B
1
2

s

r ÷ *s*:

A
α
β

Division Operator

Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

$$t \times s \subseteq r$$

E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$

then $r \div s$ gives us students who have taken all courses in the Biology department

Can write $r \div s$ as

$$temp1 \leftarrow \prod_{R\text{-}S}(r)$$
$$temp2 \leftarrow \prod_{R\text{-}S}((temp1 \times s) - \prod_{R\text{-}S,S}(r))$$
$$result = temp1 - temp2$$

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.

Another Division Example

Relations *r, s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

r ÷ *s*:

A	*B*	*C*
α	a	γ
γ	a	γ

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.

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

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

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

Aggregate Functions and Operations

Aggregation function （聚合函数）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

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

E is any relational-algebra expression

G_1, G_2 …, 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*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

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

sum(*c*)
27

Aggregate Operation – Example

Find the average salary in each department

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

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

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

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（多重集） 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*

set operators

- 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

$$\Pi_{A1, .., An} (\sigma_{P} (r1 \times r2 \times .. \times rm))$$

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

$$A1, A2\, \mathcal{G}\, sum(A3) (\sigma_{P} (r1 \times r2 \times .. \times rm)))$$

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

$$\Pi_{A1,sumA3}(\ {}_{A1,A2}\mathcal{G}\ \text{sum}(A3)\ \text{as sumA3}(\sigma_{P}\ (r1 \times r2 \times .. \times rm)))$$

End of Chapter 2