RAWDATA
Section 1

Relational Model & Relational Algebra

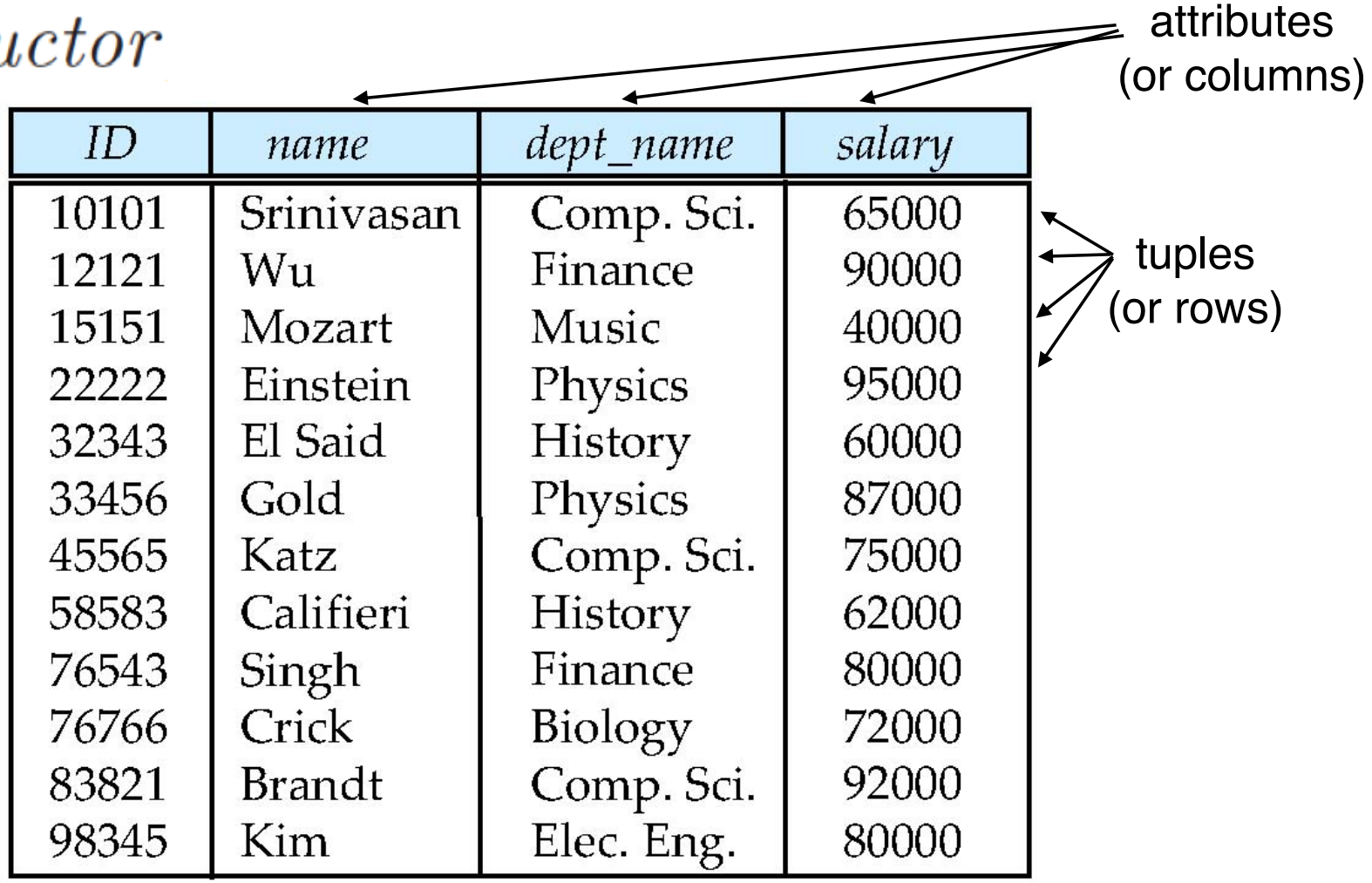
Henrik Bulskov & Troels Andreasen

Relational model

- Relational model
  - primary data model for commercial dataprocessing applications
  - successful mainly due to its simplicity
- Relational database
  - consists of a collection of relations
- Relation
  - a set of tuples
- Relation = Table
  - a relation can be considered a table,
  - tuples = rows

Example of a Relation (or table)

instructor



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

Attribute Types

- The set of allowed values for an attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, their values are indivisible
- The special value ***null*** is a member of every domain
  - ***null****:* unknown (or inapplicable) value
- The null value causes complications in the definition of many operations

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

Relation Schema and Instance

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

    Example:

    $instructor = (ID, name, dept\_name, salary)$
- **relation instance**
  - Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a subset of the product of these sets

    $r \subseteq D_1 \times D_2 \times \ldots \times D_n$

    Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$ (that is, D_i is the domain of a_i)
  - The current values of a relation ***r*** are specified by a table
  - An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

instructor

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

Relations are Unordered

- Relations are sets
  - thus the 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

- A database consists of multiple relations
- Information about an enterprise is (normally) broken up into parts, for instance

  instructor
  student
  advisor

- Bad design example:
  univ (*instructor -ID, name, dept_name, salary, student_Id*, ...)
  results in
  - repetition of information
    (e.g. when two students have the same instructor)
  - the need for null values (e.g. for a student with no advisor)
- Normalization theory (to be covered later) deals with how to design "good" relational schemas

Keys - Superkey

- Let *K* be one or more attributes from schema *R*, thus $K \subseteq R$
  - with *R* = (*ID, name, dept_name, salary*)
    K could for instance be *(ID)* or *(name)* or *(ID,name)*

- *K* is a **superkey** of *R* if *K* is **unique**, that is, values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Superkeys of *instructor*?
    - (*ID, name, dept_name, salary*) ?
    - (*ID,dept_name*) ?
    - (*ID*) ?
    - (*name*) ?
    - (dept_*name*) ?
    - (dept_*name, salary*) ?

instructor

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

Keys – Candidate and Primary

- Superkey *K* is a **candidate key** if *K* is **minimal**
  - Example: (*ID*) is a candidate key for *Instructor*
  - others?
    - (*ID, name, dept_name, salary*) ?
    - (*ID,name*) ?
    - (*name*) ?
    - (dept_*name, salary*) ?

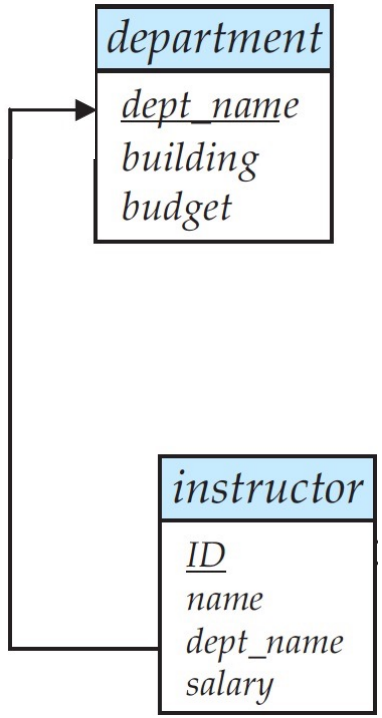
- One of the candidate keys is selected to be the **primary key**.

instructor

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

Keys – Foreign key

- **Foreign key** constraint:
  Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
- Example
  - (referencing) *instructor*, by attribute *dept_name*
  - (referenced) *department*



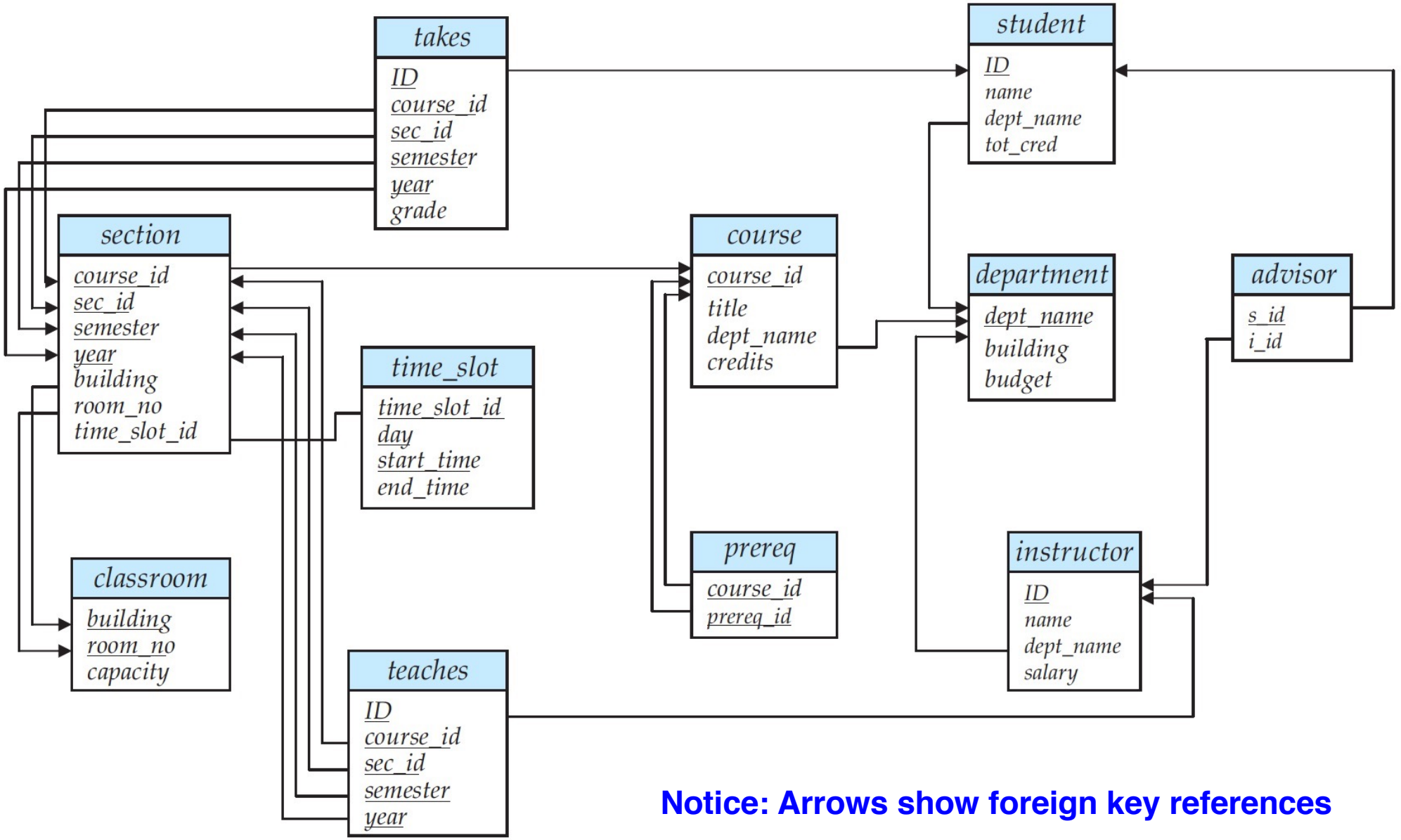
instructor

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

department

dept_name	building	budget
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

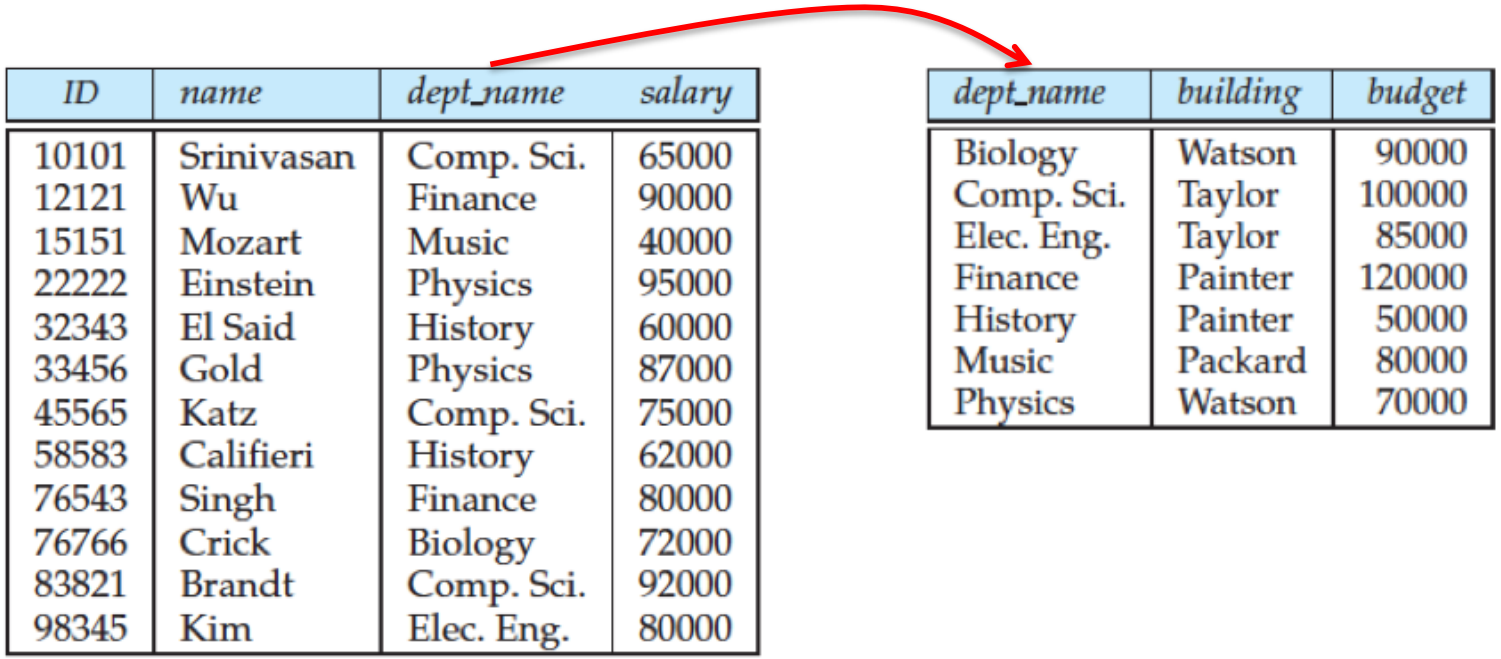
Schema Diagram for University Database



Notice: Arrows show foreign key references

2.2

- Consider the foreign key constraint from the dept_name attribute of instructor to the department relation. Give examples of inserts and deletes to these relations, which can cause a violation of the foreign key constraint.



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

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Relational Query Languages

- "Pure" languages:
  - Relational algebra
  - Tuple relational calculus / Domain relational calculus

- Procedural / operational language
  - the user instructs the system to perform a sequence of operations
  - thus tells: what and how things should be done
- Non-procedural / declarative language
  - the user only describes the desired information
  - thus tells only: what should be done

- We cover only (among the "Pure")
  - Relational Algebra which is a procedural language

- We have already considered
  - SQL which is a non-procedural language

Relational Operations

- can be applied to either
  - single relation or
  - pair of relations

- Single relation
  - Selection of tuples (rows) $\sigma_{D>5}(r)$
  - Selecting of columns $\Pi_{A,C}(r)$

- Pair of relations
  - Cartesian product (Set product) $r \times s$
  - Union, $r \cup s$
  - Difference, $r - s$
  - Intersection $r \cap s$
  - Natural Join $r \bowtie s$

Selection of tuples (rows)

- Relation *r*

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

- Select tuples with A=B and D > 5
  - $\sigma_{A=B \text{ and } D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Selection of Columns (Attributes)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- Select A and C
  - Projection
  - $\Pi_{A, C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Joining two relations – Cartesian Product

- 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

Union of two relations

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cup s$:

A	B
α	1
α	2
β	1
β	3

Set difference of two relations

- 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 Intersection of two relations

- 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

Joining two relations – Natural Join

- Combines tuples from *r* and *s* in the natural way:
  - same value on common attributes

- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, the "natural join" of relations *R* and *S* is a relation on schema $R \cup S$ (union of attributes of *R* and *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:

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

  $S = (B, D, E)$

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

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

- Natural Join
  - $r \bowtie s$

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

Relational Operations

- Operations
  - Selection of tuples (rows) $\sigma_{D>5}(r)$
  - Selecting of columns $\Pi_{A,C}(r)$
  - Cartesian product (Set product) $r \times s$
  - Union, $r \cup s$
  - Difference, $r - s$
  - Intersection $r \cap s$
  - Natural Join $r \bowtie s$

- Combine operations
  - algebra on numbers take one or more numbers as input and return a number
  - relational algebra take one or two relations as input and return a relation
  - we can thus combine operations, like in

$$\Pi_{name}\left(\sigma_{dept\_name = \text{“Physics”}}\left(instructor\right)\right)$$

Combining operations

- Example

$instructor$

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

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

ID	name	dept_name	salary
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

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

name
Einstein
Gold

Algebra vs SQL

- What are the courses offered by the Comp. Sci. department?
  - in SQL we can find the answer by the following query

```
mysql> SELECT course_id, title
    -> FROM section NATURAL JOIN course
    -> WHERE dept_name = "Comp. Sci.";
+-----------+----------------------------+
| course_id | title                      |
+-----------+----------------------------+
| CS-101    | Intro. to Computer Science |
| CS-101    | Intro. to Computer Science |
| CS-190    | Game Design                |
| CS-190    | Game Design                |
| CS-315    | Robotics                   |
| CS-319    | Image Processing           |
| CS-319    | Image Processing           |
| CS-347    | Database System Concepts   |
+-----------+----------------------------+
```

  - So in algebra this would be?
    - $\Pi_{course\_id,\ title}\ (\sigma_{dept\_name="Comp.\ Sci."}\ (section \bowtie course))$
  - and the answer?

Algebra vs SQL

- The SQL **select** clause list the attributes desired in the result of a query.
  - In relational algebra this corresponds to the **projection** operation
- The SQL **from** clause lists the relations involved in the query
  - In relational algebra this corresponds to the **cartesian product** operation
- The SQL **where** clause specifies conditions that each row in the result must satisfy
  - In relational algebra this corresponds to the **selection** operation

2.7

- 2.7 Consider the relational database of Figure 2.14. Give an expression in the relational algebra to express each of the following queries:
  - a. Find the names of all employees who live in city “Miami”.
  - b. Find the names of all employees whose salary is greater than $100,000.
  - c. Find the names of all employees who live in “Miami” and whose salary is greater than $100,000.

employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)

Figure 2.14 Relational database for Exercises 2.1, 2.7, and 2.12.

Relational Algebra, definitions

- Procedural language
- Six basic operators
  - select: σ
  - project: Π
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ

- The operators take one or two relations as inputs and produce a new relation as a result.

- Remember relations are sets
  - what does this mean?

Select Operation

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

- Notation: $\sigma_p(r)$
  - *p* is called the **selection predicate**
- Provides the set of tuples that satisfies *p,* thus defined as :

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

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

<attribute> *op* <attribute> or <constant>

where *op* is one of: **=, ≠, >, ≥, <, ≤**

- Examples of selection:

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

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

Select Operation – Example

- $instructor$ 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

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

ID	name	dept_name	salary
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

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

ID	name	dept_name	salary
22222	Einstein	Physics	95000

Project Operation

- 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*

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

- 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
β	1
β	2

Project Operation – Example

- $instructor$ 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

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

ID	name	salary
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Union Operation

- Relations *r, s:*

A	B
α	1
α	2
β	1

A	B
α	2
β	3

- 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: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

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

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

Union Operation – Example

- *section* relation:

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

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

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Set Difference Operation

- 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

- 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

- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

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

Set Difference – Example

- *section* relation:

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

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

course_id
CS-347
PHY-101

Cartesian-Product Operation

- Notation

$$r \times s$$

- Defined as:

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

- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

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 – Example

instructor

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

teaches

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

- Cartesian product: $instructor \times teaches$

inst.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Physics	95000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Physics	95000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Physics	95000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Physics	95000	10101	FIN-201	1	Spring	2010
10101	Srinivasan	Physics	95000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Physics	95000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Physics	95000	10101	CS-101	1	Fall	2009
12121	Wu	Physics	95000	10101	CS-315	1	Spring	2010
12121	Wu	Physics	95000	10101	CS-347	1	Fall	2009
12121	Wu	Physics	95000	10101	FIN-201	1	Spring	2010

Notice naming of attributes here

Cartesian-Product Operation – Example

instructor

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

teaches

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

- name and course_id for courses taught by teachers of the Physics department
  - one solution: projection of a selection of a selection of a product

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

name	course_id
Einstein	PHY-101

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:

$$\rho_{\textit{instructor\_detail}}(\textit{instructor} \bowtie \textit{department})$$

returns *instructor* $\bowtie$ *department* under the name *instructor_detail*

- 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 Query

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

- So, what would be returned from the following query?
    - $\Pi_{salary}(instructor) - \Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_{d}(instructor)))$

Example Query

instructor

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

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

salary
65000
90000
40000
60000
87000
75000
62000
72000
80000
92000

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

salary
95000

Formal Definition: Relational Algebra Expressions

- A basic relational-algebra expression consists of either one of the following:
  - A relation in the database
  - A constant relation like (*)
- 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

(*) Constant relation example: { (22222, Einstein, Physics, 95000), (76543, Singh, Finance, 80000) }

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:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible

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

- Relation r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

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

A	B
α	2

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)
  - $r \bowtie s$ is defined as:

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

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 Example

instructor

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

teaches

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

- *instructor* $\bowtie$ *teaches*

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

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 a more general join operation that combine selection and Cartesian product into a single operation, it is defined as
  - $r \bowtie_\theta s = \sigma_\theta (r \times s)$

6.2 Consider the relational database of Figure 6.22, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following queries:

b. Find the names of all employees in this database who do not work for "First Bank Corporation".

employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*person_name*, *manager_name*)

6.11

6.11 Consider the relational database of Figure 6.22, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following queries:

d. Find the names of all employees in this database who live in the same city as the company for which they work.

employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*person_name*, *manager_name*)

Assignment Operation

- The assignment operation ($\leftarrow$) 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.

- Example

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

  - Can be decomposed using assignment

    $R1 \leftarrow \sigma_{instructor.ID=teaches.ID}(instructor \times teaches)$

    $R2 \leftarrow \sigma_{dept\_name=\text{“Physics”}}(R1)$

    $Result \leftarrow \Pi_{instructor.ID,course\_id}(R2)$

Outer Join

- An extension of the join operation that avoids loss of information.
- 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.
- 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	CS-101
12121	FIN-201
76766	BIO-101

Outer Join – Example

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

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

- **Natural Join**

instructor ⋈ *teaches*

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

- **Left Outer Join**

instructor ⟕ *teaches*

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

Outer Join – Example

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

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

- **Right Outer Join**

  instructor ⟖ *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 ⟗ *teaches*

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

Null Values

- We can thus have tuples with 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 ***null***s are assumed to be the same (as in SQL)

Null Values and Three-valued logic

- Comparisons with *null* values return the special truth value: *unknown*
  - If *false* was used instead of *unknown*, then *not (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*

- Notice: Same issue as covered under SQL

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

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

  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

- $\mathcal{G}_{\text{sum(c)}}(\text{r})$

sum(*c*)
27

Aggregate Operation – Example

- Find the average salary in each department

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

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

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