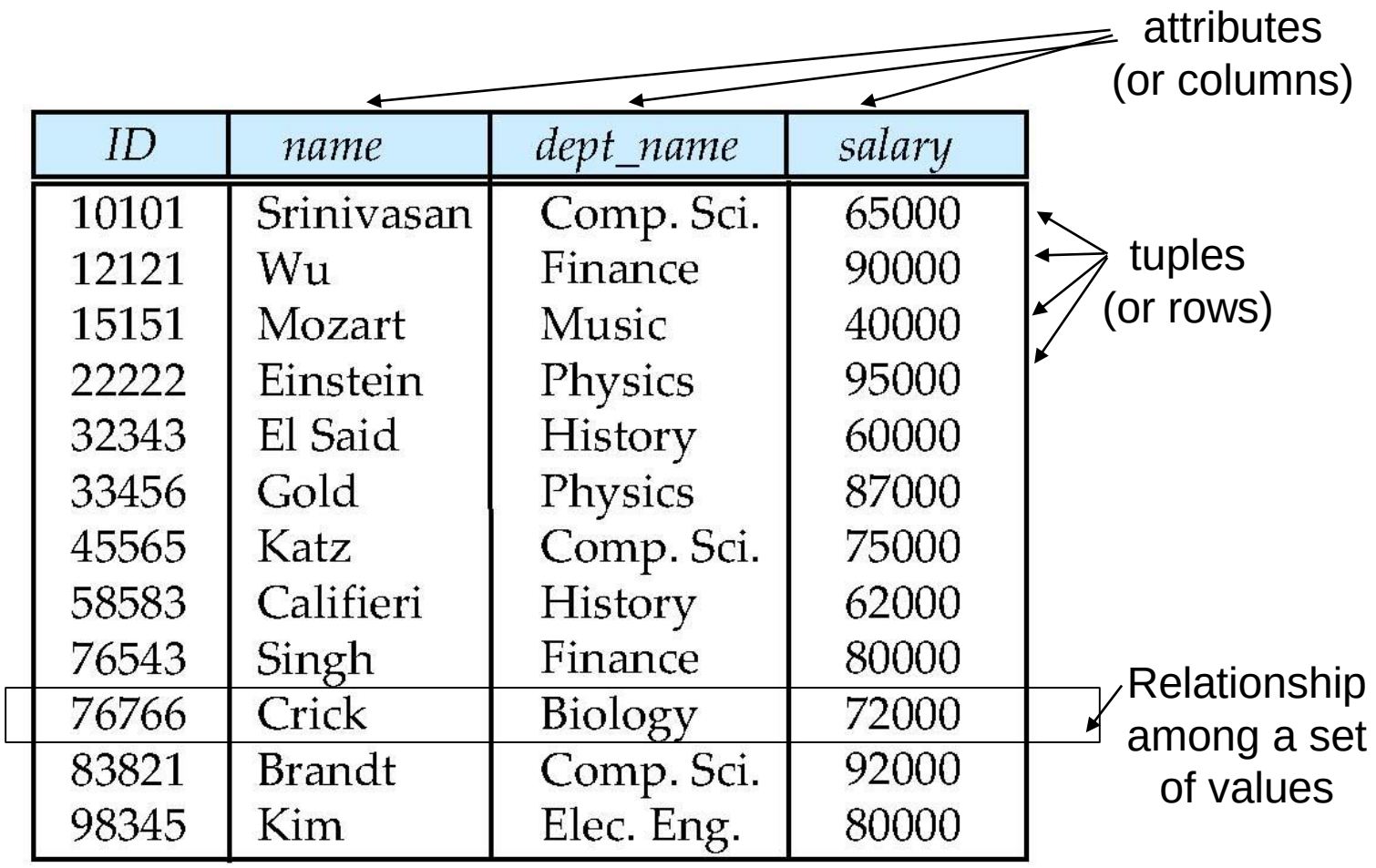
Chapter 2: Intro to Relational Model

Edited by Radhika Sukapuram. Original slides by

Database System Concepts, 6th Ed.

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

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicating that the value is “unknown” or unspecified
- The null value causes complications in the definition of many operations

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*)

- Formally, given sets $D_1, D_2, \ldots D_n$ a **relation *r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

  Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

- The current values (**relation instance**) of a relation are specified by a table
- An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

Relation schema, relation, instance

- Relation schema: Type definition
- Relation : variable
- Relation instance : value of variable

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

Keys

- Let R: set of attributes in the schema of relation *r*
- Let K $\subseteq$ R
- *K* is a **superkey** 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** if *K* is minimal.
  - No proper subset of K is a superkey
  - Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - The candidate key that is least likely to change is selected
- Keys are a property of a relation, not a tuple

Instructor and department relations

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

(a) The *instructor* table

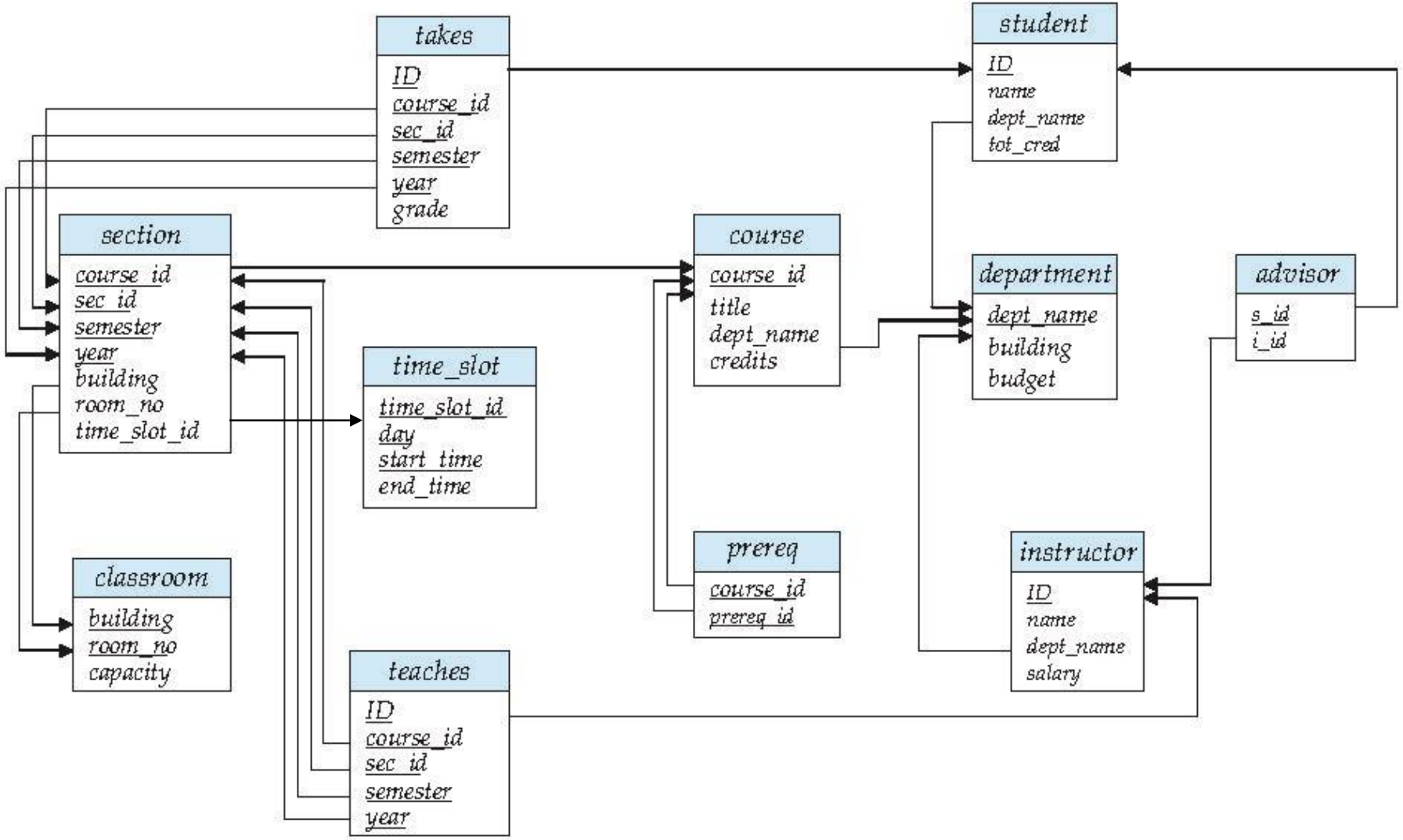
dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Keys contd.

- **Foreign key:** A relation r1 may include in its attributes the primary key k of another relation r2. k is called a foreign key from r1, referencing r2.
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation (r1)
  - **Referenced** relation (r2)
  - Example – *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

Schema Diagram for University Database



Relational Query Languages

- Imperative, functional, or declarative
- **Imperative query language:** the user instructs the system to perform a *specific sequence of operations* on the database to compute the desired result;
  - such languages usually have a notion of state variables, which are updated in the course of the computation
- **Declarative query language**: the *user describes the desired information* without giving a specific sequence of steps or function calls for obtaining that information
  - the desired information is typically described using some form of mathematical logic.
  - It is the job of the database system to figure out how to obtain the desired information.
- **Functional query language**, the computation is expressed as the *evaluation of functions* that may operate on data in the database or on the results of other functions;
  - functions are side-effect free, and they do not update the program state

Relational Query Languages

- "Pure" languages:
  - Relational algebra – functional query language
  - Tuple relational calculus - declarative
  - Domain relational calculus - declarative
- The above 3 pure languages are equivalent in computing power
- Not Turing complete

Introduction to relational algebra

Why is it required ?

- SQL is loosely based on relational algebra
- Many relational databases use relational algebra operations for representing execution plans
  - Simple, clean, effective mathematical abstraction for representing how results will be generated
  - Equivalent efficient queries can be generated during query optimization

Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Notation:

$$\Pi_{A_1,A_2,A_3 \ldots A_k}(r)$$

  where A_1, A_2, …, A_k 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

Project Operation Example

- Example: eliminate the *dept_name* attribute of *instructor*
- Query*:*

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

- Result:

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

Project Operation – selection of columns (Attributes)

- 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

Select Operation – selection of rows (tuples)

- Relation r

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

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

predicate

A	B	C	D
α	α	1	7
β	β	23	10

Note: This is different from SQL “select”

Select Operation (Cont.)

- We allow comparisons using

  $=, \neq, >, \geq. <. \leq$

  in the **selection predicate**.
- We can combine several predicates into a larger predicate by using the connectives:

  $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
- Example: Find the instructors in Physics with a salary greater $90,000, we write:

  $$\sigma_{dept\_name=\text{“}Physics\text{”} \wedge salary > 90,000}(instructor)$$
- The select predicate may include comparisons between two attributes.
  - Example, find all departments whose name is the same as their building name:
  - $\sigma_{dept\_name=building}(department)$

Union 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
α	2
β	1
β	3

- Union of two sets r and s

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

Question: Can this be expressed in terms of other fundamental operators?

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

Union, set difference and intersection

- For union, set difference and set intersection
  - *r, s* must have the *same* **arity** (same number of attributes)
  - 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*)

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

Cartesian-product – naming issue

- Relations *r, s*:

A	*B*
α	1
β	2

r

B	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*r.B*	*s.B*	*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

Renaming a Table

- Allows us to refer to a relation, (say E) by more than one name.

$$\rho_x(E)$$

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

- Relations *r*

A	*B*
α	1
β	2

r

- $r \times \rho_s(r)$

r.A	*r.B*	*s.A*	*s.B*
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- Instead of giving the name of a relation as the argument of the selection operation, we give an expression that evaluates to a relation

- $r \times s$

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

Composition of operators

- Find the set of all courses taught in the Fall 2009 semester
- find the set of all courses taught in the Spring 2010 semester
- Find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both.

See *section* below

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

Composition of operators

- Find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

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

The Assignment Operation

- Convenient at times to write a relational-algebra expression by
  - assigning parts of it to temporary relation variables.
- Denoted by $\leftarrow$ and works like assignment in a programming language.
- Example: Find all instructors in the "Physics" and Music department.

$$Physics \leftarrow \sigma_{dept\_name="Physics"}(instructor)$$

$$Music \leftarrow \sigma_{dept\_name="Music"}(instructor)$$

$$Physics \cup Music$$

- With the assignment operation
  - a query can be written as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as the result of the query.

Joining two relations – Natural Join

- 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 $R \cup S$ 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*

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 ⋈ 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

Question: Can this be expressed in terms of other fundamental operators?

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

Theta Join

- A variant of natural join where θ is a predicate on attributes in the schema R ∪ S. The theta join operation $r \bowtie_\theta s = \sigma_\theta (rXs)$
  - Take the Cartesian product of R and S
  - Select from tuples only those that satisfy the condition ɵ

Natural Join and Theta Join

A	B	C
1	2	3
6	7	8
9	7	8

Relation U

B	C	D
2	3	4
2	3	5
7	8	10

Relation V

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

Result $U \bowtie V$

A	U.B	U.C	V.B	V.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Result of $U \underset{A<D}{\bowtie} V$

Notes about Relational Languages

- Each Query input is a table (or set of tables)
- Each query output is a table.
- All data in the output table appears in one of the input tables
- Relational Algebra is not Turing complete
- Can we compute:
  - SUM
  - AVG
  - MAX
  - MIN

Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000
- Query 1

$$\sigma_{dept\_name=\text{“}Physics\text{”} \wedge salary > 90{,}000}(instructor)$$

- Query 2

$$\sigma_{dept\_name=\text{“}Physics\text{”}}(\sigma_{salary > 90{,}000}(instructor))$$

- The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department
- Query 1

$$\sigma_{dept\_name=\text{“Physics”}}(instructor \bowtie_{instructor.ID = teaches.ID} teaches)$$

- Query 2

$$(\sigma_{dept\_name=\text{“Physics”}}(instructor)) \bowtie_{instructor.ID = teaches.ID} teaches$$

- The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{salary >= 85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID, salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department*
	Output all pairs of rows from the two input relations (regardless of whether they have the same value on all attributes that have the same name).
∪ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(instructor) - \Pi_{name}(student)$
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

End of Chapter 2

Edited by Radhika Sukapuram. Original slides by

Database System Concepts, 6th Ed.