Chapter 2: Intro to Relational Model

Database System Concepts, 7th Ed.

Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages
- The Relational Algebra

Example of a *Instructor* Relation

attributes (or columns)

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

tuples (or rows)

Attribute

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is "unknown"
- 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** 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

  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one?
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

Keys (Cont.)

Employee (
EmployeeID,
FullName,
SSN,
DeptID
)

1. Candidate Key: are individual columns in a table that qualifies for uniqueness of all the rows. Here in Employee table **EmployeeID** & **SSN** are Candidate keys.

2. Primary Key: is the columns you choose to maintain uniqueness in a table. Here in Employee table you can choose either **EmployeeID** or **SSN** columns, **EmployeeID** is preferable choice, as SSN is a secure value.

3. Alternate Key: Candidate column other the Primary column, like if EmployeeID is PK then **SSN** would be the Alternate key.

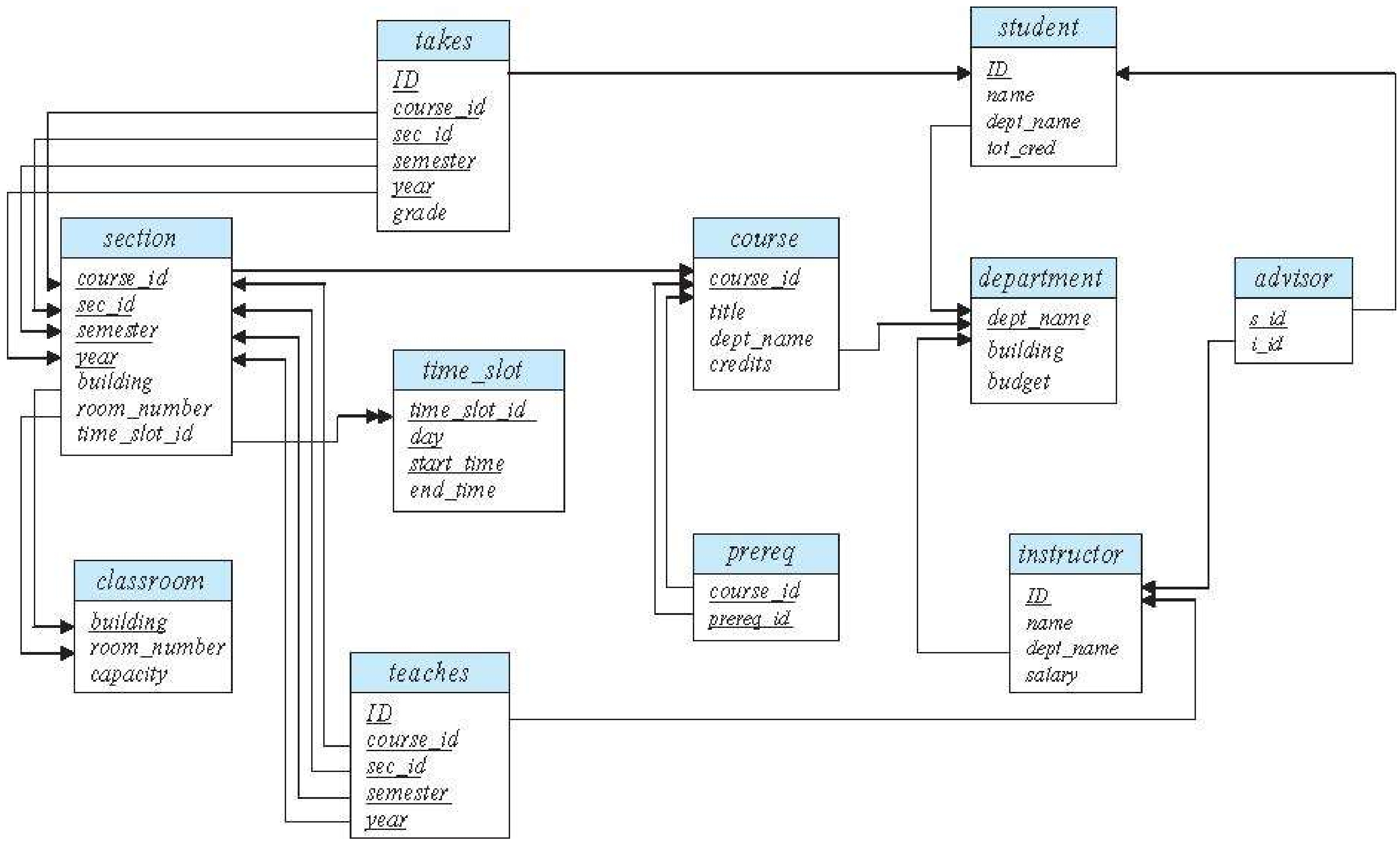
4. Super Key: If you add any other column/attribute to a Primary Key then it become a super key, like **EmployeeID + FullName** is a Super Key.

5. Composite Key: If a table do have a single columns that qualifies for a Candidate key, then you have to select 2 or more columns to make a row unique. Like if there is no **EmployeeID** or **SSN** columns, then you can make **FullName + DateOfBirth** as Composite primary Key. But still there can be a narrow chance of duplicate row.

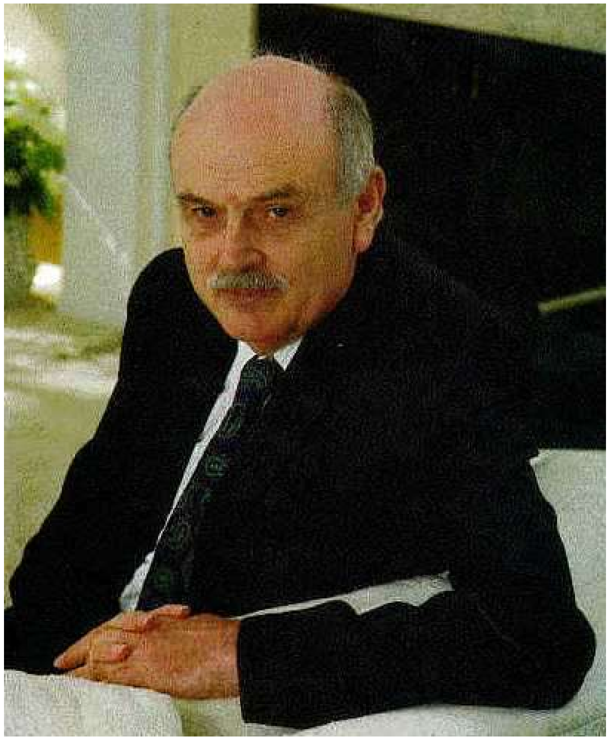
6. Foreign Key

7. Compound Key

Schema Diagram for University Database



Edgar F. Codd (1923-2003)



•PhD from U. of Michigan, Ann Arbor

•Received Turing Award in 1981.

•More see http://en.wikipedia.org/wiki/Edgar_Codd

Relational Query Languages

- Languages for describing queries on a relational database
- *Structured Query Language* (SQL)
  - Predominant application-level query language
  - Declarative
- *Relational Algebra*
  - Intermediate language used within DBMS
  - Procedural

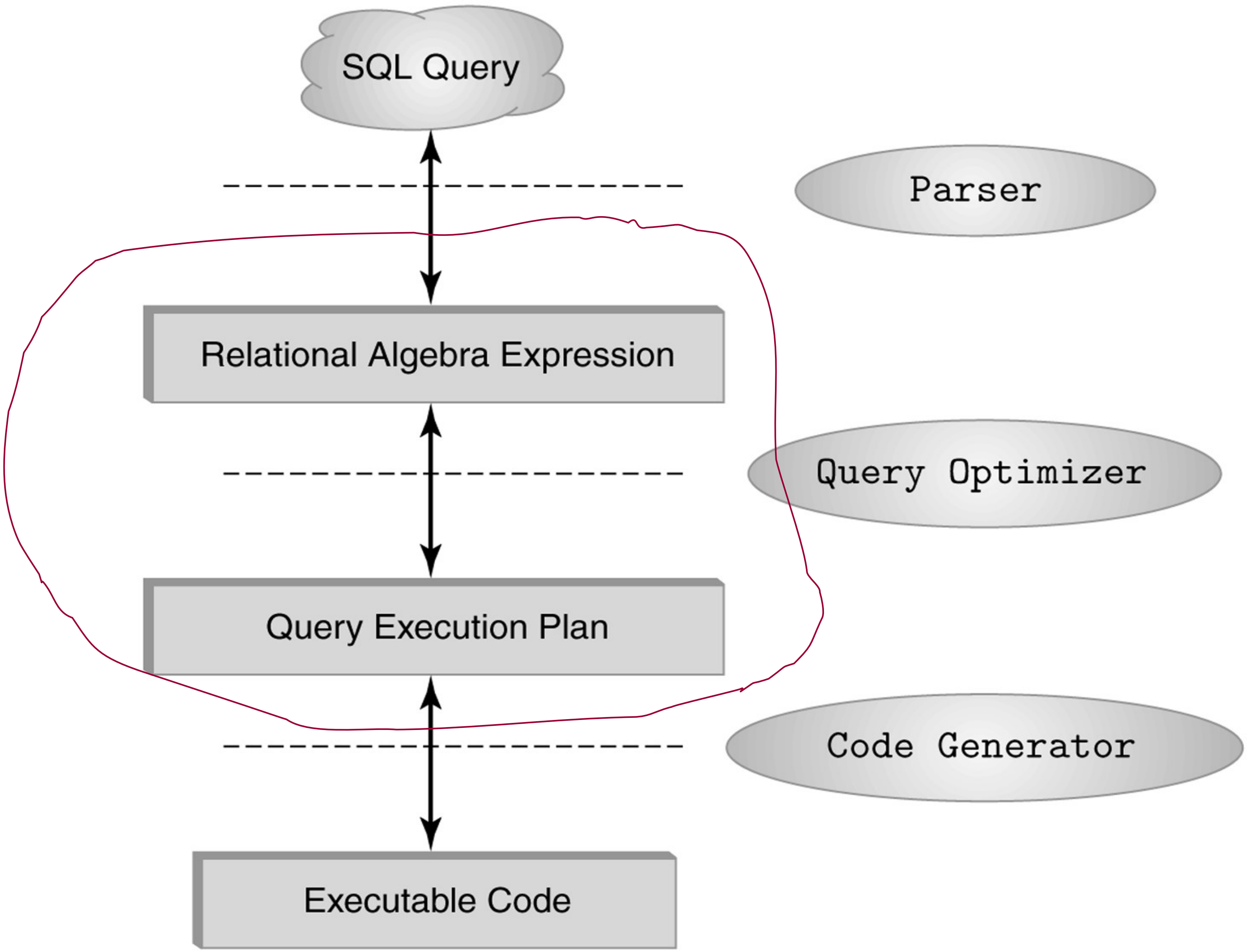
What is an Algebra?

- A language based on operators and a domain of values
- Operators map values taken from the domain into other domain values
- Hence, an expression involving operators and arguments produces a value in the domain
- When the domain is a set of all relations (and the operators are as described later), we get the *relational algebra*
- We refer to the expression as a *query* and the value produced as the *query result*

Relational Algebra

- *Domain*: set of relations
- *Basic operators*: select, project, union, set difference, Cartesian product
- *Derived operators*: set intersection, division, join
- *Procedural*: Relational expression specifies query by describing an algorithm (the sequence in which operators are applied) for determining the result of an expression

The Role of Relational Algebra in a DBMS

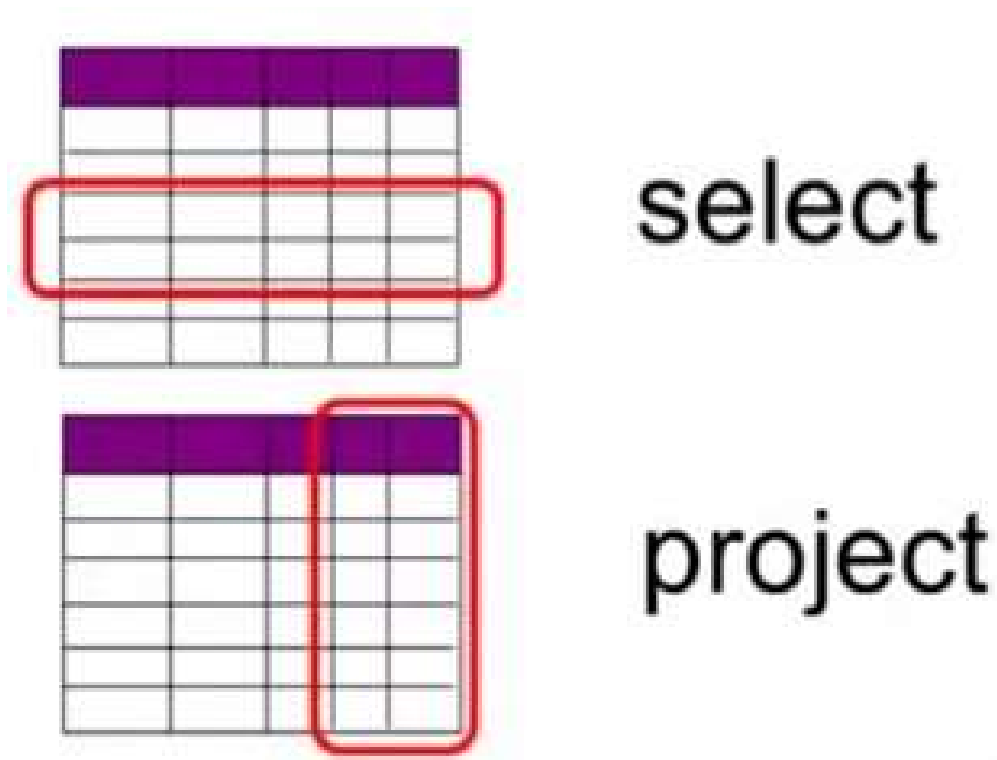


14

Relational Algebra

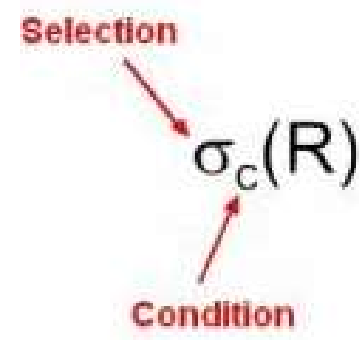
- A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.
- Six basic operators
  - Select (sigma): σ
  - Project (pi): Π
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - Rename (rho): ρ

Select and Project Operators



Select Operation

- The **select** operation selects tuples that satisfy a given condition.
- Produces **table** containing subset of rows of argument table satisfying condition
- Notation: $\sigma_c(R)$
- *c* is called the **selection condition**



Example:

Select those tuples of the *instructor* relation where the instructor is in the "Physics" department.

Query:

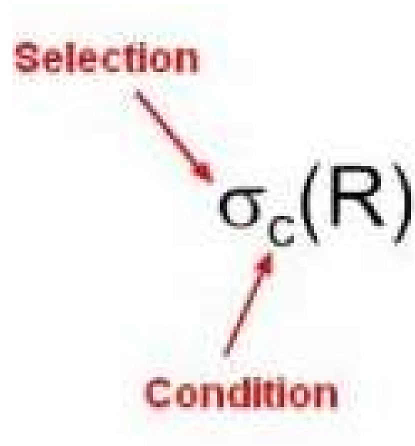
$$\sigma_{dept\_name\ =\ \text{“Physics”}}(instructor)$$

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

Result:

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

Select Operator



Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\sigma_{Hobby=\text{'stamps'}}(\text{Person})$

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
9876	Bart	5 Pine St	stamps

Selection Condition

- Operators: $<$, $\leq$, $\geq$, $>$, $=$, $\neq$
- Simple selection **condition**:
  - *<attribute> operator <constant>*
  - *<attribute> operator <attribute>*

- We can combine several predicates into a larger predicate by using the connectives:

  $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

- *<condition>* $\wedge$ *<condition>*
- *<condition>* $\vee$ *<condition>*
- $\neg$ *<condition>*

Selection Condition - Examples

- $\sigma_{Id > 3000 \ \vee \ Hobby = \text{'hiking'}}(\text{Person})$
- $\sigma_{Id > 3000 \wedge Id < 3999}(\text{Person})$
- $\sigma_{\neg(Hobby = \text{'hiking'})}(\text{Person})$
- $\sigma_{Hobby \neq \text{'hiking'}}(\text{Person})$

Select Operation (Cont.)

Example:

Find the instructors in Physics with a salary greater $90,000, we write:

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

Result:

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000

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

Truth Table

∧ (**AND**),
∨ (**OR**),
¬ (**NOT**)

A	B	A AND B	A OR B	NOT A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Select Operation (Cont.)

The select condition may include comparisons between two attributes.

Example:

Find all departments whose name is the same as their building name:

$$\sigma_{dept\_name=building}(department)$$

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

Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Produces table containing subset of columns of argument table
-
- Notation:

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

  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

Project Operation (Cont.)

- 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

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

Project Operator

$$\Pi_{A1,\ldots,An}(R)$$

Columns

Example:

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$\pi_{Name,Hobby}$(Person)

Name	*Hobby*
John	stamps
John	coins
Mary	hiking
Bart	stamps

Result is a table (no duplicates); can have fewer tuples than the original

Composition of Relational Operations

- The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**.
- Consider the query -- Find the names of all instructors in the Physics department.

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

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

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

Example

$$\pi_{\,Id,\ Name}\ (\sigma_{\,Hobby=\text{'stamps'}\ \lor\ Hobby=\text{'coins'}}\ (\text{Person})\ )$$

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

Person

Id	*Name*
1123	John
9876	Bart

Result

$$\sigma_{\,Hobby=\text{'stamps'}\ \lor\ Hobby=\text{'coins'}}\ (\pi_{\,Id,\ Name}\ (\text{Person})\ )\ ??$$

Cartesian Product

- If *R* and *S* are two relations, $R \times S$ is the set of all concatenated tuples <*x*, *y*>, where *x* is a tuple in *R* and *y* is a tuple in *S*
  - *R* and *S* need not be union compatible.
  - *But R* and *S* must have distinct attribute names. Why?
- $R \times S$ is expensive to compute.

A	B
x1	x2
x3	x4

R

C	D
y1	y2
y3	y4

S

A	B	C	D
x1	x2	y1	y2
x1	x2	y3	y4
x3	x4	y1	y2
x3	x4	y3	y4

$R \times S$

The size of this cartesian product is then the size of R multiplied by the size of S.

Cartesian-Product Operation

- The Cartesian-product operation (denoted by X) allows us to combine information from any two relations.
- Example: the Cartesian product of the relations *instructor* and t*eaches* is written as:

  instructor X *teaches*

- We construct a tuple of the result out of each possible pair of tuples: one from the *instructor* relation and one from the *teaches* relation (see next slide)
- Since the instructor *ID* appears in both relations we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came.
  - *instructor.ID*
  - *teaches.ID*

The *instructor* x *teaches* *table*

Instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Relational Algebra (RA)

- Basic operations:
  - ***Selection*** - Selects a subset of rows from relation.
  - ***Projection*** - Deletes unwanted columns from relation.
  - ***Cross-product*** - Allows us to combine two relations.
  - ***Set-difference*** - Tuples in reln. 1, but not in reln. 2.
  - ***Union*** - Tuples in reln. 1 and tuples in reln. 2.
  - ***Rename*** – Assigns a(nother) name to a relation
- Additional operations:
  - ***intersection, join, division, assignment***: not essential, but very useful
- The operators take one or two relations as inputs and give a new relation as a result.
- Operations can be ***composed.***

Selection

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

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

Set of tuples of r that satisfy p

Where *p* is a formula in propositional calculus consisting of

predicates

connectives : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

A **predicate** is one of:

<attribute> *op* <attribute> or

<attribute> *op* <constant>

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

- Result schema is same as r's schema

Selection Example 1

- 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

Selection Example 2

cust

cid	cname	rating	salary
21	Y. Yuppy	5	95
50	B. Rusty	10	65
55	S. Sneezy	8	70

$$\sigma_{rating>7}(cust)$$

cid	cname	rating	salary
50	R. Rusty	10	65
55	S. Sneezy	8	70

Projection

- Notation(pi): $\boldsymbol{\pi_{A_1, A_2, \ldots, A_k}(r)}$

  where $A_1, \ldots, A_k$ are attributes (the projection list) and r is a relation.
- The result = relation over the *k* attributes $A_1, A_2, \ldots, A_k$ obtained from r by erasing the columns that are not listed and **eliminating duplicate rows**.
- Remember: relations are sets!

Projection Example 1

- 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

Projection Example 2

Cust

<u>cid</u>	cname	rating	salary
38	R. Rudy	9	95
32	G. Grumpy	8	55
51	S. Sneezy	5	95
78	R. Rusty	10	55

π_{salary} (Cust)

salary
95
55

$\pi_{cname,\, rating}(Cust)$

cname	rating
R. Rudy	9
G. Grumpy	8
S. Sneezy	5
R. Rusty	10

$\pi_{cname,\, rating}(\sigma_{rating>7}(Cust))$?

Cartesian (or Cross)-Product

- Notation: $r \times s$
- 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, $R \cap S = \emptyset$).
- If r and s have common attributes, they must be renamed in the result.

Cartesian-Product Example 1

r

A	B
α	1
β	2

s

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

***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

Cartesian-Product Example 2

Customer

cid	cname	rating	salary
22	J. Justin	7	65
31	R. Rubber	8	85
58	N. Nusty	10	85

Order

cid	iid	day	qty
22	101	10/10/06	10
58	103	11/12/06	5

conflicting names

Customer x Order

Customer .cid	sname	rating	salary	Order. cid	iid	day	qty
22	J. Justin	7	65	22	101	10/10/96	10
22	J. Justin	7	65	58	103	11/12/96	5
31	R. Rubber	8	85	22	101	10/10/96	10
31	R. Rubber	8	85	58	103	11/12/96	5
58	N. Nusty	10	85	22	101	10/10/96	10
58	N. Nusty	10	85	58	103	11/12/96	5

Join Operation

- The Cartesian-Product

  instructor X *teaches*

  associates every tuple of instructor with every tuple of teaches.
  - Most of the resulting rows have information about instructors who did NOT teach a particular course.
- To get only those tuples of "*instructor* X *teaches* " that pertain to instructors and the courses that they taught, we write:

  $$\sigma_{instructor.id = teaches.id}\ (instructor \times teaches\ ))$$

  - We get only those tuples of "*instructor* X *teaches*" that pertain to instructors and the courses that they taught.
- The result of this expression, shown in the next slide

Join Operation (Cont.)

- The table corresponding to:

$$\sigma_{instructor.id\ =\ teaches.id}\ (instructor \times teaches))$$

Instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

Join Operation (Cont.)

- The **join** operation allows us to combine a select operation and a Cartesian-Product operation into a single operation.
- Consider relations *r* (*R*) and *s* (*S*)
- Let "theta" be a predicate on attributes in the schema R "union" S. The join operation r $\bowtie_\theta$ s is defined as follows:

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

- Thus

$$\sigma_{\textit{instructor.id} = \textit{teaches.id}}(\textit{instructor} \ \text{x} \ \textit{teaches})$$

- Can equivalently be written as

$$\textit{instructor} \bowtie_{\textit{Instructor.id} = \textit{teaches.id}} \textit{teaches}.$$

Union, Intersection, Set-Difference

- Notation: $\boldsymbol{r \cup s}$ $\boldsymbol{r \cap s}$ $\boldsymbol{r - s}$
- Defined as:

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

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

  $r - s = \{t \mid t \in r \text{ and } t \notin s\}$
- For these operations to be well-defined:
  1. *r, s* must have the *same arity* (same number of attributes)
  2. The attribute domains must be *compatible* (e.g., 2nd column of *r* has same domain of values as the 2nd column of *s*)
- What is the schema of the result?

Union Compatible Relations

- Two relations are *union compatible* if
  - Both have same number of columns (attributes)
  - Names of attributes are the same in both
  - Attributes with the same name in both relations have the same domain
- Union compatible relations can be combined using *union*, *intersection*, and *set difference*

Union, Int., Diff. Examples

Relations *r, s:*

r:

A	B
α	1
α	2
β	1

s:

A	B
α	2
β	3

$r \cup s$:

A	B
α	1
α	2
β	1
β	3

$r \cap s$:

A	B
α	2

$r - s$:

A	B
α	1
β	1

Union,Int., Diff. Examples

C1

cid	cname	rating	salary
22	J. Justin	7	65
31	R. Rubber	8	85
58	N. Nusty	10	85

C2

cid	cname	rating	salary
28	Y. Yuppy	9	95
31	R. Rubber	8	85
44	G. Guppy	5	70
58	N. Nusty	10	85

$C1 \cup C2$

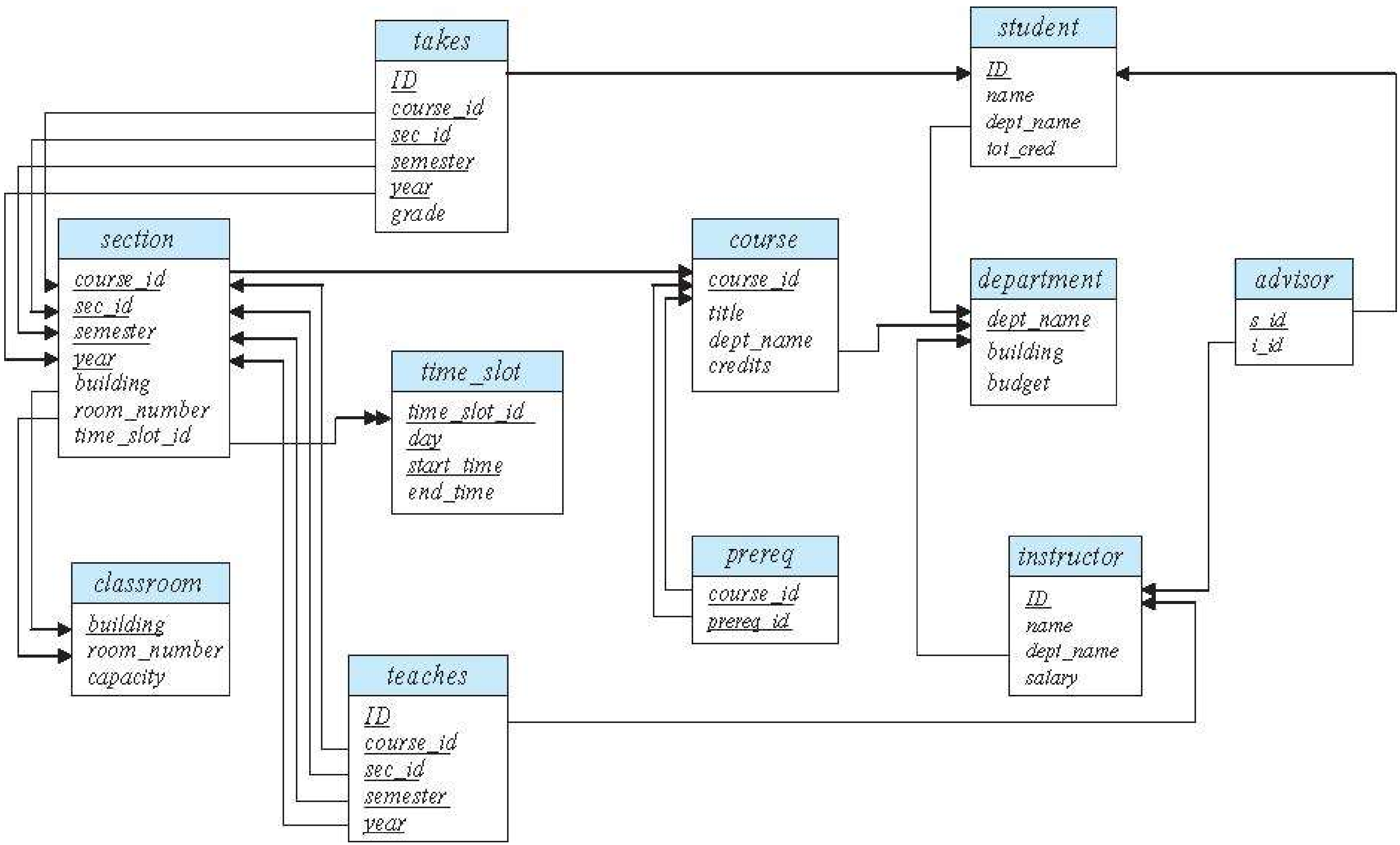
cid	cname	rating	salary
22	J. Justin	7	65
31	R. Rubber	8	85
58	N. Nusty	10	85
44	G. Guppy	5	70
28	Y. Yuppy	9	95

$C1 - C2$

cid	cname	rating	salary
22	J. Justin	7	65

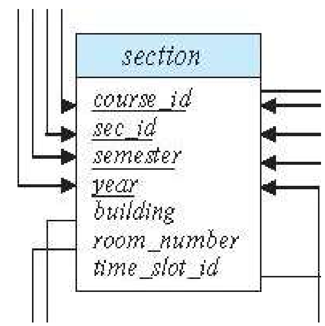
$C1 \cap C2$

cid	cname	rating	salary
31	R.Rubber	8	85
58	N. Nusty	10	85


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

Union Operation

- The union operation allows us to combine two relations
- **Example:** find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both



$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Fall”}\ \land\ year=2017}\,(section))\ \cup$$
$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Spring”}\ \land\ year=2018}\,(section))$$

Union Operation (Cont.)

- Result of:

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \land year=2017} (section)) \cup$$

$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \land year=2018} (section))$$

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

Set-Intersection Operation

- The set-intersection operation allows us to find tuples that are in both the input relations.
- **Example:** find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \land year=2017} (section)) \cap$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \land year=2018} (section))$$

Result:

course_id
CS-101

Set Difference Operation

- The set-difference operation allows us to find tuples that are in one relation but are not in another.
- **Example**: find all courses taught in the Fall 2019 semester, but not in the Spring 2020 semester

$$\Pi_{course\_id}(\sigma_{semester=\text{"Fall"} \land year=2019}(section)) -$$
$$\Pi_{course\_id}(\sigma_{semester=\text{"Spring"} \land year=2020}(section))$$

course_id
CS-347
PHY-101

The Assignment Operation

- It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables.
- The assignment operation is denoted by $\leftarrow$ and works like assignment in a programming language.
- **Example**: find all instructor in the "Physics" and Music department.

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

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

$$Physics \cup Music$$

- 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.

The Rename Operation

- The results of relational-algebra expressions do not have a name that we can use to refer to them.
- The rename operator, ρ, is provided for that purpose
- The expression:

$$\rho_{\boldsymbol{x}}\ (\boldsymbol{E})$$

returns the result of expression *E* under the name *x*

- Another form of the rename operation:

$$\rho_{x(A1,A2,\ ..\ An)}\ (E)$$

Rename Example

- cust(cid, cname, rating, salary).
- Find pairs of customer names (c1,c2) such that c1 is rated higher than c2 but is paid less.

In RA: $\pi_{cname,cust1.cname}(\sigma_{rating>cust1.rating \land salary<cust1.salary}$
$$(cust \times \rho_{cust1}(cust))).$$

$$\pi_{cname,cname'}(\sigma_{rating>rating' \land salary<salary'}$$
$$(cust \times \rho_{cid \rightarrow cid',cname \rightarrow cname',rating \rightarrow rating',salary \rightarrow salary'}(cust))).$$

Rename Example – another way

Transcript (*StudId*, *CrsCode*, *Semester*, *Grade*)

Teaching (*ProfId*, *CrsCode*, *Semester*)

$$\pi_{StudId,\ CrsCode} \text{(Transcript)}[StudId,\ CrsCode1]$$

$$\times\ \pi_{ProfId,\ CrsCode} \text{(Teaching)}\ [ProfId,\ CrsCode2]$$

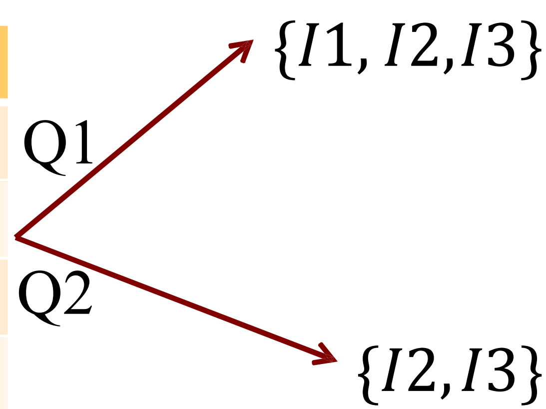
This is a relation with 4 attributes:

StudId, *CrsCode1*, *ProfId*, *CrsCode2*

≠ vs -

- Recall the relations cust(cid, cname, rating, salary) and ord(cid, iid, day, qty) and consider the queries:
- Q1: Find items (iid) ordered by someone other than the customer with cid 32.
- Q2: Find items in *ord* that are not ordered by customer with cid 32.

cid	iid	day	qty
32	I1	15/01/2013	5
23	I1	16/01/2013	3
23	I2	17/01/2013	2
16	I3	15/01/2013	2



An instance of *ord*

- We can express Q1 as $\pi_{iid}(\sigma_{cid \neq 32}(ord))$
- We can express Q2 as

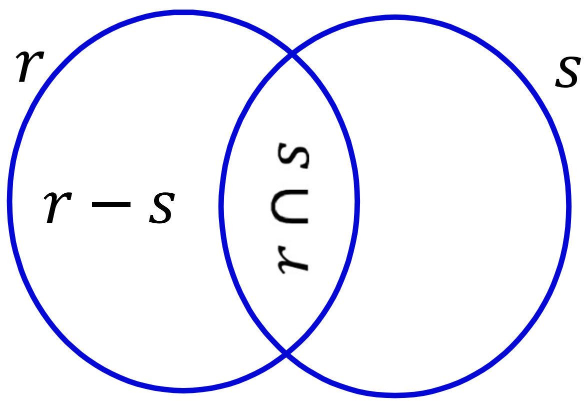
  $\pi_{iid}(ord) - \pi_{iid}(\sigma_{cid=32}(ord))$

Additional Operations

- They can be defined in terms of the primitive operations
- They are added for convenience
- They are:
  - Set intersection (we've seen it)
  - Join (Condition, Equi-, Natural)
  - Division
  - Assignment

Set intersection in terms of *minus*

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



Joins

Join: One of the most important ops implemented in a DBMS. Many efficient algorithms.

- *Condition Join*:

$$R \bowtie_c S = \sigma_c (R \times S)$$

- *Result schema* same as that of cross-product.
- Fewer tuples than cross-product
  - might be able to compute more efficiently
- Sometimes called a *theta-join*.

Condition Join Example

C1

cid	cname	rating	salary
22	J. Justin	7	80
31	R. Rubber	8	70
58	N. Nusty	10	90

O1

cid	iid	day	qty
22	101	10/10/96	10
58	103	11/12/96	5

$$C1 \bowtie_{C1.cid < O1.cid} O1$$

C1.cid	cname	rating	salary	O1.cid	iid	day	qty
22	J. Justin	7	80	58	103	11/12/96	5
31	R. Rubber	8	70	58	103	11/12/96	5

Equi-Join & Natural Join

- *Equi-Join*: A special case of condition join where the condition *c* contains only ***equalities***
  - *Result schema:* similar to cross-product, but contains only one copy of fields for which equality is specified
- *Natural Join*: Equijoin on *all* common attrs.
  - *Result schema:* similar to cross-product, but contains only one copy of each common field
  - no need to show the condition

Equi & Natural Join Examples

O1

cid	iid	day	qty
22	101	10/10/96	10
58	103	11/12/96	5

C1

cid	cname	rating	salary
22	J. Justin	7	85
31	R. Rubber	8	95
58	N. Nusty	10	90

$$C1 \bowtie_{C1.cid=O1.cid} O1$$

cid	cname	rating	salary	iid	day	qty
22	J. Justin	7	85	101	10/10/96	10
58	N. Nusty	10	90	103	11/12/96	5

$$C1 \bowtie O1$$

cid	cname	rating	salary	iid	day	qty
22	J. Justin	7	85	101	10/10/96	10
58	N. Nusty	10	90	103	11/12/96	5

Division

- Notation: ***r / s* or *r ÷ s***
- Useful for expressing queries that include a "**for all**" or "**for every**" phrase
- Let *r* and *s* be relations on schemas R and S respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  Then r/s is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

  defined as

  $$r / s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s \, (tu \in r) \}$$
- Informally, *r / s* contains the (parts of) tuples of r that are associated with every tuple in s.

Examples of Division A/B

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A/B1

sno
s1
s2
s3
s4

A/B2

sno
s1
s4

A/B3

sno
s1

More on Division

cust(cid, cname, rating, salary)
ord(cid, iid, day, qty)

Query: Find items (iid) that are ordered by **every** customer.

Don't know beforehand how many customers there are.

If there are 5 customers and you know their cid's (or look them up), how will you write this query in RA? What if there are 100?

Division lets us write this query concisely no matter how many customers …

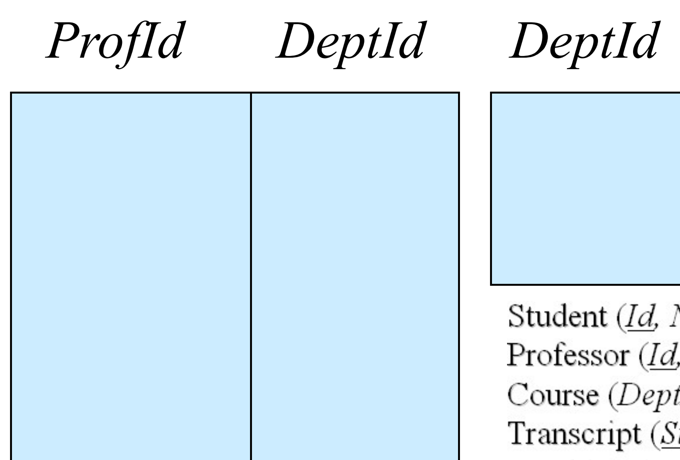
Division (contd.)

$$\pi_{iid,\,cid}(ord) \div \pi_{cid}(cust)$$

Division in SQL

- *Query type*: Find the subset of items in one set that are related to *all* items in another set
- *Example*: Find professors who taught courses in *all* departments
  - Why does this involve division?

Contains row <*p*,*d*> if professor *p* taught a course In department *d*



All department Ids

Student (Id, Name, Addr, Status)
Professor (Id, Name, DeptId)
Course (DeptId, CrsCode, CrsName, Descr)
Transcript (StudId, CrsCode, Semester, Grade)
Teaching (ProfId, CrsCode, Semester)
Department (DeptId, Name)

$$\pi_{\text{ProfId, DeptId}}(\text{Teaching} \bowtie \text{Course}) \;/\; \pi_{\text{DeptId}}(\text{Department})$$

Division Solution Sketch (1)

SELECT P.*Id*
FROM Professor P
WHERE P taught courses in **all** departments

SELECT P.*Id*
FROM Professor P
WHERE there does not **exist** any department that P has never taught a course

SELECT P.*Id*
FROM Professor P
WHERE NOT EXISTS (the departments that P has never taught a course)

Division Solution Sketch (1)

SELECT P.*Id*
FROM Professor P
WHERE NOT EXISTS(the departments that P has never taught a course)

⇩

SELECT P.*Id*
FROM Professor P
WHERE NOT EXISTS (
  B: All departments
  EXCEPT
  A: the departments that P has ever taught a course)

- But how do we formulate A and B?

Division – SQL Solution in details

```
SELECT  P.Id
FROM  Professor P
WHERE NOT EXISTS
   ( SELECT  D.DeptId          -- set B of all dept Ids
     FROM  Department D
     EXCEPT
    SELECT  C.DeptId          -- set A of dept Ids of depts in which P taught a course
    FROM  Teaching T, Course C
    WHERE  T.ProfId = P.Id      -- global variable
    AND  T.CrsCode = C.CrsCode)
```

Division (contd.)

$\pi_{iid,\, cid}(ord) \div \pi_{cid}(cust)$.

Notice the projections!
Notice the order of attrs!

- In RA, using only basic ops:
  $\pi_{iid}(ord) - \pi_{iid}((\pi_{cid}(cust) \times \pi_{iid}(ord)) - \pi_{cid,iid}(ord))$.
- Notice the correspondence between double negation and double minus. Make sure to understand why we need double minus (negation) for this query!
- Notice type compatibility: only items are being subtracted from items.

Expressing r÷s Using Basic *Operators*

- Generalizing from previous example …
- To express r÷s think as
- *Idea*:
  - let X = R-S (X is the set of attributes of R that are not in S)
  - (1) compute the X-projection of r
  - (2) compute all *X*-projection values of r that are `disqualified' by some value in *s*.
    - value *x* is *disqualified* if by attaching *y* value from *s*, we obtain an *xy* tuple that is not in *r*.
  - result is (1)-(2)
- So,
  - Disqualified *x* values: $$\pi_X((\pi_X(r) \times s) - r)$$
  - r ÷ s is $$\pi_X(r) - \pi_X((\pi_X(r) \times s) - r)$$

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

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

- Query 2

$$\sigma_{\textit{dept\_name}=\text{“}\textit{Physics}\text{”}}(\sigma_{\textit{salary} > 90.000}(\textit{instructor}))$$

- 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.

End of Chapter 2

Query

Professor(ssn, profname, status)

Course(crscode, crsname, credits)

Taught(crscode, semester, ssn)

Return those professors who have taught both ‘csc6710’ and ‘csc7710’.

Relational Algebra Solution

Professor(ssn, profname, status)

Course(crscode, crsname, credits)

Taught(crscode, semester, ssn)

$\pi_{ssn}(\sigma_{crscode=\text{'csc6710'} \wedge crscode=\text{'csc7710'}}(Taught)$, wrong!

$\pi_{ssn}(\sigma_{crscode=\text{'csc6710'}}(Taught)) \cap \pi_{ssn}(\sigma_{crscode=\text{'csc7710'}}(Taught))$, correct!

Query

Professor(ssn, profname, status)

Course(crscode, crsname, credits)

Taught(crscode, semester, ssn)

Return those professors who have never taught ‘csc7710’.

Relational Algebra Solution

Professor(ssn, profname, status)

Course(crscode, crsname, credits)

Taught(crscode, semester, ssn)

$\pi_{ssn}(\sigma_{crscode<>\text{'csc7710'}}(Taught))$, wrong answer!

$\pi_{ssn}(Professor)-\pi_{ssn}(\sigma_{crscode=\text{'csc7710'}}(Taught))$, correct answer!