Chapter 2: Intro to Relational Model and Chapter 6.1: Relational Algebra

Database System Concepts, 6th Ed.

Relational Databases

- Organize the data in a series of "tables" or "relations"
- Conceptually same as the spreadsheets we used in our early example
- The column headers are called "attributes"
- The elements in the tables are called "rows" or "tuples"

Terms

- You will find that there are similar vocabularies used to describe the parts of a table/row/columns. These terms are sometimes used interchangeably depending on the context.

Table	Row	Column
Relation	Tuple	Attribute
File	Record	Field

Example of a Relation

attributes (or columns)

tuples (or rows)

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 (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

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 $a_i \in D_i$

Relation Instance

- 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
- Attributes not necessarily unique (many rows can have same value in a column)

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

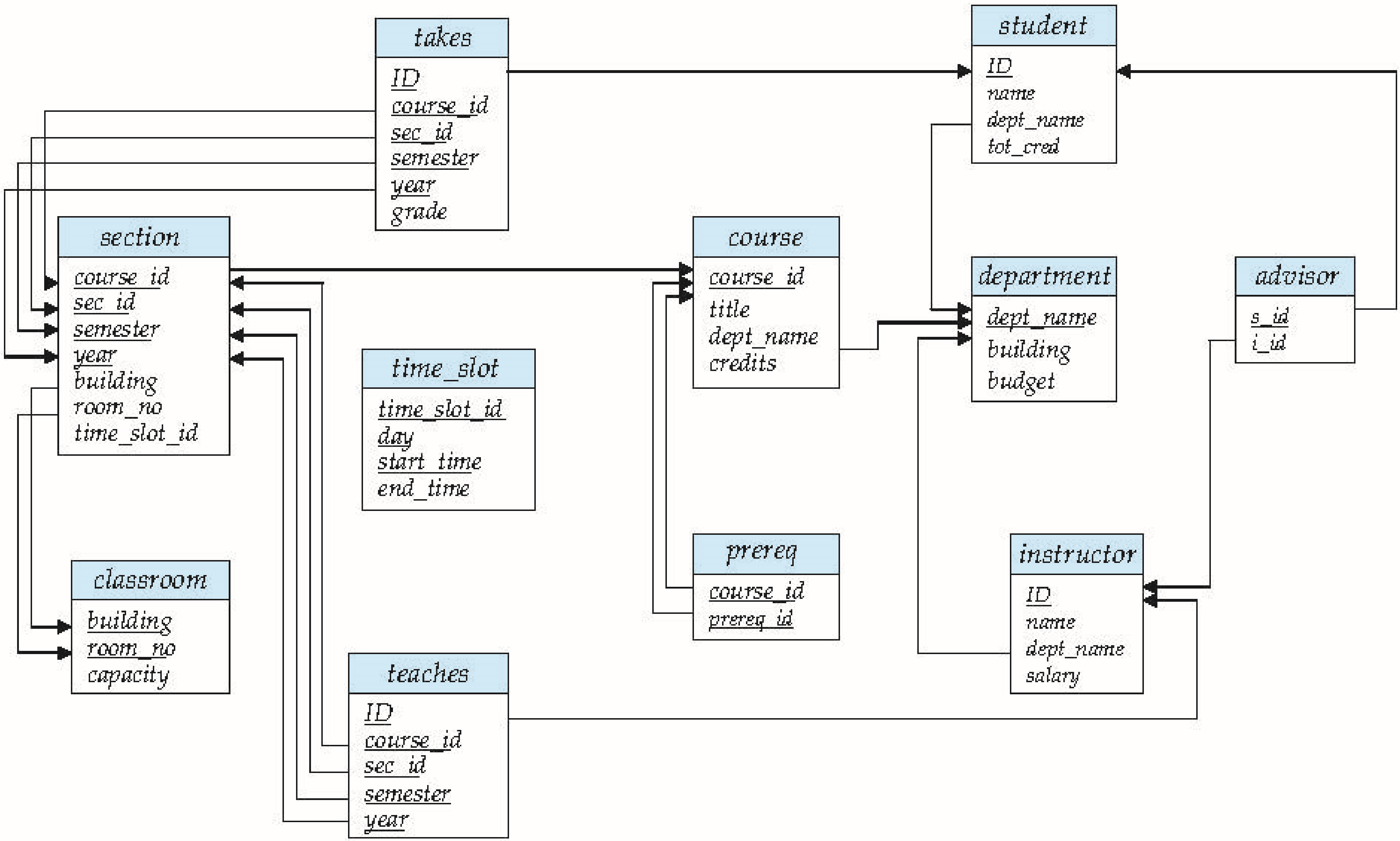
Keys

- Key – an attribute, or combination of attributes, that can be used to select a unique tuple
- 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?

Keys

- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*

Schema Diagram for University Database



Discussion

- Consider the following database, what are the appropriate primary keys?

  employee (person_name, street, city)

  works (person_name, company_name, salary)

  company (company_name, city)

- What are the foreign keys?

Relational Query Languages

- Procedural vs .non-procedural, or declarative
- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate on relational algebra
  - Not turing-machine equivalent

Select/Selection Operation

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

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

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> or <constant>

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

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

Select Operation – selection of rows

- 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

Example

Relation sells:

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

$\sigma_{bar=\text{“Joe's”}}(sells)$:

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75

By Jeffrey D. Ullman

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 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

Project Operation – selection of columns

- 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

Example

Relation sells:

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

$\Pi_{beer,price}(sells)$:

beer	price
Bud	2.50
Miller	2.75
Miller	3.00

By Jeffrey D. Ullman

Example

Figure 6.1
Results of SELECT and PROJECT operations. (a) $\sigma_{(\text{Dno}=4 \textbf{ AND } \text{Salary}>25000) \text{ OR } (\text{Dno}=5 \textbf{ AND } \text{Salary}>30000)}$(EMPLOYEE). (b) $\pi_{\text{Lname, Fname, Salary}}$(EMPLOYEE). (c) $\pi_{\text{Sex, Salary}}$(EMPLOYEE).

(a)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

(b)

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

From *Fundamentals of Database Systems*

Cartesian-Product Operation

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

Cartesian-Product Operation

- This operation is used to combine tuples from two relations in a combinatorial fashion.
- Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
- Result is a relation Q with degree n + m attributes:
  - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
- The resulting relation state has one tuple for each combination of tuples—one from R and one from S.
- Hence, if R has nR tuples (denoted as |R| = nR), and S has nS tuples, then R x S will have nR * nS tuples.
- The two operands do NOT have to be "type compatible”

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

Renaming a Table

- Useful when a query requires multiple operations
- Necessary in some cases (see JOIN operation later)

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
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

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.

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

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

Composition of Operations

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

- $r \times s$

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

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 theta-join operation r $\bowtie_\theta$ s is defined as follows:

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

- Thus

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

- Can equivalently be written as

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

Union Operation

- 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)

Union Operation

- 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 of two relations

- Relations *r, s:*

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

Example

Figure 6.3
Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

From *Fundamentals of Database Systems*

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

Set Difference Operation

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

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

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

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

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

Example

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

Figure 6.4
The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT $\cup$ INSTRUCTOR. (c) STUDENT $\cap$ INSTRUCTOR. (d) STUDENT – INSTRUCTOR. (e) INSTRUCTOR – STUDENT.

Some properties of UNION, INTERSECT, and DIFFERENCE

- Notice that both union and intersection are commutative operations; that is
  - $R \cup S = S \cup R$, and $R \cap S = S \cap R$
- Both union and intersection are associative operations; that is
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is not commutative; that is, in general
  - $R - S \neq S - R$

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.

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

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

$$Physics\ r \cup Music$$

- 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.

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

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

- Query 2

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

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

Equivalent Queries

- 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

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

More on Relational-Algebra Operations

- Generalized Projection
- Aggregate Functions
- SQL and Relational Algebra

Generalized Projection

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

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

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

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 Functions and Operations

- 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

Aggregate Operation Example

- Relation r

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

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

sum(*c*)
27

Aggregate Operation Example

- Find the average salary in each department

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

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

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 Operation Example

- Find the total number of instructors who teach a course in the Spring 2010 semester.

$$\mathcal{G}_{\textbf{count-distinct}(ID)}(\sigma_{semester=\text{“Spring”} \wedge year=2010}(teaches))$$

SQL and Relational Algebra

$$\begin{array}{l}\textbf{select } A_1,\ A_2, \ldots, A_n\\ \textbf{from } r_1,\ r_2, \ldots, r_m\\ \textbf{where } P\end{array}$$

is equivalent to

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

SQL and Relational Algebra

$$\begin{array}{l}\textbf{select } A_1,\ A_2, \textbf{sum}(A_3)\\ \textbf{from } r_1,\ r_2, \ldots, r_m\\ \textbf{where } P\\ \textbf{group by } A_1,\ A_2\end{array}$$

is equivalent to

$$_{A_1,\ A_2}\mathcal{G}_{\text{sum}(A_3)}(\Pi_{A_1,\ A_2,\ldots,\ A_n}(\sigma_P(r_1 \times r_2 \times \cdots \times r_m)))$$

Discussion

- Consider the following database

  employee (person_name, street, city)

  works (person_name, company_name, salary)

  company (company_name, city)

- 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.

Discussion

branch(*branch_name*, *branch_city, assets*)

customer (*customer_name*, *customer_street, customer_city*)

loan (*loan_number*, *branch_name, amount*)

borrower (*customer_name*, *loan_number*)

Give an expression in the relational algebra for each of the following queries.

a. Find the names of all branches located in “Chicago”.

b. Find the names of all borrowers who have a loan in the branch named “Downtown”.

Discussion

- Consider the university database

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)

Find the enrollment of each section that was offered in Fall 2009