数据库系统原理

Database System Principle

邵蓥侠

Email：shaoyx@bupt.edu.cn

北京邮电大学计算机学院

计算机应用技术中心

PART 1

RELATIONAL DATABASES

Chapter 2

Introduction to the Relational Model

Introduction to Chapter 2

- Relational ***data model***
  - relational data structure (***syntax***)
  - integrity constraints (***semantic***)
    - constraints on attributes of schemas, e.g. value domain, type
    - constraints on dependencies among attributes of a schema
    - constraints on dependencies among attributes of different schemas

      relational normal forms (*范式*， Chapter 7)
  - ***operations*** on the model

Main Parts in Chapter 2

- Relational data structure (syntax , § 2.1-2.4)
  - basic elements in relational data model, i.e. tables and attributes, and relationships among them

- Integrity constraints (semantic, Chapter 4)
  - constraints on keys in relational schema

- ***Relational Operations***

—***Relational algebra*** (关系代数, § 2.5, 2.6)

§ 2.1 Structure of Relational Databases

- A relational database consists of a collection of relational tables
  - e.g. instructor ▷ , course, prereq, department, sections, teaches
  - a row in a table represents a ***relationship*** among a set of values
  - a table is such a collection of relationships.

2.1.1 Basic structure

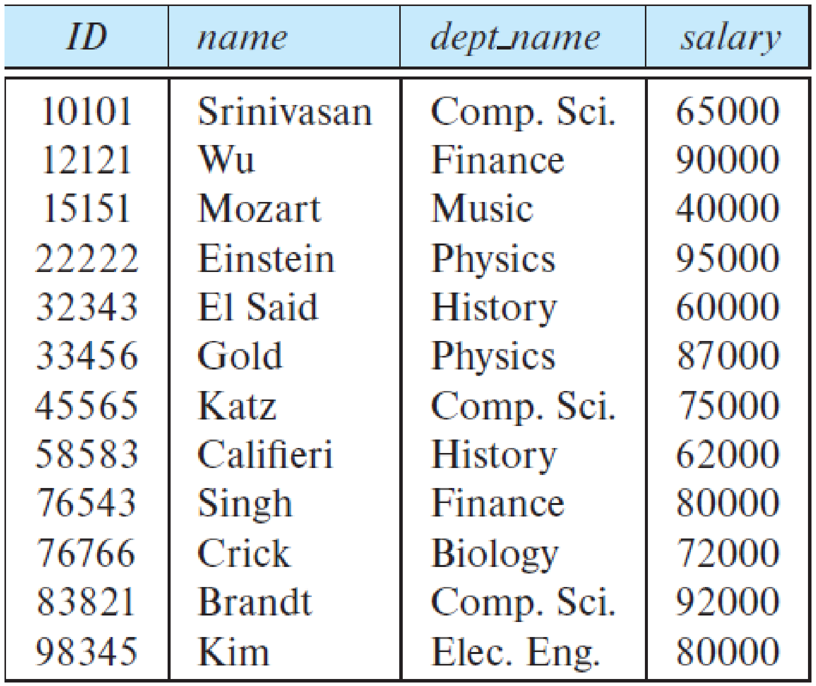
- Given sets D_1, D_2, …. D_n , a relation r is a subset of $D_1 \times D_2 \times \ldots \times D_n$
  - $r = \{(a_1, a_2, \ldots, a_n)\} \subseteq D_1 \times D_2 \ldots . \times D_n$
  - relation r = { tuple }   /*元组
- *定义：关系定义为一系列域上的笛卡尔积的子集*

北京邮电大学

2.1.1 Basic Structure (cont.)

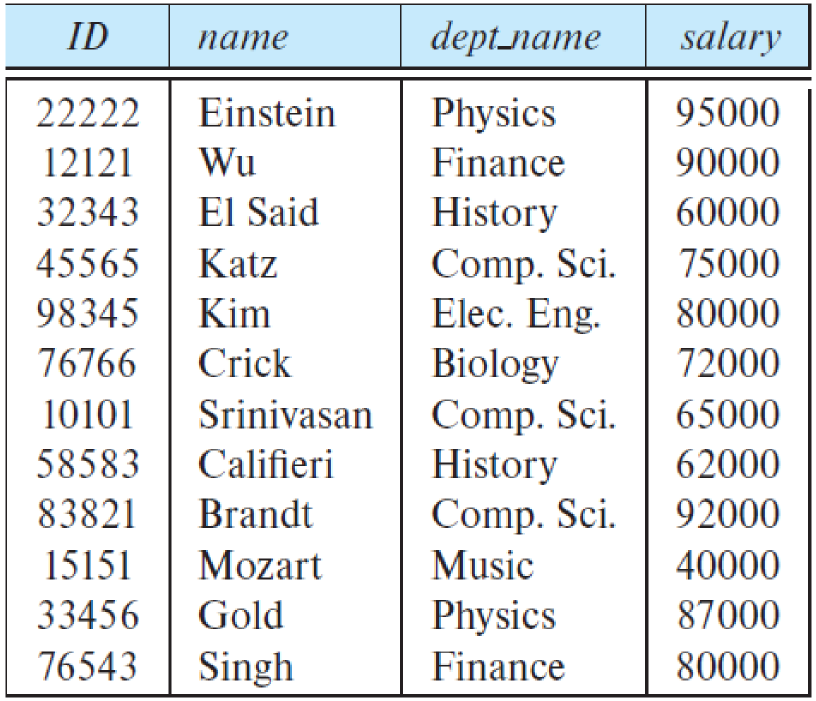
- ***Attributes*** of relation *r*
  - $A_1, A_2, \ldots, A_n$
  - ***domain*** of the relation's attributes $D_1, D_2, \ldots. D_n$
    - the set of allowed values for each attribute
- *Notes*
  - relation *r* in DBS is the limited set (有限集合)
  - attributes in tuples are non-ordered (无序性)
    - e.g. $(d_1, d_2, \ldots, d_n) = (d_2, d_1, \ldots, d_n)$
  - the order in which the tuples appear in a relations is irrelevant
    - Fig.2.1 vs. Fig.2.4
  - several attributes may have the same domain

2.1.1 Basic Structure (cont.)



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

Figure 2.1 The *instructor* relation.



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

Figure 2.4 Unsorted display of the *instructor* relation.

2.1.1 Basic Structure (cont.)

- the special value *null* is a member of every domain.
  - the *null* value causes complications in the definition of many operations
- Example
  - *student_name* = {Jones, Smith, Curry, Lindsay}
    depart= {CS, EE, Physics}
    student_city = {Harrison, Rye, Pittsfield}
  - then *r* = { (Jones, CS, Harrison),
    (Smith, EE, Rye),
    (Curry, EE, Rye),
    (Lindsay, Physics, Pittsfield)}

  is a relation over *student_name* $\times$ *depart* $\times$ *student_city*

2.1.1 Basic Structure (cont.)

- A domain is ***atomic*** if its elements are considered to be *indivisible*
  - all domains of R should be atomic, *first formal norm*
  - anti-e.g. multivalued attribute values are not atomic, { {1, 2}, {4}, {4, 6, 7} }
  - anti-e.g. composite（复合的）attribute *address*={*city*, *street*, *zipcode*} is not atomic

2.2 Database schema

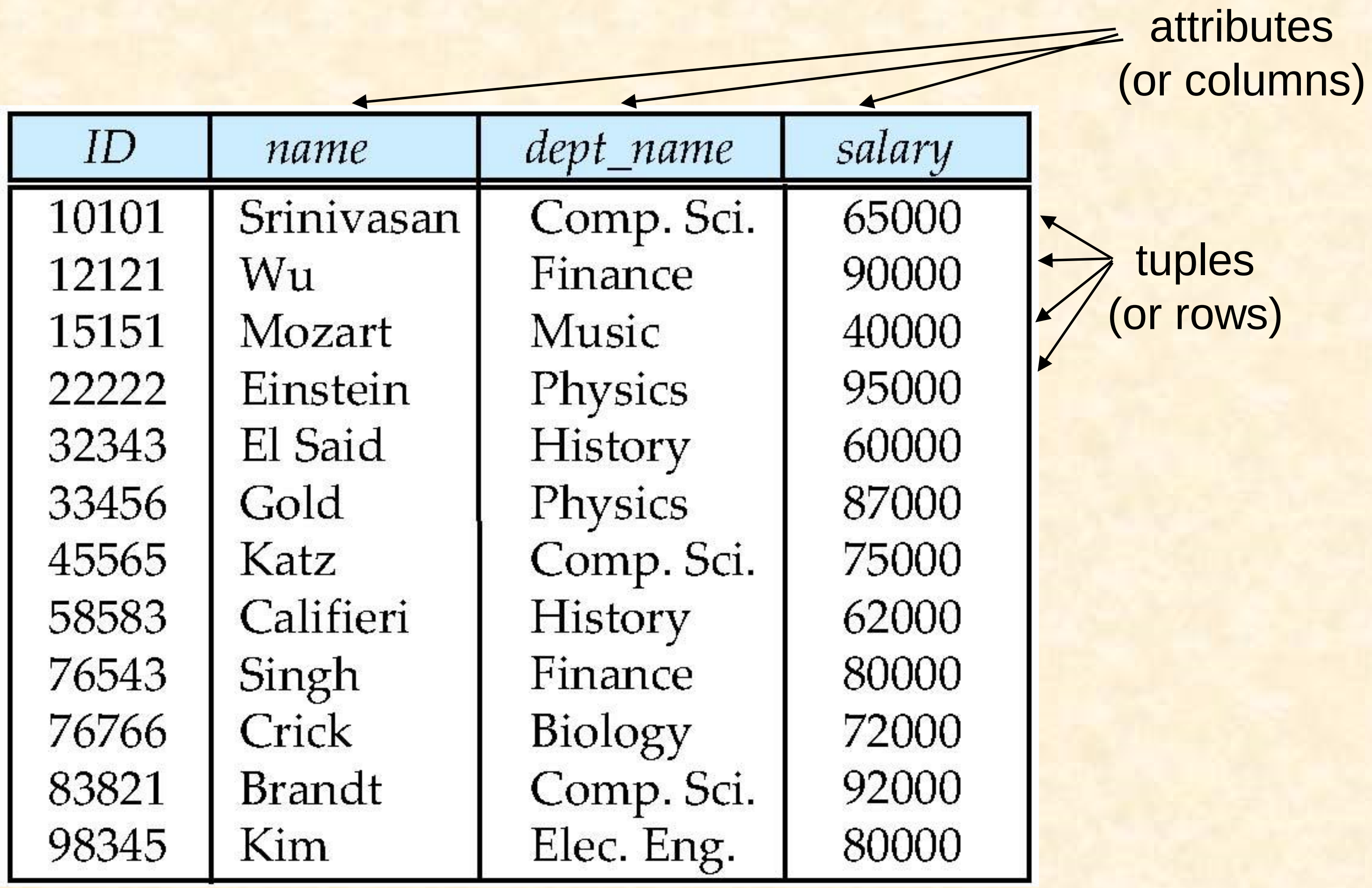
- For attributes $A_1, A_2, \ldots, A_n$, $R = (A_1, A_2, \ldots, A_n)$ is a ***relation schema***
  - e.g. in Fig. 2.1, ***Instructor****-schema* =
    (*ID*, *name*, *depart_name*, *salary*)
  - orders of attributes is irrelevant
- *r*(*R*) is a *relation* on the *relation schema R*
  - e.g. *customer (customer-name, customer-street, customer-city)* on *Customer-schema*
- *Relation instance*
  - the current values of a relation *r* at a particular time
  - specified by a table

§ 2.1.2 Relational schema (cont.)

- **the term *relation* and *relation instance* are used interchangeably in the textbook**

- Tabular（表格的） representation of *r*(R)
  - the current values of a relation *r*(R) (i,.e., *relation instance of the relation r*) are specified by a table
  - an element *t* in set *r* is a *tuple*, represented by a *row* in a table
  - tuple variable ***t***, $t[A_i]$ = the value of t on the attribute A_i , or $t[\{ A_i \}]$ = the value set of t on the attribute set $\{A_i\}$
    - e.g. in Fig.2.1, t2[*name*]= *Wu*
  - orders of tuples is irrelevant, i.e. tuples may be stored in an arbitrary order

Fig. 2.1 The *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

attributes
(or columns)
tuples
(or rows)

2.3 Keys

- Key：键，码
- A ***superkey*** is a set of one or more attributes that, taken collectively, can be used to identify uniquely a tuple in the relation
- **Def.** Given relation schema $R(A_1, A_2, \ldots, A_n)$, a subset K of R is the ***super key*** of R , if no two distinct tuples in relation $r(R)$ have the same values on all attributes in K
  - that is, if t_1 and t_2 are in r and $t_1 \neq t_2$, then $t_1[K] \neq t_2[K]$
  - the super key may contain redundant attributes

    e.g. {ID} vs {ID, *name*, *dept_name*}
- K is a ***candidate key*** if K is *minimal* super key
- 候选码：是最小的超码
  - e.g.{*ID*} is a candidate key for *instructor*, assuming no two instructors can possibly have the same identifiers

Keys (cont.)

- A relation may have several candidate keys. ***Primary key*** is a candidate key chosen by the database designer as principal（主要的） means to identify tuples within a relation

- ***Def.*** Primary attributes（主属性!）
  - taking the key as a attribute set, the attributes in the *candidate keys*, i.e. the element of the primary key is called the primary attributes
  - non-primary attributes（非主属性）

Keys (cont.)

- ***Def.*** If *X* is one or more attributes in relation r_1, and *X* is also the primary-key of another relation schema r_2

  taking Fig.2.8 as case studies

  - *X* is called a ***foreign key*** from r_1 referencing r_2
    - 由r_1参照/关联r_2的外键
  - r_1 is called the referencing relation of the foreign key dependency
    - 外键的参照/*关联关系*
  - r_2 is called the referenced relation of the foreign key
    - 外键的*被*参照/*关联关系*
  - e.g. *dept_name* in *instructor-schema* (Fig.2.8) and *department*-schema, *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*

Keys (cont.)

- A database schema, along with primary key and foreign key dependency, can be depicted as the ***schema diagram*****（模式图）**
  - e.g. Fig.2.8
  - node
    - schema, attributes, primary key
  - directed arc（有向弧）
    - foreign key dependency, from referencing relation r_1 to referenced relation r_2

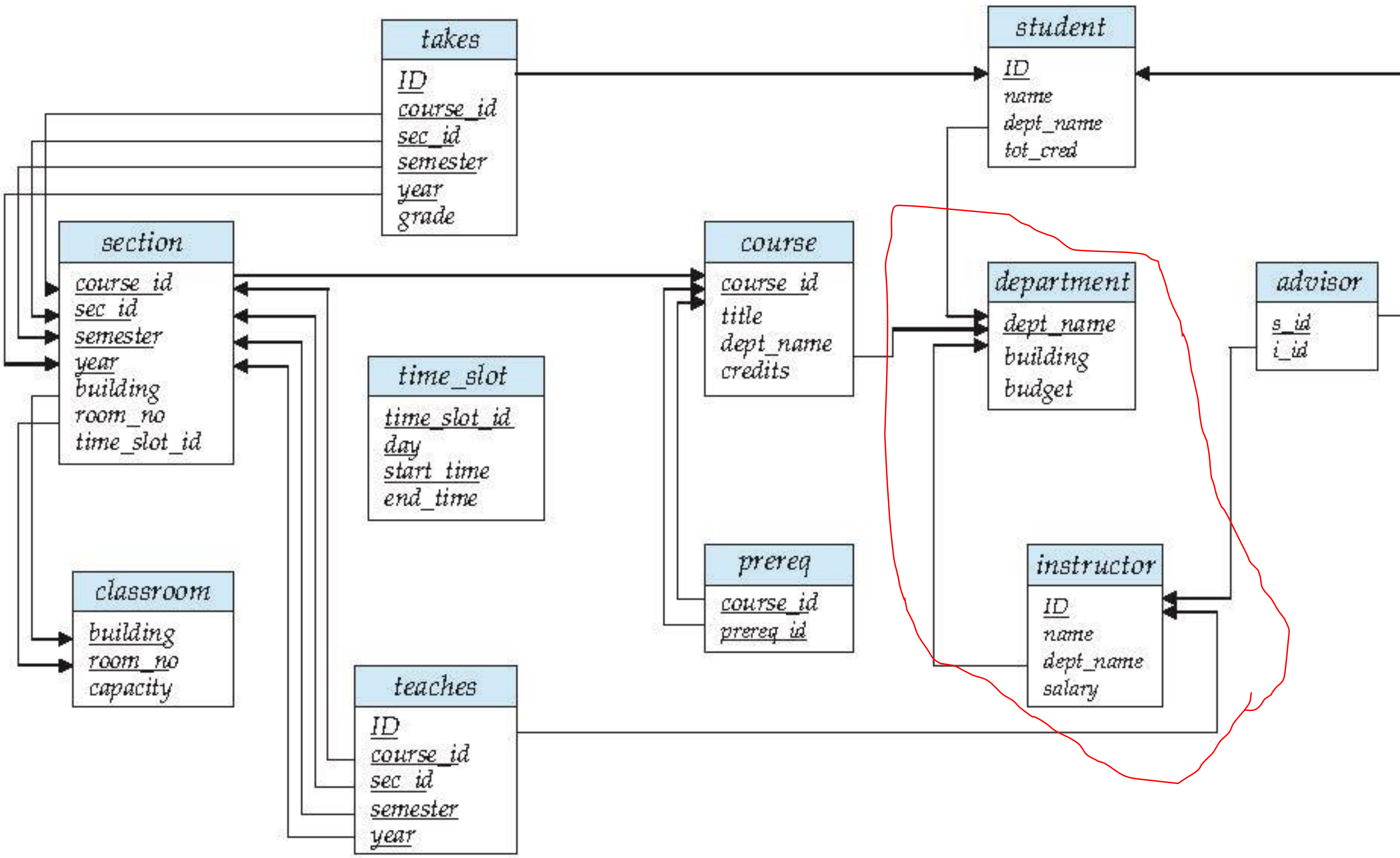


Fig. 2.8 Schema Diagram

Keys

- **Foreign key/Referential integrity** constraint
  - values appearing in specified attributes (e.g. *dept_name*)of any tuples in the referencing relation (e.g. *instructor*) must appear in specified attributes of at least one tuples in the referenced relation(e.g. *department*)

    – *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*
  - *why?*

  — the department that an instructor belongs to in the relation *instructor* （e.g. **Com.Sci**）must be a department that has been defined in the relation *department*, i.e. **Com.Sci** corresponds to a tuple in *department*

实验注意事项

- 如果数据库表之间存在外键关系，导入数据时，先导入被参照关系r_2 (e.g. *department*) 中的数据，再导入参照关系r_1 (e.g. *instructor*) 中的数据
  - e.g. 先定义各个department，再导入属于各个department的instructor的数据

2.5 Query language

- Query Language
  - the language in which user requests information from the database
- Categories of languages
  - Procedural（过程化的）：用户指导系统对数据库执行一系列操作以计算所需结果
  - Non-procedural（非过程化的）, or declarative(声明的)：用户只需描述所需信息，而不用给出获取该信息的具体过程。(SQL)
- “Pure” languages
  - relational algebra（关系代数）
  - tuple relational calculus（元组关系演算）
  - domain relational calculus（域关系演算）
- Pure languages form the underlying basis of query languages in practical uses

北京邮电大学

Relation Algebra Operations（关系代数操作）

- The relational algebra is *procedural*（过程化的） query language
  - a set of operations, take one or two *limited* relations as input and produce a new *limited* relation as output

- Three types of operations/operators on relations
  - *Fundamental*（基本的） operations, *additive* （附加的） operations, and *extended*（可扩展的） operations
  - similar *to* the logical operators in *propositional calculus* （命题演算）and *predicate calculus*（谓词演算）
    - fundamental : minimal complete set {┐, ∧}, or {┐, ∨}
    - additive: implication →, equivalence ↔
    - extended: ∀, ∃

6 basic relational algebra operators

Symbol (Name)	Example of Use
σ (Selection)选择	$\sigma_{\text{salary} >= 85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection) 投影	$\Pi_{ID,\ salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product) 笛卡尔积	*instructor* x *department*
	Output all pairs of rows from the two input relations whether or not they have the same value on all attributes that have the same name.
∪ (Union)并	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)集合差	$\Pi_{name}(instructor)$ -- $\Pi_{name}(student)$
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join) 自然连接	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

Select Operation（选择运算）

- Notation: **$\sigma_p(r)$**，选择出满足给定谓词的元组
- *p* is called the selection predicate（谓词）
- Defined as:

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

Where *p* is a formula in propositional calculus（命题演算公式） consisting of terms （项）connected by : $\wedge$ (and与), $\vee$ (or或), $\neg$ (not非)

Each term is one of:

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

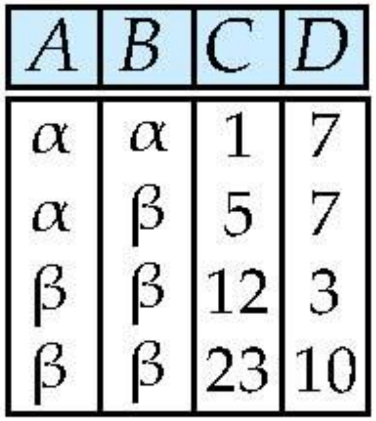
where *op* is one of: $=, \neq, >, \geq. <. \leq$

Example of selection:

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

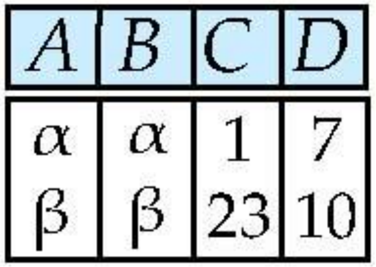
Select Operation – selection of rows (tuples)

Relation r



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

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



A	B	C	D
α	α	1	7
β	β	23	10

Project Operation（投影运算）

- Notation:

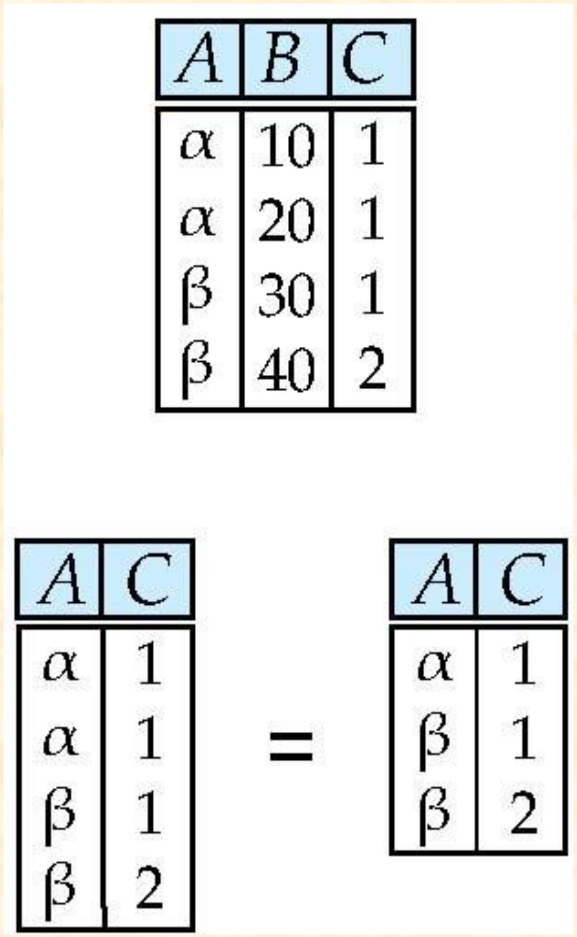
$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

  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（排除） the columns that are not listed
- Duplicate（重复的）rows are removed from result, since relations are sets（集合）
- Example: To eliminate the *dept_name* attribute of *instructor*

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

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

北京郵電大學

Union Operation（并运算）

- Notation: ***r* ∪ *s***
- Defined as:

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

- For *r* ∪ *s* to be valid.

  1. *r*, *s* must have the *same* arity（元） (same number of attributes 相同的属性个数)

  2. The attribute domains must be compatible（相容的）

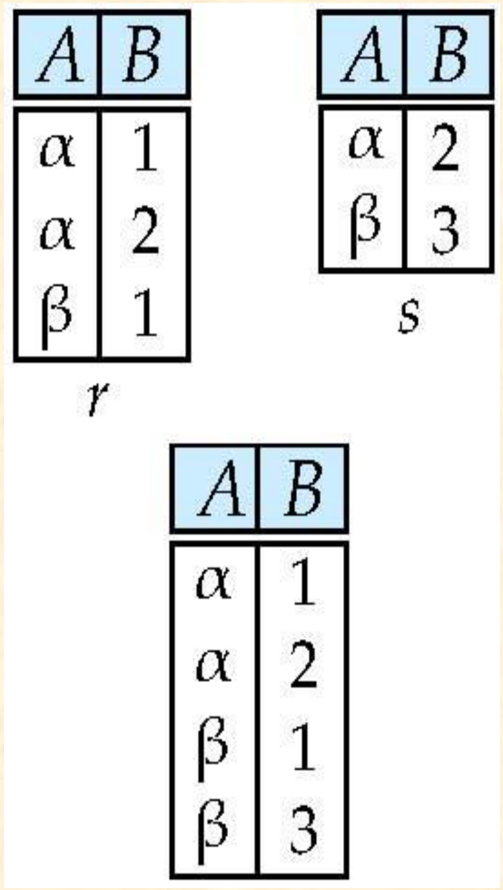
     (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

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

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

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

Set Difference Operation（集合差运算）

- Notation：***r – s***
- Defined as:

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

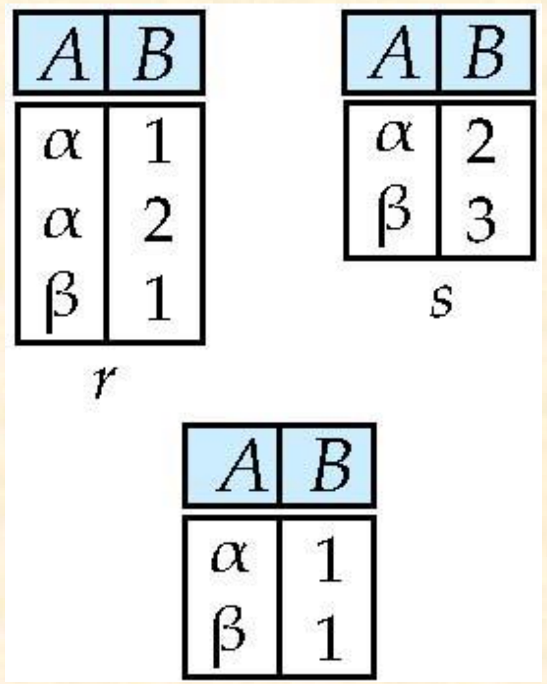
- Set differences must be taken between compatible relations.
  - *r* and *s* must have the same arity（元）
  - attribute domains of *r* and *s* must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

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

Set difference of two relations

- Relations *r*, *s*:

r – *s*:



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

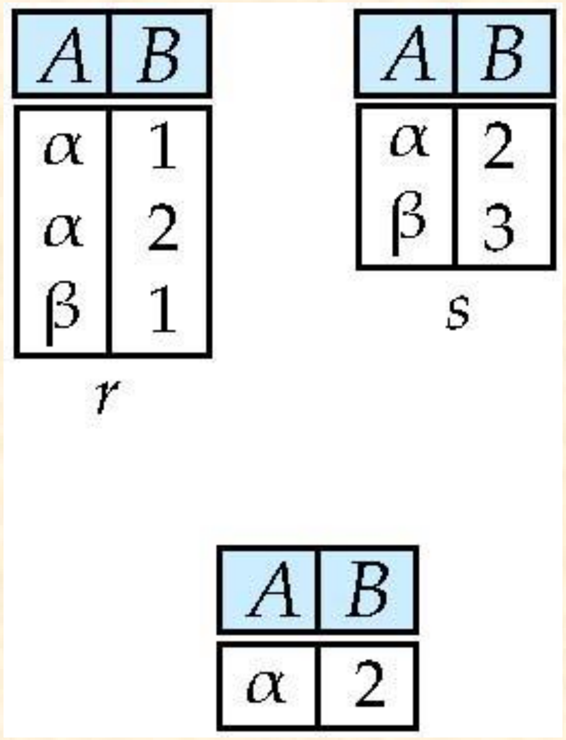
A	B
α	1
β	1

Set Intersection（集合交） Operation

- Notation: $\boldsymbol{r \cap s}$
- Defined as:
- $r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$
- 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

A	B
α	2

- $r \cap s$

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

Cartesian-Product Operation（笛卡尔积运算）

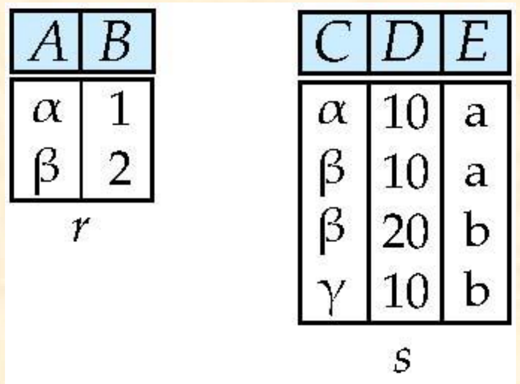
- Notation： ***r* x *s***
- Defined as:

$$r \text{ x } s = \{t\, q \mid t \in r \text{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint（不相交的）. (That is, $R \cap S = \varnothing$).
- If attributes of *r(R)* and *s*(*S*) are not disjoint, then renaming must be used. （为了加以区分，通过表名前缀进行重命名）

joining two relations -- Cartesian-product

Relations *r*, *s*:



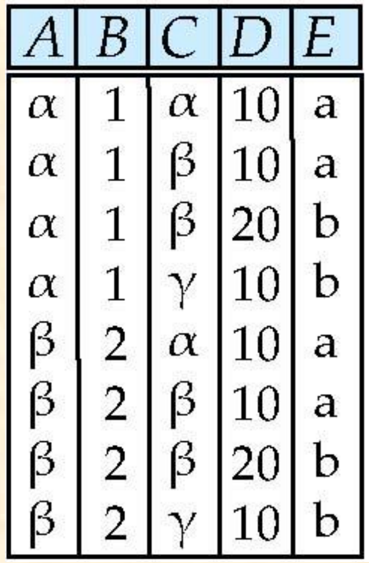
A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

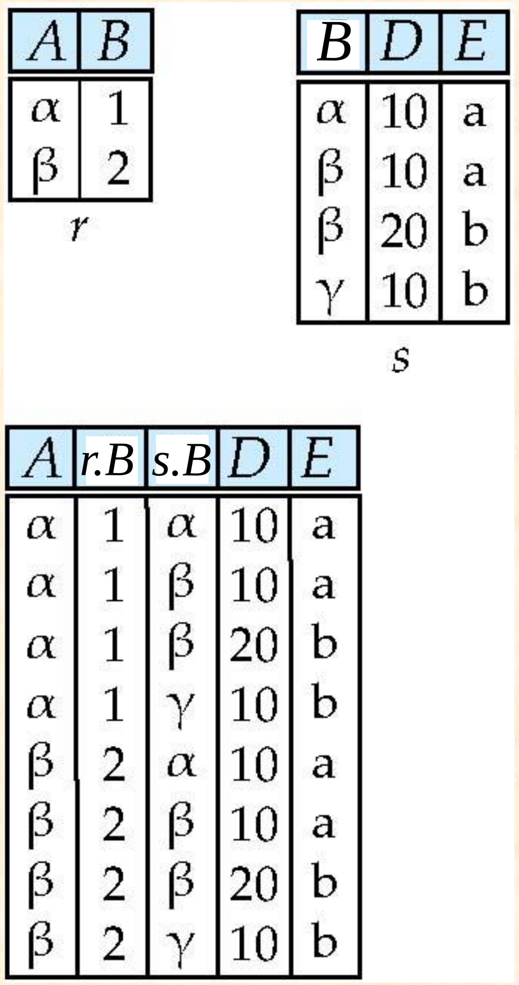


A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-product – naming issue

Relations *r*, *s*:

r x *s*:



A	B
α	1
β	2

r

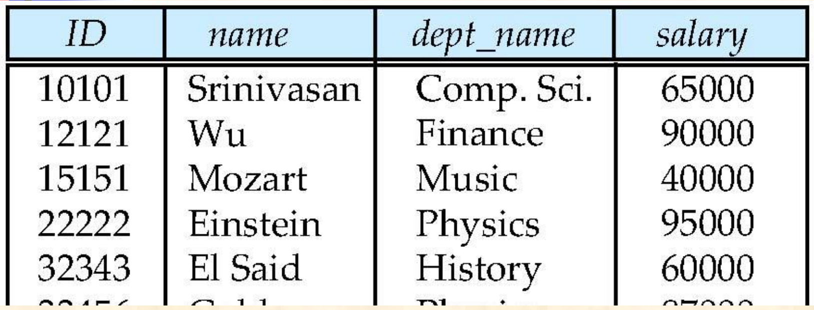
B	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

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

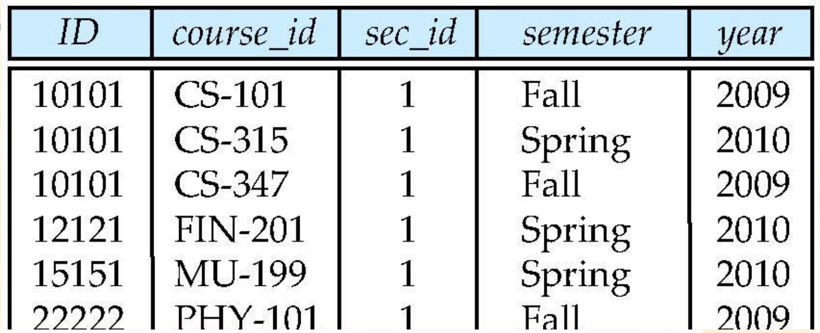
Cartesian Product

instructor

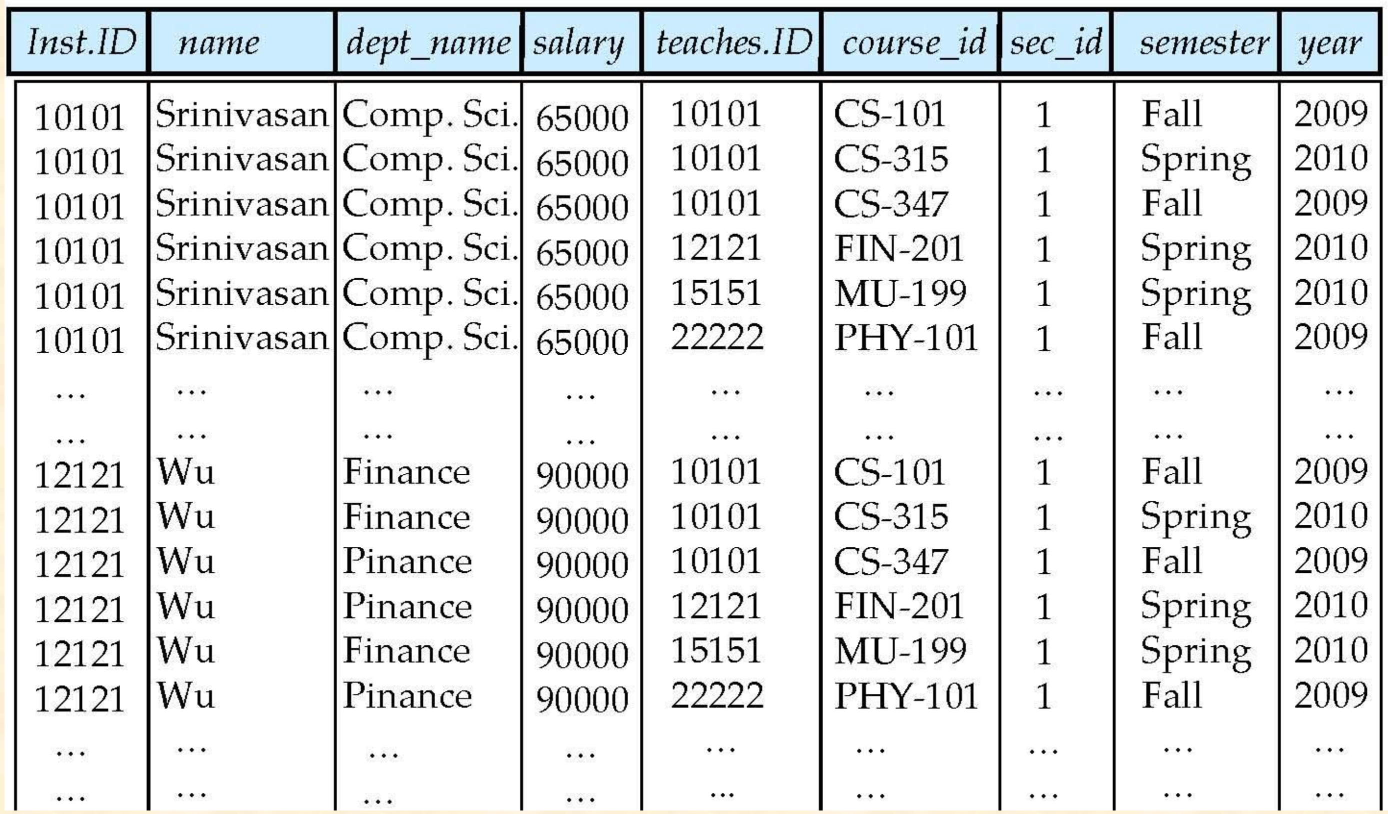


ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000

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



Inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Pinance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Pinance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Pinance	90000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

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（自然连接） Operation

- Example:

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

  $S = (E, B, D)$

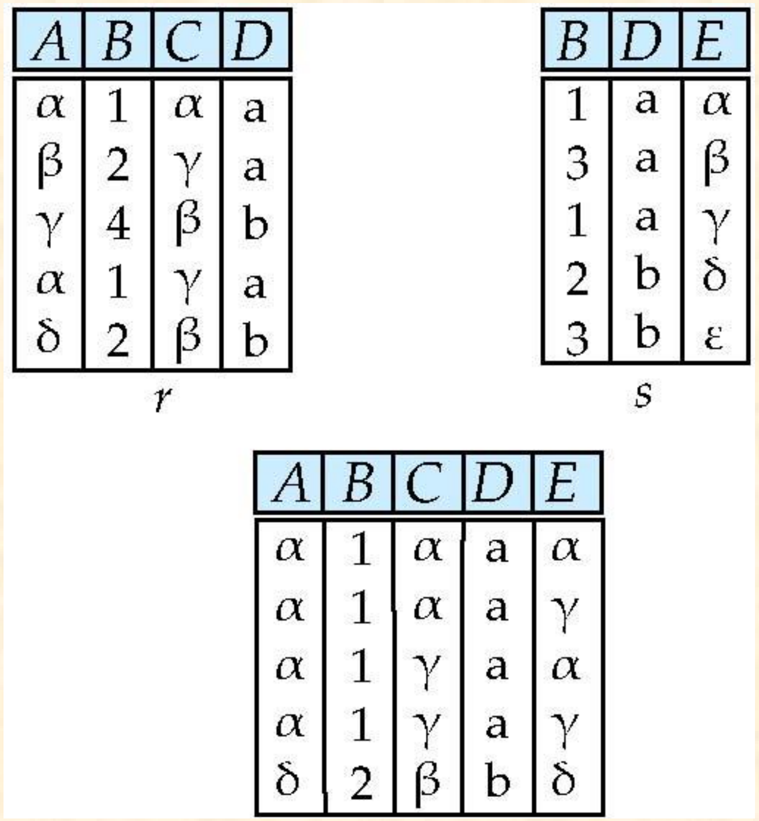
  - Result schema = (A, B, C, D, E)
  - $r \bowtie s$ is defined as:

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

注：自然连接运算首先形成它的两个参数的笛卡儿积，然后基于两个关系模式中都出现的属性上的相等性进行选择，最后还要去除重复属性。

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

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

Rename Operation

- The rename operation on relation *r* or relational algebra expression ***E***

$$\rho_x(r)\ ,\ \ \rho_x(E)$$

  - renaming ***r*** or ***E*** as ***X***

- If a relational-algebra expression ***E*** has arity *n*, then

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

returns the result of expression *E* under the name *X*, and with the attributes renamed to *A1, A2, …., An*

- E.g. $\rho_{\mathbf{s}\ (A1,\ A2,\ \ldots,\ An)}(\,\boldsymbol{r}\ (\mathrm{B}_1, \mathrm{B}_2, \ldots, \mathrm{B}_n)) = \mathbf{s}\ (A_1, A_2, \ \ldots,\ A_n)$

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

$r \times \rho_s(r)$

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

北京郵電大學

Rename Operation – An Example

- Find the highest *salary* in the *univsersity*
- Principle
  - /*将对一个关系*instructor*中同一个属性*salary*的不同取值的比较转换为利用换名操作、迪卡尔乘积和选择操作对关系*instructor*和它的副本ρ_d (*instructor*)在同一属性*salary*上取值的比较
- Step1.

$$instructor \times \rho_d (instructor)$$

- Step2. all tuples that have *non-largest salary* values in *instructor* relation:

$$\sigma_{instructor.salary < d.salary} (instructor \times \rho_d (instructor))$$

  - the result are tuple t_2, t_3, t_6

Rename Operation – An Example (cont.)

ins.ID	*ins.name*	*ins.salary*
120005	Zhang	1000
130007	Wang	2000
198032	Li	3000

(a) *instructor*

d.ID	*d.name*	*d.salary*
120005	Zhang	1000
130007	Wang	2000
198032	Li	3000

(b) $d = \rho_{d}(instructor)$

Fig.2.06-1 relation *instructor* and *d*

	instructor			d		
	ins.ID	*ins.name*	*ins.salary*	*d.ID*	*d.name*	*d.salary*
	120005	Zhang	1000	120005	Zhang	1000
t_2	120005	Zhang	1000	130007	Wang	2000
t_3	120005	Zhang	1000	198032	Li	3000
	130007	Wang	2000	120005	Zhang	1000
	130007	Wang	2000	130007	Wang	2000
t_6	130007	Wang	2000	198032	Li	3000
	198032	Li	3000	120005	Zhang	1000
	198032	Li	3000	130007	Wang	2000
	198032	Li	3000	198032	Li	3000

Fig.2.06-2 *instructor* $\times \rho_d$ (*instructor*)

Rename Operation – An Example (cont.)

- Step3. all *non-largest salary* values in relation *instructor*

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

  - the result is {*1000, 2000*}

- Step4. All largest balance values in relation *account*

  $\Pi_{salary}(instructor)$ —

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

  - the result is {1000, 2000, 3000} — {*1000*, *2000*} = {3000}

Rename Operation – An Example (cont.)

- This query can also be solved by ***Select*** statement in SQL, by means of
  - ***tuple variables*** in §3.3.5?
  - ***set comparison*** in §3.7.2?
  - ***aggregate function*** *max* in §3.5?
- Homework and Questions
  - 是否可用下式代替step4中的查询表达式

$$\Pi_{instructor.salary}\,(\sigma_{instructor.salary >= d.salary}\,(instructor \times \rho_d\,(instructor)))$$

  - 如何求解 “Find the *smallest salary* in the university ”

Composition of Operations

- Can build expressions using multiple 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

- 查询计算机系所有教师的名字：

$\Pi_{name}(\sigma_{dept= \text{“Comp.Sci.”}}(\text{instructor}))$

Composition of Operations

- 查询所有教师的姓名，连同他们教的所有课程的course_id

  $\Pi_{name,\ course\_id}$ (instructor $\bowtie$ teaches)

- 查询计算机系的所有教师，以及他们教授的所有课程的名称

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

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 Turning complete
- Can we compute ?
  - SUM
  - AVG
  - MAX
  - MIN

Conclusion

- The relational data model is based on a collection of tables.
- The schema of a relation refers to its logical design, while an instance of the relation refers to its contents at a point in time.
- The schema of a relation includes its attributes, and optionally the types of the attributes and constraints on the relation such as primary and foreign-key constraints.
    - superkey, key, candidate key, foreign-key
- Relational query language
    - Relational algebra

Exercises

1.8 Explain the concept of physical data independence and its importance in database systems.

1.11 Assume that two students are trying to register for a course in which there is only one open seat. What component of a database system prevents both students from being given that last seat?

2.1 Consider the employee database of Figure 2.17 (in textbook). What are the appropriate primary keys?

2.10 Describe the differences in meaning between the terms *relation* and *relation schema*.

Exercises

2.6 练习题：考虑如下关系数据库。

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

给出关系代数表达式来表示下列每一个查询：

1）找出居住在“**Miami**”城市的所有员工姓名。

2）找出工资在**100000**美元以上的所有员工姓名。

3）找出居住在“**Miami**”并且工资在**100000**美元以上的所有员工姓名。

Exercises

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

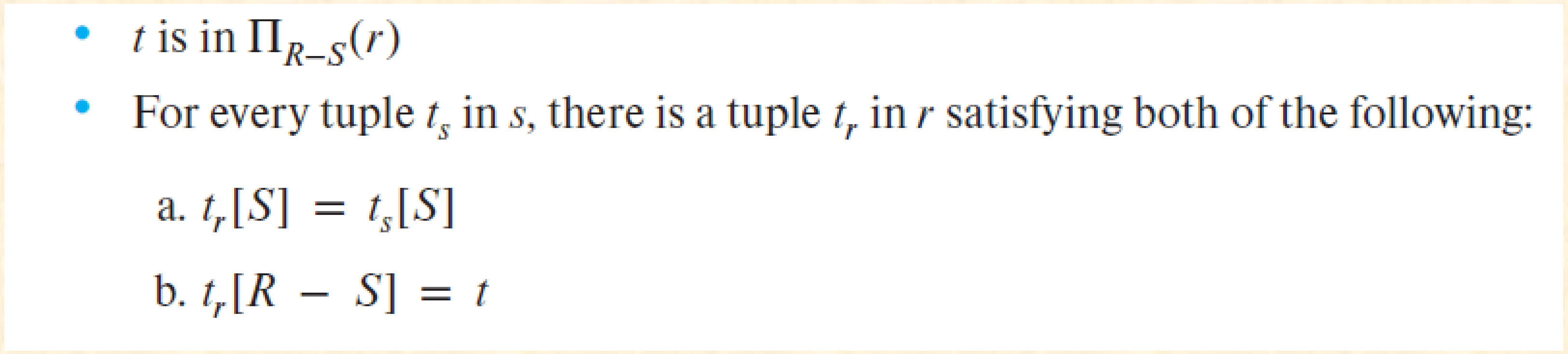
4）找出**First Bank Corparation**所有员工的姓名和居住城市。

5）找出**First Bank Corparation**所有年收入在**10000**美元以上的员工姓名和居住的街道、城市。

6）找出所有居住地与工作的公司在同一城市的员工姓名。

Exercises

2.9 The **division operator** of relational algebra, "÷", is defined as follows. Let *r*(*R*) and *s*(*S*) be relations, and let $S \subseteq R$; that is, every attribute of schema *S* is also in schema *R*. Given a tuple *t*, let *t*[*S*] denote the projection of tuple *t* on the attributes in *S*. Then $r \div s$ is a relation on schema $R - S$ (that is, on the schema containing all attributes of schema *R* that are not in schema *S*). A tuple *t* is in $r \div s$ if and only if both of two conditions hold:



- t is in $\Pi_{R-S}(r)$
- For every tuple t_s in s, there is a tuple t_r in r satisfying both of the following:

  a. $t_r[S] = t_s[S]$

  b. $t_r[R - S] = t$

Exercises

Given the above definition:

- Write a relational algebra expression using the division operator to find the IDs of all students who have taken all Comp. Sci. courses. The database schema is shown in Figure 2.8, and only *takes* and *course* are needed.. (Hint: project *takes* to just ID and *course id*, and generate the set of all Comp. Sci. *course id*s using a select expression, before doing the division.)
- Show how to write the above query in relational algebra, without using division. (By doing so, you would have shown how to define the division operation using the other relational algebra operations.)