Lecture 2

Relational Model

Outline

- Structure of Relational Databases
- Fundamental Relational-Algebra-Operations
- Additional Relational-Algebra-Operations
- Extended Relational-Algebra-Operations

Relational Database

- Relation

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Relational Database

- Relation
  – Is used to refer to a table
  – conceptual representation of a table
- Attribute
  – A *column* in a *table*
- Tuple
  – A *row* in a *table*
  – A list of values

属性 attributes (列 columns)

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

元组 tuples (行 rows)

Relational Database

- Attributes
  - **Each attribute of a relation has a name**
  - **Domain of the attribute**
    - **the set of allowed values for the attribute**
  - **Atomicity**
    - **Attributes are normally required to be atomic, i.e. indivisible**
    - **multivalued attribute 多值属性**
    - **composite attribute 复合属性**
  - **Null**
    - **The special value *null* is a member of every domain**

Relational Database

- Relation Instance: A specific instance of a relation (containing a specific set of rows)
- Relation Schema: The structure of a relation
  - *instructor* is a relation

    instructor (id, name, dept_name, salary)
  - Corresponding relation schema

    R = (*id, name, dept_name, salary*)
  - *instructor* is a relation on relation schema R

    instructor(R)
- In general:

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

  A relation defined over schema R is denoted by $r(R)$.

Relational Database

- Given sets of values D_1, D_2, …. D_n, a relation r is a subset of
  $D_1 \times D_2 \times \ldots \times D_n$

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

{22222,Einstein, Physicis, 95000}

{12121, Wu, Finance, 90000}

……

D_1={22222, 12121, ……}

D_3 ={Physics, Finance, History, Comp. Sci., ……}

D_4 ={95000, 90000, 85000, 80000, 75000, ……}

Relational Database

- **Order of tuples is irrelevant**
  - We do not know the order of the tuples.
  - We have to specify the order if we want to visit the tuples orderly.
  - We can order the tuples by any attribute(s) that we specify.

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

Relational Database

- Database
  - A database consists of multiple relations
  - Each relation stores one part of information
    - Information about an enterprise is broken up into parts

student : stores information about students

advisor : stores information about which instructors advise which students

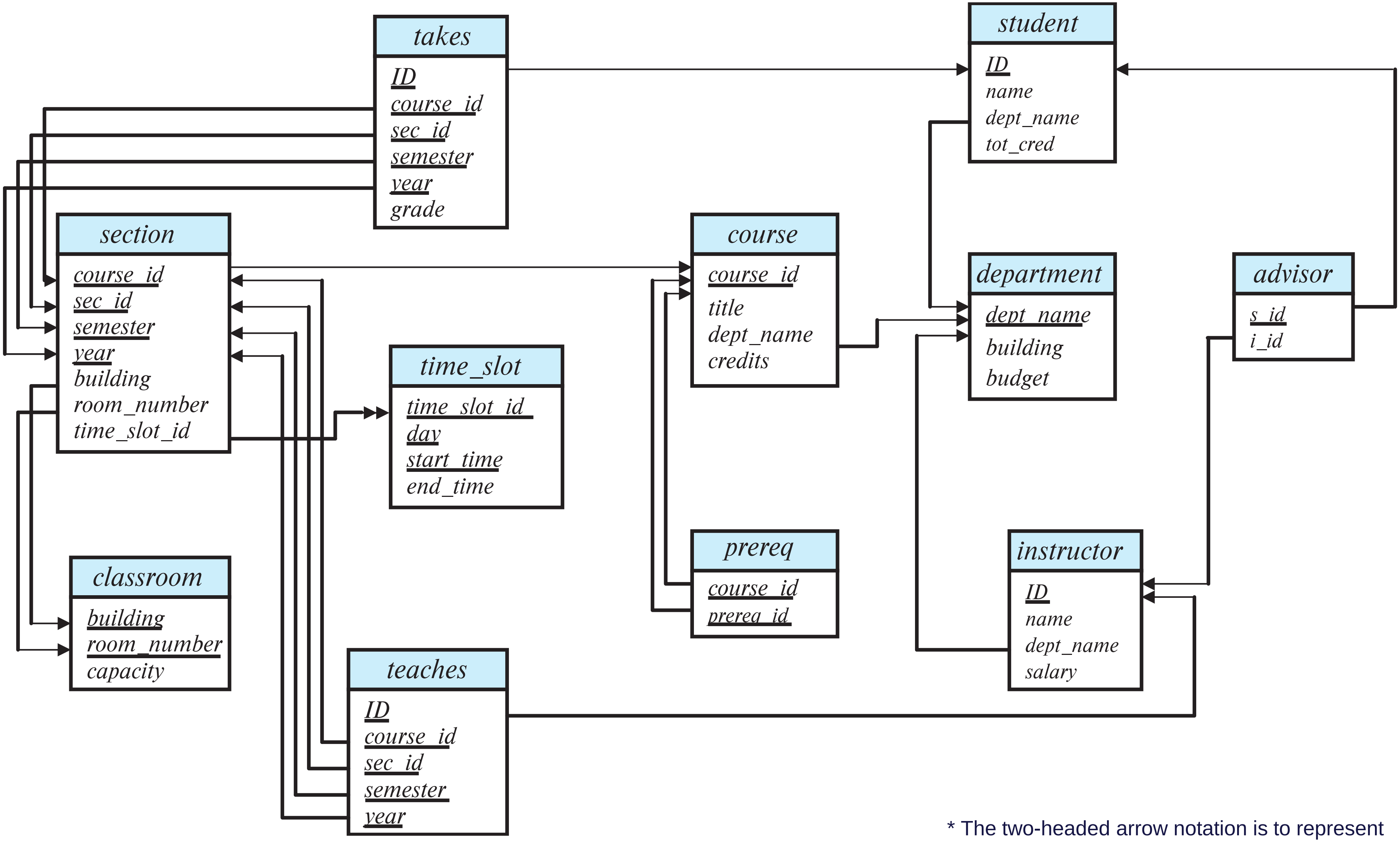
instructor : stores information about instructors

- Why not store all information as a single relation?
- How to break up these information reasonably?

Relational Database

- Key (键)
  - Superkey (超键)
    - Set of attributes
    - Values for which are sufficient to identify a unique tuple of each possible relation $r(R)$
  - Candidate key (候选键)
    - Is a superkey, and it is
    - Minimal (any part of which is not a superkey)
  - Primary key (主键)
    - A selected candidate key
  - Foreign key (外键)
    - Set of attributes, which are primary keys in some other relation(s)

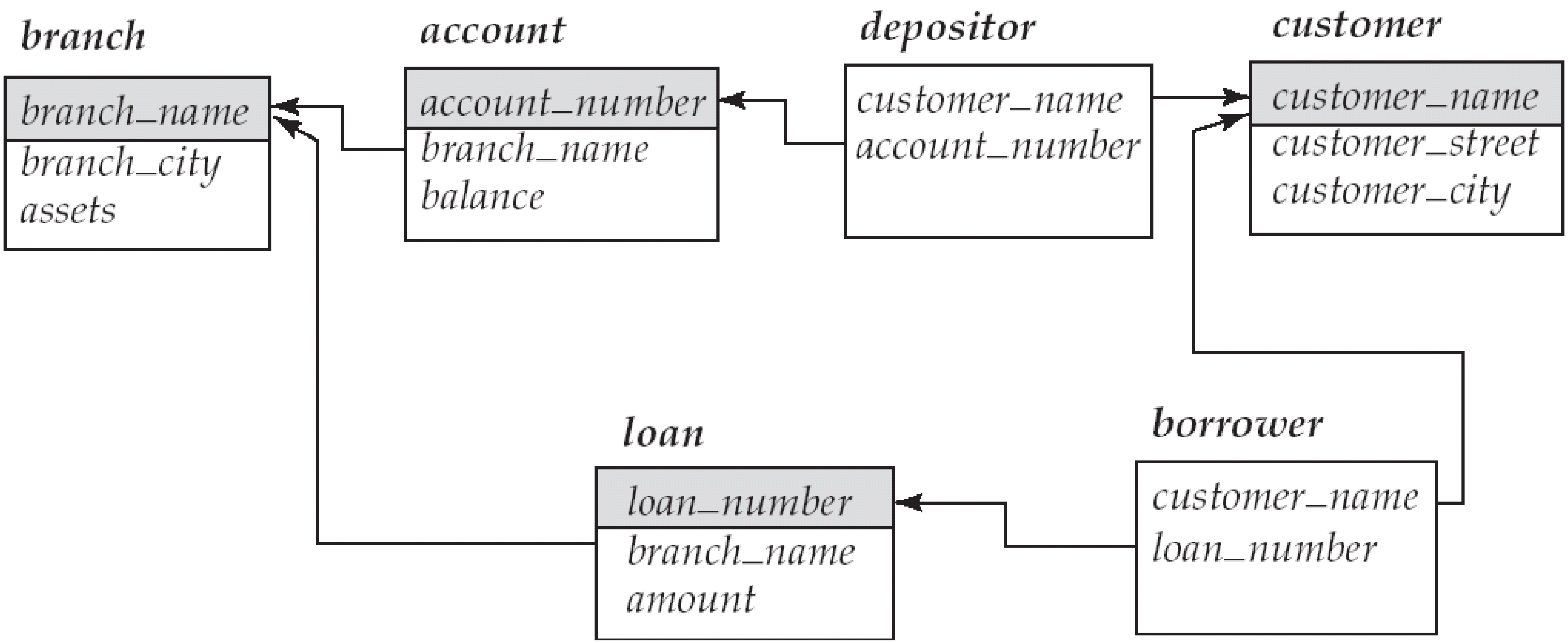
Schema Diagram



* The two-headed arrow notation is to represent a special type of referential integrity.

Schema Diagram

Another Example:



Relational Database

- Basic concepts. * SUMMARY *
  - 关系模式——属性序列
  - 关系——建立在关系模式上
    - 其值表示为集合形式，也可以表示为“表”
  - 关系实例——关系的当前值
  - 表——一般指关系的当前值，即表示关系实例
  - 属性——在关系(模式)上讨论，也可以在表上讨论
    - 在表上讨论时，可称作“列”
  - 元组——关系的（某个）元素；关系模式上并没有元组，但元组必定基于某个关系模式
    - 在表上讨论时，可称作“行”
  - (关系)数据库——一组关系，实现为一组表
  - 键

Relational Algebra

- How to make use of a database?
  - Language in which user requests information from the database
  - Query language does more than “querying” data.

- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- These three pure languages are equivalent in computing power
  - form up a basis of query languages that people use

Relational Algebra

- Procedural language
- Six basic operators
  - 选择 select: σ
  - 投影 project: Π
  - 并 union: $\cup$
  - 差集 set difference: –
  - 笛卡尔积 Cartesian product: x
  - 重命名 rename: ρ
- Inputs: one or two relations (incl. constant relation)
- Outputs: a new relation

选择 (select)

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

选择 (select)

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

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

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>

<attribute> *op* <constant>

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

- Example of selection:

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

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

投影 (project)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\prod_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

投影 (project)

- Notation:

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

  where A_1, A_2 are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *salary* and *id* attributes of *instructor*

$$\Pi_{name,\ dept\_name}\ (instructor)$$

并 (union)

- 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

并 (union)

- 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**

差集 (set difference)

- 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
β	1

差集 (set difference)

- 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

笛卡尔积 (Cartesian product)

- Relations *r, s*:

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*

笛卡尔积 (Cartesian 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 attributes of *r(R)* and *s(S*) are not disjoint, then **renaming** must be used.

笛卡尔积 (Cartesian product)

- Relations *r, s*:

r

A	*B*
α	1
β	2

s

A	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

- *r* x *s*:

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

- Allows us to refer to a relation by more than one name
  - 可以通过不同的名字引用同一个关系
- Example:

$$\rho_{X}(E)$$

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

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

$$\rho_{x(A_1, A_2, \ldots, A_n)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to A_1, A_2, …., A_n .

Renaming

- Relation *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

Composite Operation

- 使用多个运算构造表达式
- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$

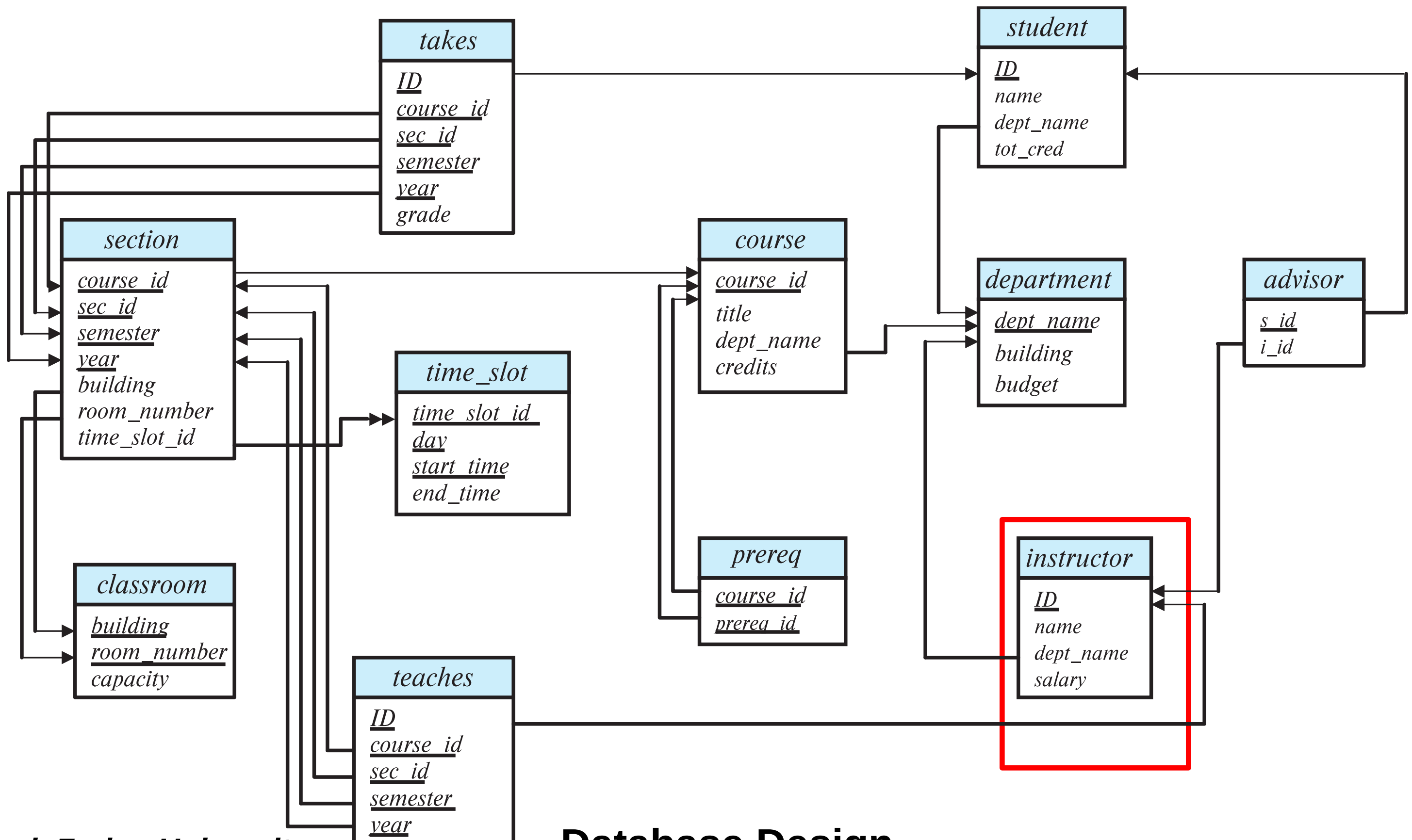
A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

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

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

Example Queries - *University*

- Select and Project
  - Find instructors with salary greater than $85,000
  - Find the ID and salary of the instructors with salary greater than $85,000

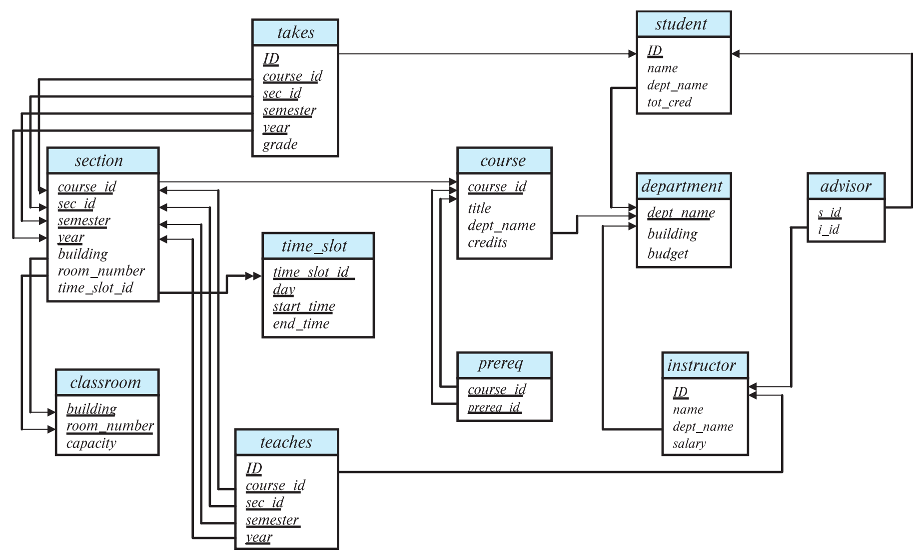


Example Queries - *University*

- Select and Project
  - Find instructors with salary greater than $85,000
  - Find the ID and salary of the instructors with salary greater than $85,000

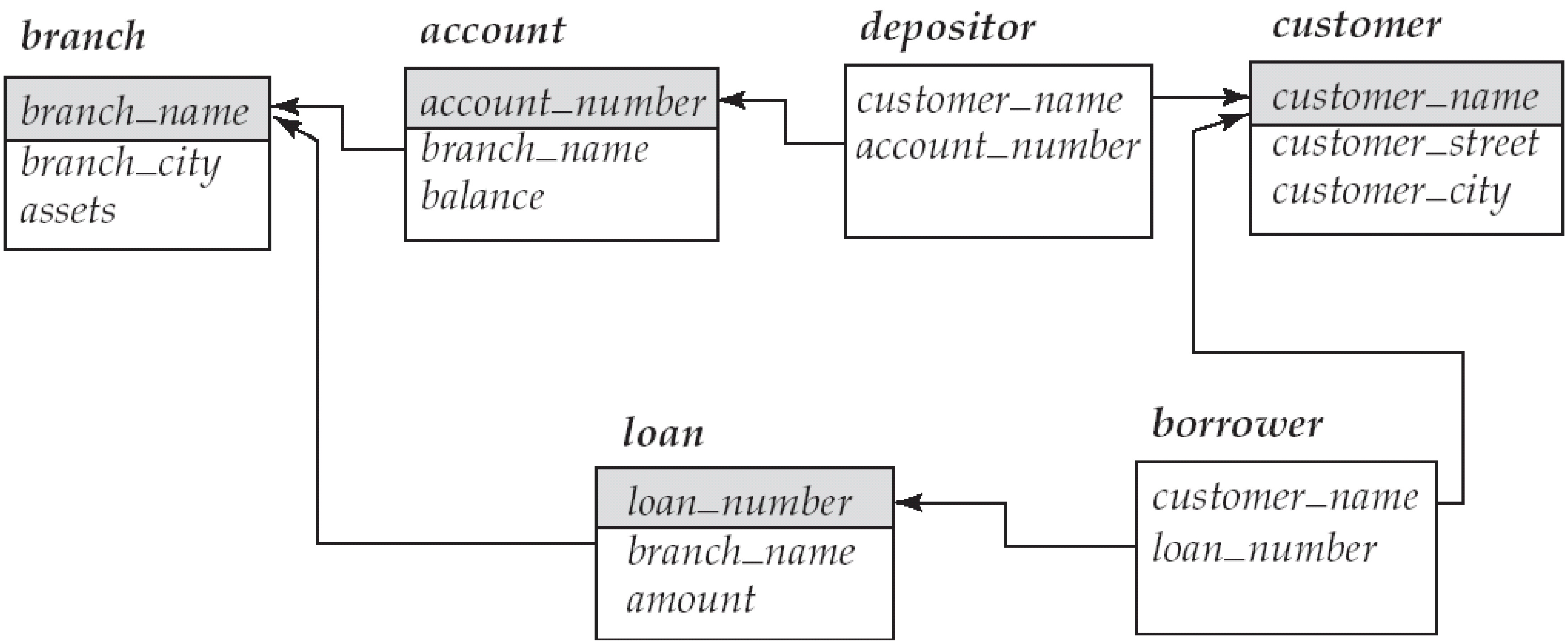
$$\sigma_{salary > 85000}\ (instructor)$$

$$\Pi_{id,\ salary}\ (\sigma_{salary > 85000}\ (instructor))$$



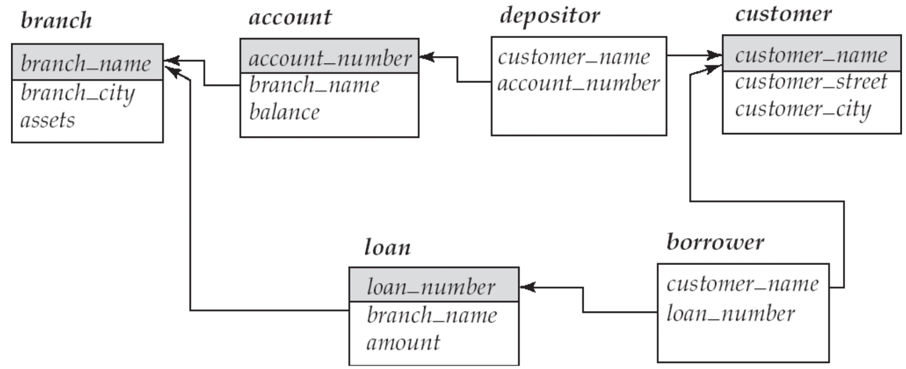
Banking Example

Schema diagram:



Example Queries

- Union and Set-Difference
  - 拥有贷款或者存款的客户名
  - 拥有存款但没有贷款的客户名
  - 拥有存款和贷款的客户名

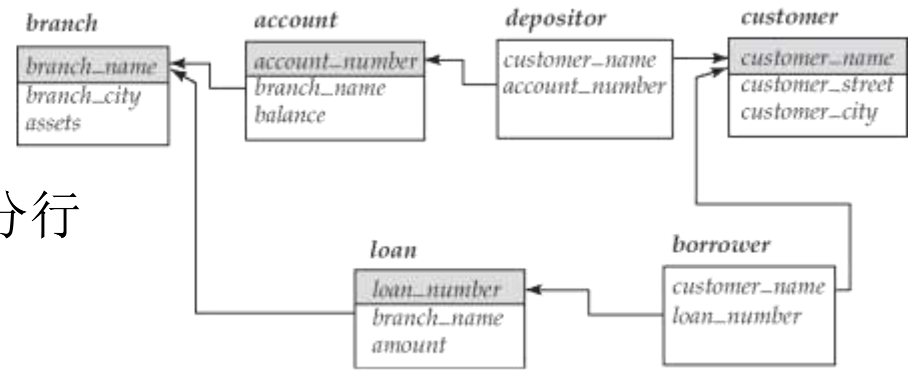


$$\Pi_{customer\_name}(borrower) \cup \Pi_{customer\_name}(depositor)$$

$$\Pi_{customer\_name}(depositor) - \Pi_{customer\_name}(borrower)$$

$$\Pi_{customer\_name}(depositor) - (\Pi_{customer\_name}(depositor) - \Pi_{customer\_name}(borrower))$$

Example Queries



- Cartesian-Product
  - 在Perryridge分行有贷款的客户名
  - 在Perryridge分行有贷款且在任何分行都没有存款的客户名

$$\Pi_{customer\_name}\,(\sigma_{branch\_name=\text{“Perryridge”}}\,(\sigma_{borrower.loan\_number\,=\,loan.loan\_number}(borrower \times loan)))$$

$$\Pi_{customer\_name}\,(\sigma_{branch\_name\,=\,\text{“Perryridge”}}\,(\sigma_{borrower.loan\_number\,=\,loan.loan\_number}(borrower \times loan))) \;-\; \Pi_{customer\_name}(depositor)$$

Example Queries

- 关系运算的组合顺序
  - 在Perryridge分行有贷款的客户名
    - 先算borrower x loan

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"}Perryridge\text{"}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \ x \ loan)))$$

    - 先对loan进行branch_name="Perryridge"选择

$$\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}((\sigma_{branch\_name = \text{"}Perryridge\text{"}}(loan)) \ x \ borrower))$$

Basic Operators - Summary

- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p(E_1)$, P is a predicate on attributes in E_1
  - $\prod_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x(E_1)$, x is the new name for the result of E_1

Additional Operations

- 交集 (Intersection)
- 自然联结 (Natural-join)
- 除 (Division)
- 赋值 (Assignment)

交集 (intersection)

- 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

交集 (intersection)

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r$ **and** $t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

自然联结（natural-join）

- 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	c

r

B	*D*	*E*
1	a	α
3	a	β
1	a	γ
2	c	δ
3	b	$\in$

s

r ⋈ s

A	*B*	*C*	*D*	*E*
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	c	δ

自然联结（natural-join）

- Let *r* and *s* be relations on schemas *R* and *S* respectively.
  Then, natural-join r⋈s is a relation on schema $R \cup S$ ("union") 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 $R \cap S$ ("intersection"), add a tuple *t* to the result, where
    - *t* has the same value as t_r on *r*
    - *t* has the same value as t_s on *s*
- Example:
  $R = (A, B, C, D)$
  $S = (E, B, D)$
  - Result schema = (A, B, C, D, E)
  - $r \bowtie s$ is defined as:

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

Theta Join

- The **join** operation allows us to combine a ***select*** operation and a ***Cartesian-Product*** operation into a single operation.
- The Theta Join operation is a generalized form of natural join.
- 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

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

Can equivalently be written as

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

- In this case, above query is equivalent to *instructor* $\bowtie$ *teaches*

除 (division)

Relations *r, s*:

A	*B*
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ε	6
ε	1
β	2

r

B
1
2

s

r ÷ s:

A
α
β

除 (division)

- Let r and s be relations on schemas R and S respectively where

  $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$

  $S = (B_1, \ldots, B_n)$

  The result of $r \div s$ is a relation on schema

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

  $$r \div s = \{ t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s \, (tu \in r) \}$$

  Where tu means the concatenation of tuples t and u to produce a single tuple

除 (division)

Relations r, s:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

r ÷ s:

A	*B*	*C*
α	a	γ
γ	a	γ

除 (division)

- Property
  - $q = r \div s$
  - Then q is the largest relation satisfying $q \times s \subseteq r$
- 用关系代数基本运算表示除

Let $r(R)$ and $s(S)$ be relations, and let $S \subseteq R$

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

r

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

s

D	E
a	1
b	1

赋值 (assignment)

- Notation $\leftarrow$
  - 使用临时变量保存关系运算的结果
  - 简化关系表达式的表示
- $r \div s$ may be expressed as:

$$temp1 \leftarrow \Pi_{R-S}(r)$$

$$temp2 \leftarrow \Pi_{R-S}((temp1 \times s) - \Pi_{R-S,S}(r))$$

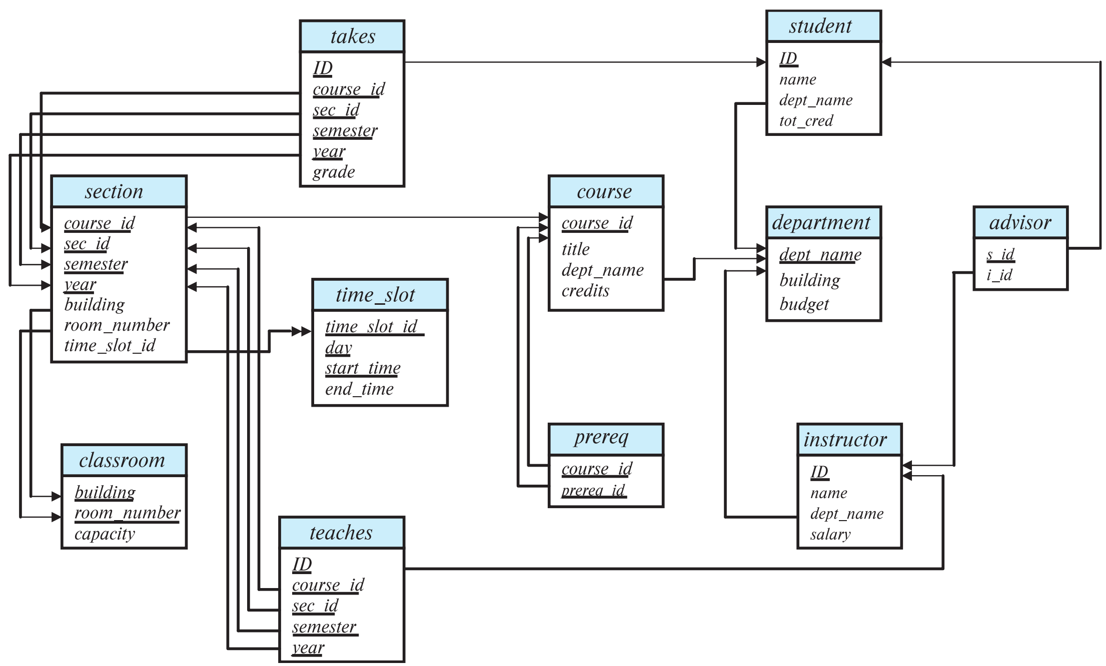
$$r \div s = temp1 - temp2$$

  - The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$
  - May use variable in subsequent expressions

Example Queries

- Set Intersection
  - Find the ids of the courses taught in Fall 2009 as well as in Spring 2010

$$\Pi_{course\_id} (\sigma_{semester = \text{'Fall'} \wedge year = 2009} (section))$$
$$\cap \quad \Pi_{course\_id} (\sigma_{semester = \text{'Spring'} \wedge year = 2010} (section))$$



Example Queries

- Natural-join
  - Find the names of instructors and the corresponding ids of the courses taught by the instructors

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

  - Find the names of instructors and the corresponding ids of the courses taught by the instructors, and show the year in which the courses were taught

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

Example Queries

- Division
  - Find the names of instructors and the corresponding ids of the courses taught by the instructors in both 2010 and 2011

$$\Pi_{name,\ course\_id,\ year}\ (instrcutor \bowtie teaches)$$
$$\div\ \rho_{temp(year)}\ (\{(2010),\ (2011)\})$$

- 一般的查询方法

$$\Pi_{name,\ course\_id}\ (\sigma_{year\ =\ 2010}\ (instrcutor \bowtie teaches))$$
$$\cap\ \Pi_{name,\ course\_id}\ (\sigma_{year\ =\ 2011}\ (instrcutor \bowtie teaches))$$

  - What does the following query mean?

$$\Pi_{name,\ course\_id,\ year}\ (instrcutor \bowtie teaches)$$
$$\div\ \Pi_{year}\ (\sigma_{semester\ =\ \text{"Fall"}}\ (section))$$

Review

- 基本的关系代数运算
  - 六种基本运算符，其中：
  - 一元运算符和二元运算符各三种
- 附加的关系代数运算
  - 能够由基本关系代数运算表达
  - 丰富和简化了关系代数的表达方式

Extended Operations

- 泛化投影 (Generalized Projection)
- 聚合函数 (Aggregate Functions)
- 外联结 (Outer Join)

Generalized Projection

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

E is any relational-algebra expression

Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of E

- Extended project operation
  - allowing arithmetic functions to be used in the projection list
- Example
  - Relation ***credit_info(customer_name, limit, credit_balance)***

    Find how much more each person can spend :

    $\Pi_{\textit{customer\_name, limit – credit\_balance}}$ *(credit_info)*

聚合 (Aggregation)

- **Aggregation function (聚合函数)**： takes a collection of values and returns a single value as a result

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

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

E is any relational-algebra expression

– $G_1, G_2 \ldots, G_n$ 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

Also use the symbol γ： $_{G_1, G_2, \ldots, G_3}\gamma_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(E)$

Aggregation Examples

- Relation *r*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

$\mathcal{g}_{\textbf{sum(C)}}\textbf{(r)}$

sum(C)
27

Aggregation Examples

- Relation *instructor* grouped by *dept_name*:

name	*dept_name*	*salary*
Einstein	Physics	95000
Gold	Physics	87000
Wu	Finance	90000
Singh	Finance	80000
Mozart	Music	40000

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

dept_name	**sum**(*salary*)
Physics	182000
Finance	170000
Music	40000

聚合 (Aggregation)

- Renaming the result of aggregation
  - Result of aggregation does not have a name
  - For convenience, we permit renaming as part of aggregate operation

$$_{dept\_name}\mathcal{G}_{\textbf{sum}(salary)\ \textbf{as}\ sum\_salary}(instructor)$$

dept_name	*sum_salary*
Physics	182000
Finance	170000
Music	40000

Outer Join

- Relation *course*

course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Inner Join (Natural Join)
  course ⋈ *prereq*

course_id	*title*	*dept_name*	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp.Sci.	4	CS-101

Outer Join

- Relation *course*

course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Left Outer Join
  course ⟕ *prereq*

course_id	*title*	*dept_name*	*credits*	*prere_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*

Outer Join

- Relation *course*

course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Right Outer Join
  course ⟖ *prereq*

course_id	*title*	*dept_name*	*credits*	*prere_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

Outer Join

- Relation *course*

course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Full Outer Join
  course ⟗ *prereq*

course_id	*title*	*dept_name*	*credits*	*prere_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

Null Values (空值)

- Denoted by *null*
- *null* signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null.*
- Aggregate functions simply ignore null values (as in SQL)
- For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

Null Values (空值)

- Three-valued logic using the truth value *unknown*:
  - OR: (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown*) = *unknown*
  - AND: (*true* **and** *unknown*) = *unknown,*
    (*false* **and** *unknown*) = *false,*
    (*unknown* **and** *unknown*) = *unknown*
  - NOT*:* (**not** *unknown*) = *unknown*
- Result of select predicate is treated as *false* if it evaluates to *unknown*

Modification of the Database

- The content of the database may be modified using the following operations
  - 添加(insertion)
  - 删除(deletion)
  - 修改(updating)
- Expressed using the assignment operator

添加 (insertion)

$r \leftarrow r \cup E$

where r is a relation and E is a relational algebra expression

- To insert data into a relation, we either:
  - specify a tuple to be inserted (E is a constant relation, containing only one tuple)
  - write a query whose result is a set of tuples to be inserted
- 例

$instructor \leftarrow instructor \cup$ {(54321, “Mickey”, Music, 45000)}

删除 (deletion)

$r \leftarrow r - E$

where r is a relation and E is a relational algebra query

- Selected tuples are removed from the database.
  - similarly to a query
  - can delete only whole tuples
  - cannot delete values on only particular attributes
- 例

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

$$student \leftarrow student - \sigma_{tot\_cred \geq 1 \land tot\_cred \leq 5}(student)$$

删除 (deletion)

- 例
  - 删除位于Needham市的分行的所有存款帐户

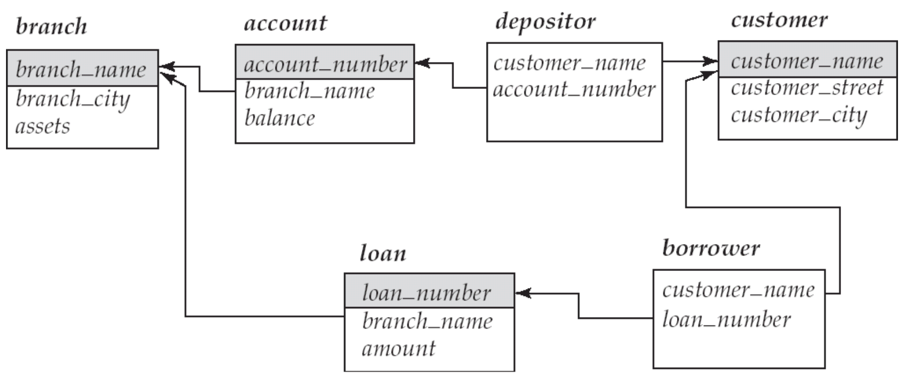
$$r_1 \leftarrow \sigma_{branch\_city = \text{“Needham”}} (account \bowtie branch)$$

$$r_2 \leftarrow \Pi_{account\_number,\ branch\_name,\ balance} (r_1)$$

$$r_3 \leftarrow \Pi_{customer\_name,\ account\_number} (r_2 \bowtie \text{depositor})$$

$$account \leftarrow account - r_2$$

$$depositor \leftarrow depositor - r_3$$



更新 (Updating)

$$r \leftarrow \prod_{F_1, F_2, \ldots, F_l}(r)$$

其中*r*是关系，

F_i是

- *r* 的第 *i* 个属性（如果第 *i* 个属性不需要修改）； 或
- 仅包含常量和*r*中属性的算术表达式，根据此值更新 *r* 的第 *i* 个属性

Each F_i is either

- **the i th attribute of *r*, if the i th attribute is not updated, or,**
- **if the attribute is to be updated, an expression, involving only constants and the attributes of *r*, which gives the new value for the attribute**

- Change a value in a tuple without changing *all* values in the tuple

更新 (Updating)

- 例
  - 向存款帐户发放利息，金额为存款帐户余额的5%

$$account \leftarrow \Pi_{account\_number,\ branch\_name,\ balance\ *\ 1.05}\ (account)$$

  - 对余额超过10000元的存款帐户支付6%的红利，其他存款帐户支付5%的红利

$$account \leftarrow$$
$$\Pi_{account\_number,\ branch\_name,\ balance*1.06}\ (\sigma_{balance\ >\ 10000}\ (account\ ))$$
$$\cup\ \ \Pi_{account\_number,\ branch\_name,\ balance*1.05}\ (\sigma_{balance\ \leq\ 10000}\ (account))$$

小结

- 什么是关系数据库？
- 关系代数的六个基本操作以及其他的关系运算
- 采用关系代数表达式表示数据库查询和修改操作
  - 简单查询
  - 组合查询
  - 增、删、改
- 外联结在数据库查询中的作用

Next Lecture

- SQL
  - 从纯语言到应用
  - 在实践中学习各类查询语言的用法

End of
Lecture 2