关系数据库的基本特征是使用关系模型的组织数据，20世纪80年代以后，在商用DBMS中，关系模型逐步取代早期的网状模型和层次模型。

作为数据模型，关系模型包含三个组成要素：

关系数据结构、关系操作集合和关系完整性约束。

1.关系数据结构

结构只包含单一的数据结构（关系），现实世界的实体与实体间的各种联系均用关系来表示。关系模型是吧数据库表示为关系的集合，并以二维表格的形式组织数据。

基本术语

1. **表(Table)：**也称为关系，是二维数据结构，由表名、构成表的各列及若干行数据组成，每个表由唯一的表名，每一行数据描述一条具体的记录值。
2. **关系（Relation）：**一个关系**逻辑上对应一张二维表**，可以为每个关系取一个名称来标识。关系有三种类型：基本关系（基表，实际存在的表，是实际存储数据的逻辑表示）、查询表（查询结构对应的表）和视图表（由基本表或其他视图导出的表，不对应实际存储的数据）。
3. **关系模式（Relation Schema）：**通数据模型一样，数据库也有型和值，在关系数据库中关系模式是型，关系是值，关系模式是对关系的描述。

   关系数据库中，关系模式是型，关系是值。**关系模式是对关系的描述。**关系的描述称为关系模式（relation schema），它可以形式化地表示为R（U，D，DOM，F），其中R为关系名，U为组成该关系的属性名集合，D为 U中属性所来自的域，DOM为属性向域的映像集合，F为属性间数据的依赖关系集合。

   关系是关系模式在某一时刻的状态或内容。关系模式是静态的、稳定的，而关系是动态的、随时间不断变化的。
   关系模式是静态的、稳定的，而关系是动态的、随时间不断变化，因为关系操作在不断地更新着数据库的数据实际工作中关系模式和关系统称为关系。
4. **关系数据库（Relation Database）**：**所有关系的集合**

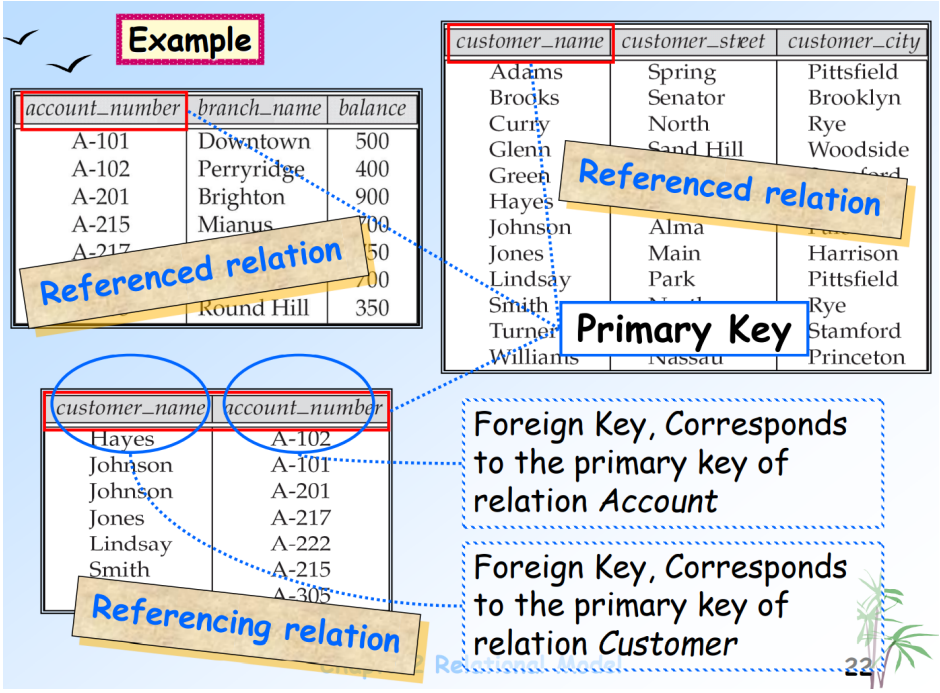
   以关系模型作为数据的逻辑模型，并采用关系作为数据组织方式的一类数据库，其数据库操作建立在关系代数的基础上。在给定的应用领域中，所以关系的集合构成一个关系数据库。

   关系数据库对关系的限定：

   - **每个属性都不可分解**，是关系数据库对关系的最基本的限定，要求关系的每个分量必须是一个不可分的数据项，即不允许表中有表
   - **一个关系对应一种关系模式**，模式中的属性的数据类型及属性的个数是相对固定的
   - 每个关系模式中的属性必须命名，**在同一模式中，属性名必须是不同的**
   - 同一关系中不允许出现候选码或键值完全相同的元组
   - 关系中的元组顺序是可任意交换
   - 关系中的属性顺序可以任意交换
5. **列（Column）：**称为字段（Field）或属性（Attribute）。每一列有一个名称，表示实体属性，具有相同数据类型。在一个数据库中，表名，字段名必须唯一，不同的表可以有相同的字段名，且命名须有意义，简单。
6. **属性（Attribute）**：表列即属性，给属性起名即属性名。属性的个数称为关系的元或度。列值为属性值；取值范围为值域。
7. **行（Row）：**称为元组（Tuple）或记录（Record）。表中的数据按行存储，一行数据即一条记录或元组，每行又若干个字段值组成。

8. 分量（Component）：元组的属性值为分量

9. 超码或超键（Super Key）：能用来标识该关系的元组，则为该关系的码或键，每个关系至少有一个默认的超码（所有属性的集合）。
10. 候选码或键（Candidate Key）：关系中的一个码或键中，**不能去除任何一个属性**，否则它就不是对应关系表的超码或键，则此码为候选码（键），它是**关系表中最小的超码或超键。**
11. 主键/码(Primary Key)：在一张关系表中的若干**候选键中指定一个用来唯一标识该关系的元组**，则该候选键为主键。
12. 全键/码（All-Key）：一个关系中所有的属性集合是是这个关系是主键/码，则为全键/码。
13. 主/非属性（Primary Attribute/Nonprimary Attribute）：关系中包含任何一个候选键/码的属性为主/码属性，不包含任何一个候选码的属性为非主/码属性。
14. 外键/码（Foreign Key）：关系中的某个属性或属性组不是这个关系的主键或候选键，而是另一个关系的主键，则该属性（属性组）为关系的外键/码。



15. 参照关系(Referencing Relation)/被参照关系（Referenced Relation）：参照关系也称从关系，被参照关系又称主关系，它们指以外码相关联的两个关系。而以外码为主码的关系为被参照关系；外码所在的关系为参照关系，这种联系通常是一对多关联。
16. Reference Constraint 参照约束：只有出现在被参照关系主码中的属性才能出现在参照发、关系的外码中
17. 域（Domain）：指属性取值范围。
18. **数据类型（Data Type）：**每列（元关系）都有相应的数据类型，用于限制该列中存储的数据。
19. 视图：**视图是由几个基本表导出的表，但不存储在数据库中，数据库中只有视图的定义，而不存放对应的数据**

关系完整性约束

关系模型的完整性规则是对关系的某种约束条件。关系模型中有三类完整性约束：实体完整性（entity integrity）、参照完整性（referential integrity）和用户定义的完整性（user-defined integnty）。

1．实体完整性

（1）定义 若属性（指一个或一组属性）A是基本关系R的主属性，则A不能取空值（null value）。所谓空值就是“不知 道”或“不存在”或“无意义”的值。如果主码由若干属性组成，则所有这些主属性都不能取空值。

（2）规则说明

① 实体完整性规则是针对基本关系而言的。一个基本表通常对应现实世界的一个实体集。

② 现实世界中的实体是可区分的，即它们具有某种唯一性标识。

③ **以主码作为唯一性标识。**

④ **主属性不能取空值。**

2．参照完整性

（1）定义 设F是基本关系R的一个或一组属性，但不是关系R的码，Ks是基本关系 S的主码。如果F与Ks相对应，则称 F是R的外码（foreign key），并称基本关系R为参照关系（referencing relation），基本关系S为被参照关系 （referenced relation）或目标关系（target relation）。

（2）参照完整性规则 参照完整性规则就是定义外码与主码之间的引用规则。若属性（或属性组）F是基本关系R的外码，它与基 本关系S的主码Ks相对应（基本关系R和S不一定是不同的关系），则对于R中每个元组在F上的值必须取空值（F 的每个属性值均为空值），或者等于S中某个元组的主码值。

3．用户定义的完整性

用户定义的完整性是针对某一具体关系数据库的约束条件，它反映某一具体应用所涉及的数据必须满足的 语义要求。关系模型应提供定义和检验这类完整性的机制，以便用统一的系统的方法处理它们，而不需由应用 程序承担这一功能。

题

1．试述关系模型的3个组成部分。

答：关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。

（1）关系数据结构：在关系模型中，现实世界的实体以及实体间的各种联系均用单一的结构类型即关系来表示。

（2）关系操作集合：关系模型中常用的关系操作包括**查询操作**和**插入**、**删除**、**修改**操作。

（3）关系完整性约束：关系模型中有实体完整性约束、参照完整性约束和用户定义的完整性约束三类约束。

2. 关系操作集合

基本的关系操作

增（插入 Insert）、删(Delete)、改(Update)、查（Query）。关系的查询表达能力是关系操作最主要的部分。查又可分为选择、投影、连接、除、并、差、交、笛卡尔积。集合操作方式（操作的对象和结果都是集合）。又称为一次一集合（set-at-a-time）

五种最基本操作：

- ```
  1 并、差、选择、投影、乘积
  ```

关系数据语言的分类
  ```

通过关系语言实现关系操作。它高度非过程化用户不必请求DBM为其建立特殊的存取路径，由 DBMS 的优化机制来完成。

1. 代数方式：主要有关系代数，通过对关系的操作来表达查询要求
2. 逻辑方式：主要有关系演算，是用谓词来表达查询要求，关系演算又按谓词变元的基本对象（元组变量或域变量），分为元组关系演算和域关系演算。
3. 介于前两者之间的结构化查询语言（Structured Query Language,SQL）：SQL具有丰富的查询功能、数据定义和数据控制功能。集查询、数据定义语言（DDL）、数据操作语言（DML）和数据控制语言（Data Control Language, DCL）于一体；是关系数据库的标准语言。

关系代数(Relation Algebra)

关系代数是关系操作语言中的传统表示方式，以集合代数为基础发展而来。任何一种操作都是将一定的操作符作用域一定的操作对象上，得到预期的操作结果。
操作包含三大要素：操作对象、操作符、操作结果。而对象和结果均为关系。关系代数直接应用关系的运算来表达操作的目的，运算符包括集合运算符和专门的关系运算符。

运算符	含义
集合运算符	
∪	并 Union
–	差
∩	交 intersection
×	笛卡尔积
专门的关系运算符	
σ	选择 Select
π	投影 Project
⋈	自然连接
÷	除
比较操作符	
>	大于
≥	大于等于
<	小于
≤	小于等于
=	等于
≠	不等于
比较操作符	
¬	非
∧	与
∨	或

关系代数操作经过有限次复合的式子称为关系代数操作表达式（关系代数表达式），可使用表达式表示所需要执行的各种数据查询和修改处理，所有关系代数是一种抽象的查询语言，通过对关系的操作来表达查询。

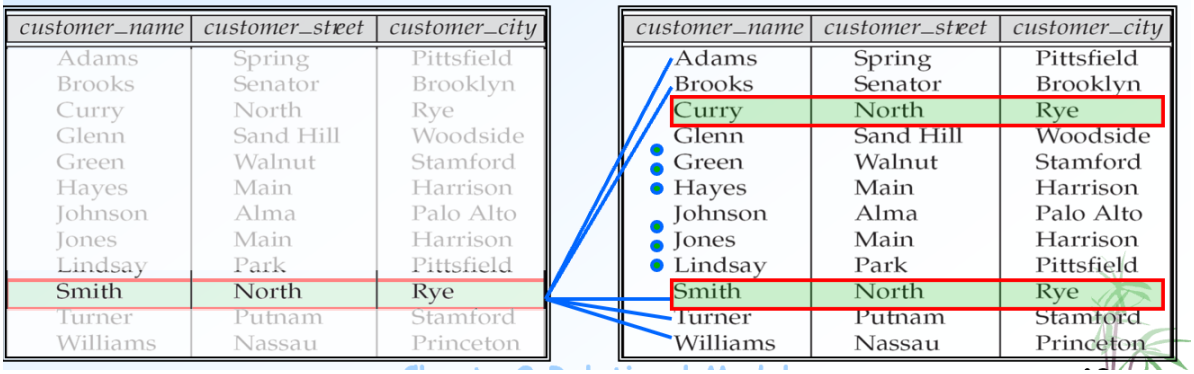
按运算符分类，关系代数操作可分为：传统的集合运算和专门的关系运算

PPT上的关系表达式例子

查找与史密斯居住在同一城市的所有客户的姓名

$$\Pi_{customer.customer\_name}(\sigma_{customer.customer\_city = smith\_info.customer\_city}(customer \times \rho_{smith\_infor}(\sigma_{customer\_name="smith"}(customer))))$$

customer_name	customer_street	customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

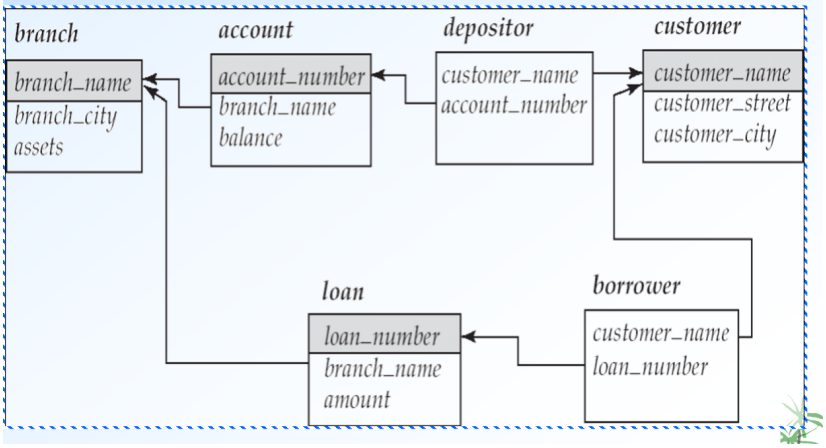


这里使用了重命名

银行的例子：

- branch (branch_name, branch_city, assets)
- customer (customer_name, customer_street, customer_city)
- account (account_number, branch_name, balance)
- loan (loan_number, branch_name, amount)
- depositor (customer_name, account_number)
- borrower (customer_name, loan_number)

Schema Diagram



Example Queries

- Find all loans of over $1200

$$\sigma_{amount > 1200}(loan)$$

- Find the loan number for each loan of an amount greater than $1200

$$\Pi_{loan\_number}(\sigma_{amount > 1200}(loan))$$

- Find the names of all customers who have a loan, an account, or both, from the bank

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

不写投影符号就是默认投影所有行

注意关系代数是自动去除重复的

- Find the names of all customers who have a loan at the Perryridge branch

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

- Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank

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

- Find the names of all customers who have a loan at the Perryridge branch

Query 1

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

Query 2

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

后面的是使用了Additional Operations

Additional Operations

注意自然连接的写法

- Find the names of all customers who have a loan and an account at bank

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

- Find the name of all customers who have a loan at the bank and the loan amount

$$\Pi_{customer\_name,\ loan\_number,\ amount}(borrower \bowtie loan)$$

两个都有用交

实体集和关系集查询用自然连接

- Find all customers who have an account from at least the “Downtown” and the “Uptown” branches.
  - Query 1

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Downtown”}}(depositor \bowtie account)) \cap \Pi_{customer\_name}(\sigma_{branch\_name = \text{“Uptown”}}(depositor \bowtie account))$$

  - Query 2

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \rho_{temp(branch\_name)}(\{(\text{“Downtown”}), (\text{“Uptown”})\})$$

注意除法里面的元素的写法

{ }表示列，（）表示列的属性，字符串需要加“ ”

查找好几个属性都有的，使用除法

- Find all customers who have an account at all branches located in Brooklyn city

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \Pi_{branch\_name}(\sigma_{branch\_city = \text{“Brooklyn”}}(branch))$$

找到有在好几个银行都存款的人，这个好几个银行就可以当作被除项

而这好几个银行也有限制条件，就是他们的city

Extends Operations

- **Aggregate operation** in relational algebra

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

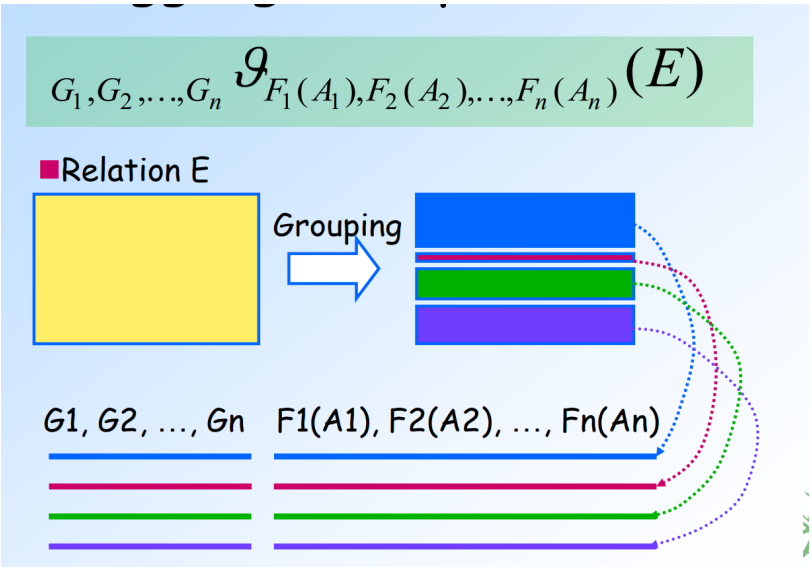
Example

r

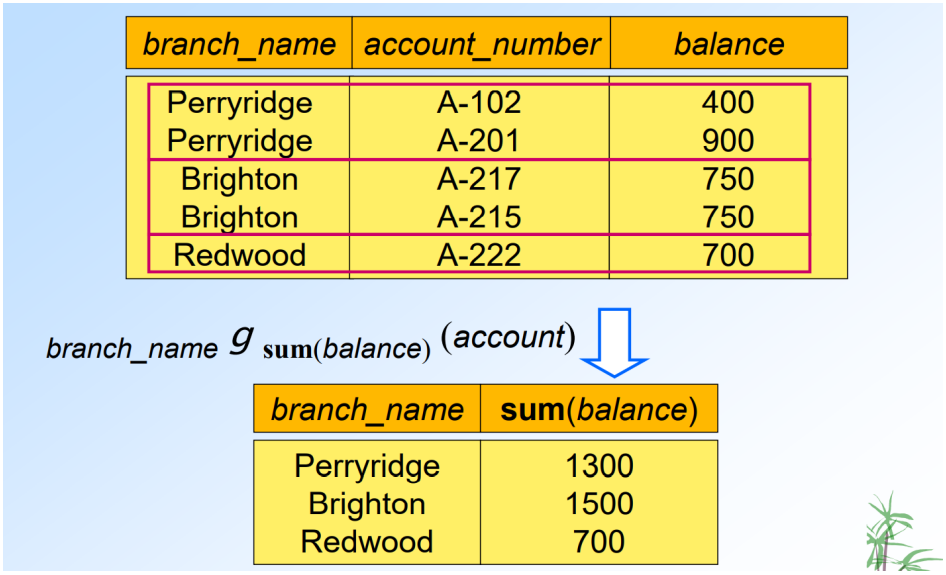
A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

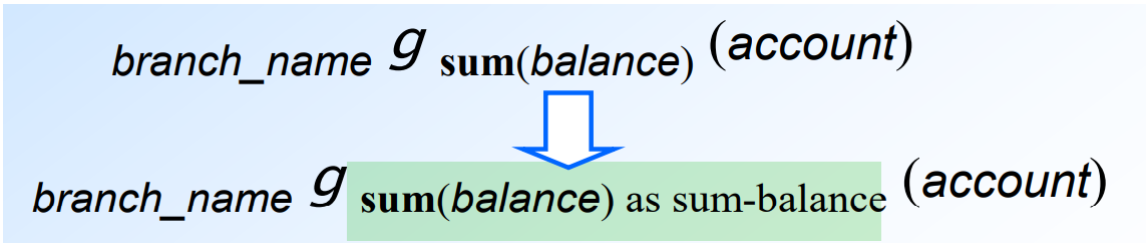
$g_{sum(c)}(r)$

sum(c)
27



分组使用g，前面是分组条件，后面是分组之后的操作（聚集函数）





使用as重命名

外连接略

Null Values

对于元组的某些属性，元组可能有一个空值，用 null 表示

null 表示未知值或值不存在。

任何涉及 null 的算术表达式的结果都是 null

聚合函数只是忽略 null 值（如在 SQL）

- 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*
  - In SQL "*P* **is unknown**" evaluates to true if predicate *P* evaluates to *unknown*

删除的时候就不要加投影了

- Delete all account records in the Perryridge branch

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}}(account)$$

- Delete all loan records with amount in the range of 0 to 50

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$$

在选择操作里用and、or连接不同的条件

- Delete all accounts at branches located in Needham.

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

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

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

$$account \leftarrow account - r_2$$

$$depositor \leftarrow depositor - r_3$$

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch

$$account \leftarrow account \cup \{(\text{“A-973”}, \text{“Perryridge”}, 1200)\}$$

$$depositor \leftarrow depositor \cup \{(\text{“Smith”}, \text{“A-973”})\}$$

- Provide as a gift for all loan customers in the Perryridge branch, a $200 savings account. Let the loan number serve as the account number for the new savings account.

$$r_1 \leftarrow (\sigma_{branch\_name = \text{“Perryridge”}}(borrower \bowtie loan))$$

$$account \leftarrow account \cup \prod_{loan\_number,\ branch\_name,\ 200}(r_1)$$

$$depositor \leftarrow depositor \cup \prod_{customer\_name,\ loan\_number}(r_1)$$

- Pay all accounts 5 percent interest

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

- Pay all accounts with balances over $10,000 6 percent interest and pay all others 5 percent

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

题

2 在关系代数运算中，五种基本运算为***A***

A. 并、差、选择、投影、乘积

B. 并、差、选择、投影、自然连接

C. 并、差、交、选择、投影

D. 并、差、交、选择、乘积

- 注意区分投影结果之间的运算符和选择时的运算符

5 如下图所示，查询既学习课程号为001课程又学习课程号为002号课程的学生的学号,正确的是_____

关系 Student

S#	Sname	Ssex	Sage	D#	Sclass
980301	张三	男	20	03	9803
980401	李四	男	18	04	9804
980402	王五	男	21	04	9804

关系 SC

S#	C#	Score
980301	001	92.0
980301	002	85.0
980401	003	88.0
980402	002	84.5

A. $\pi_{S\#}(\sigma_{C\#="001"}(SC)) \cap \pi_{S\#}(\sigma_{C\#="002"}(SC))$

B. $\pi_{S\#}(\sigma_{C\#="001" \cap C\#="002"}(SC))$

C. $\pi_{S\#}(\sigma_{C\#="001"}(SC)) \wedge \pi_{S\#}(\sigma_{C\#="002"}(SC))$

D. $\pi_{S\#}(\sigma_{C\#="001" \wedge C\#="002"}(SC))$

外连接，看图就能看懂。

对于如下运算,结果是_____。

teacher

T#	Tname	Salary
001	赵三	1200.00
002	赵四	1400.00
003	赵五	1000.00
004	赵六	1100.00

Teach

T#	C#
001	001
002	002
004	002

Course

C#	Cname
001	数学
002	物理
003	化学

T#	Tname	Salary	C#	Cname
001	赵三	1200.00	001	数学
002	赵四	1400.00	002	物理
004	赵六	1100.00	002	物理
null	null	null	003	化学