

扩展关系代数运算

- 广义投影
- 聚集函数
- 外连接

- 允许在投影列表中使用算术函数来对投影操作进行扩展，广义投影运算形式为：

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

- E是任意关系代数表达式
- F_1, F_2, \dots, F_n 是涉及E模式中常量和属性的算术表达式
- 给出关系 *credit-info(customer-name, limit, credit-balance)*，找出每个客户还能花费多少，可以表述为：

$$\Pi_{customer-name, \textcolor{brown}{limit} - \textcolor{brown}{credit-balance}}(\textcolor{brown}{credit-info})$$

聚集函数

□ 聚合函数输入一个值集合，然后返回单一值作为结果

- avg: 平均值
- min: 最小值
- max: 最大值
- sum: 值的总和
- count: 值的数量

$g_{avg(balance)}(account)$
(求平均存款余额)

□ 聚集函数的关系代数表示:

$$G_1, G_2, \dots, G_n \quad g_{F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(E)$$

- E是任意关系表达式
- G_1, G_2, \dots, G_n 是用于分组的一系列属性
- F_i 是聚集函数
- A_i 是属性名

聚集函数

□ 例，关系r

A	B	C
---	---	---

α	α	7
α	β	7
β	β	8
β	α	14

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

avg-c
9

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

A	sum-c
α	14
β	22

$$B g_{avg(c)}(r)$$

B	avg-c
α	10.5
β	7.5

按branch-name将关系account分组

branch-name	account-number	balance
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

branch-name g $sum(balance)$ ($account$)

sum-balance

branch-name	
Perryridge	1300
Brighton	1500
Redwood	700

□ 聚集运算的结果是没有名称的

- 可以使用更名运算为其命名

- 可以把重命名作为聚集运算的一部分，如：

branch-name *g* *sum(balance) as sum-balance* (*account*)

- 外连接运算是连接运算的扩展，可以处理缺失信息
- 保留一侧关系中所有与另一侧关系的任意元组都不匹配的元组，再把产生的元组加到自然连接的结构上
- 使用空值：
 - 空值表示值不知道或不存在

外连接

loan_number	branch_name	amount
L_170	Downtown	3000
L_230	Redwood	4000
L_260	Perryridge	1700

customer_name	loan_number
Jones	L_170
Smith	L_230
Hayes	L_155

假设由于
某种原因
造成帐目
不符

loan

borrower

内连接

loan ⋈ *borrower*

loan_number	branch_name	amount	customer_name
L_170	Downtown	3000	Jones
L_230	Redwood	4000	Smith

左外连接

loan ⋈_L *borrower*

loan_number	branch_name	amount	customer_name
L_170	Downtown	3000	Jones
L_230	Redwood	4000	Smith
L_260	Perryridge	1700	null

外连接

右外连接

loan ⋈_r *borrower*

loan_number	branch_name	amount	customer_name
L_170	Downtown	3000	Jones
L_230	Redwood	4000	Smith
L_155	null	null	Hayes

全外连接

loan ⋈_{all} *borrower*

loan_number	branch_name	amount	customer_name
L_170	Downtown	3000	Jones
L_230	Redwood	4000	Smith
L_260	Perryridge	1700	null
L_155	null	null	Hayes

- ❑ 元组的某些属性值是可以为空的
- ❑ null表示未知值或值不存在
- ❑ 涉及空的任何算术表达式的结果为空
- ❑ 聚集函数会忽略空值
 - 可以返回空值作为结果
 - 我们遵循SQL对空值的处理语义
- ❑ 为了消除重复和分组，空值和其他值同等对待
 - 一种方法是两个空值被认为是相同的
 - 另一种方法是假设每个空值都是不同的
 - 这两种方法都可行，但我们更愿意遵循SQL对空值的处理语义

- ❑ 与空值的比较将返回一个特殊值：unknown
- ❑ 如果用false代替unknown，那么not (A<5) 与 A>=5 的结果就会不相等
- ❑ 使用特殊值unknown的三值逻辑：
 - 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

- ❑ 在SQL中，如果谓词P的值为unknown，那么“ P is unknown”的值为真
- ❑ 如果选择谓词的值未知，那么选择谓词的结果被认为false

数据库的修改

- 数据库的内容可以使用下面的操作来修改：
 - 删除
 - 插入
 - 更新
- 所有这些操作都使用赋值操作表示

删除

- ❑ 删除请求的表达与查询的表达非常相似，不同的是，前者不是要将找出的元组显示给用户，而是要将它们从数据库中去除
- ❑ 这样只能将元组整个地删除，而不能仅删除某些属性上的值
- ❑ 使用关系代数，删除可表达为：

$$r \leftarrow r - E$$

其中， r 是关系， E 是关系代数查询

删除示例

- 删除Perryridge分支机构的所有账户

$account \leftarrow account - \sigma_{branch-name = "Perryridge"} (account)$

- 删除贷款额在0到50之间的所有贷款

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

- 删除位于Needham的分支机构的所有账户

$r_1 \leftarrow \sigma_{branch-city = "Needham"} (account \bowtie branch)$

$r_2 \leftarrow \Pi_{branch-name, account-number, balance} (r_1)$

$r_3 \leftarrow \Pi_{customer-name, account-number} (r_2 \bowtie depositor)$

$account \leftarrow account - r_2$

$depositor \leftarrow depositor - r_3$

插入

- 为了将数据插入关系中：
 - 要么指明一个要插入的元组
 - 要么写出一个查询，其结果是要插入的元组集合
- 使用关系代数，插入可表达为：

$$r \leftarrow r \cup E$$

r 是关系， E 是关系代数表达式

- 如果让 E 是一个只包含元组的常量关系，就可以表达为向关系中插入单一元组

插入示例

- 向数据库中插入这样的信息：Smith在Perryridge分支机构的账户A-973上有\$1200

$$account \leftarrow account \cup \{(\text{"Perryridge"}, A-973, 1200)\}$$
$$depositor \leftarrow depositor \cup \{(\text{"Smith"}, A-973)\}$$

- 假设想对Perryridge分支机构的每一个贷款客户赠送一个新的\$200的存款账户，并将其贷款号码作为此账户的号码

$$r_1 \leftarrow (\sigma_{branch-name = \text{"Perryridge"}} (borrower \bowtie loan))$$
$$account \leftarrow account \cup \Pi_{branch-name, loan-number, 200} (r_1)$$
$$depositor \leftarrow depositor \cup \Pi_{customer-name, loan-number} (r_1)$$

- ❑ 某些情况下，可能只希望改变元组中的某个值，而不希望改变元组中的所有值
- ❑ 可以用广义投影运算来完成这个任务：

$$r \leftarrow \Pi_{F_1, F_2, \dots, F_l}(r)$$

- 其中，当第*i*个属性不被修改时， F_i 表示的是*r*的第*i*个属性
- 当第*i*个属性将被修改时， F_i 表示的是一个只涉及常量和*r*的属性的表达式，表达式给出了此属性的新值

更新示例

- 假设要付给所有账户5%的利息

$$account \leftarrow \Pi_{AN, BN, BAL * 1.05} (account)$$

- AN , BN 和 BAL 分别代表 $account-number$, $branch-name$ 和 $balance$

- 假设余额超过\$10 000以上的账户得到6%的利息，而其他账户得到5%的利息

$$account \leftarrow \Pi_{AN, BN, BAL * 1.06} (\sigma_{BAL > 10000} (account)) \\ \cup \Pi_{AN, BN, BAL * 1.05} (\sigma_{BAL \leq 10000} (account))$$

谢谢！