查询语言

- 查询语言：用户用来从数据库中请求获取信息的语言
- “纯”查询语言：
  - 关系代数 – SQL的基础
  - 元组关系演算
  - 域关系演算
- “纯”查询语言奠定了人们使用查询语言的基础，如SQL

关系代数

- 在某种程度上是过程化语言
- 六个基本运算
  - Select　选择
  - Project　投影
  - Union　并
  - set difference　差（集合差）
  - Cartesian product　笛卡儿积
  - Rename　更名（重命名）
- 用户输入一个或两个关系，并得到新的关系

关系代数

- 附加运算
  - Set intersection　交
  - Natural join　自然连接
  - Division　除
  - Assignment　赋值

选择运算

- 例：关系r

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

注：执行选择时，选择条件必须是针对同一元组中的相应属性值代入进行比较。

- 选择元组，满足：A = B and D > 5
  - $\sigma_{A = B \text{ and } D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

选择运算

- 定义：$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$，$\sigma$ 读作sigma，其中p为选择谓词
- 其中p是由逻辑连词 $\wedge$（与），$\vee$（或），$\neg$（非）连接起来的公式。逻辑连词的运算对象可以是包含比较运算符<、<=、>、>=、=和><的表达式
- 例如：$\sigma_{dept\_name = 'Finance'}(department)$

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

投影运算

- 例：关系r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- 选择属性A、C
  - 投影：$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

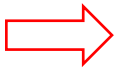
=

A	C
α	1
β	1
β	2

投影

- 定义：
  - $\Pi_{A1, A2, \ldots, Ak}(r)$，$\Pi$读作pi，$A_1$，… A_k是属性名，r为关系名
  - 其结果为保留此k列的值，并删除重复的行
- 例如：$\Pi_{building}(department)$

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000



building
Watson
Taylor
Painter
Packard

并运算

- 例：关系r

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- 并运算
  - r ∪ s:

A	B
α	1
α	2
β	1
β	3

并运算

- 定义：$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$
- $r \cup s$条件：
  - 等目，同元，即他们的属性数目必须相同
  - 对任意i，r的第i个属性域和s的第i个属性域相同
- 例如：$\prod_{name}$（instructor）$\cup$ $\prod_{name}$（student）

差运算

- 例：关系r，s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- 差运算
  - r – s:

A	B
α	1
β	1

差运算

- 定义：$r - s = \{t \mid t \in r \text{ and } t \notin s\}$
- r – s条件：
  - 等目，同元，即他们的属性数目必须相同
  - 对任意i，r的第i个属性域和s的第i个属性域相同

广义笛卡尔积

- 例：关系r ， s

- 广义笛卡尔积
  - r x s:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

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

广义笛卡尔积

- 定义：r x s = { {t q} | t∈r and q∈s}
- 假设r（R）的属性和s（S）的属性没有交集
- 如果r（R）和s（S）的属性有交集，那么必须重命名这些有交集的属性

广义笛卡尔积

- 例：关系r ， s

A	B
α	1
α	2
β	1

r

B	C
k	2
d	3

s

r x s =

A	r. B	s. B	C
α	1	k	2
α	2	k	2
β	1	k	2
α	1	d	3
α	2	d	3
β	1	d	3

复合运算

- 可以使用多种运算符构建表达式
- 例： $\sigma_{A=C}(r \times s)$

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

$r \times s =$

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	20	a
β	2	β	20	b

更名运算

- 允许我们使用其他名字指代关系
- 例：$\rho_X(E)$， ρ 读作rho，返回表达式E的结果，并把名字X赋给了它。假设关系代数表达式E是n元的，则表达式

  $\rho_{X(A1, A2, \ldots, An)}(E)$（对关系E及其属性都重命名）

  返回表达式E的结果，并赋给它名字X，同时将属性重命名为A1，A2，…，An.

银行示例

- *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)*

查询示例

- 例1：找出贷款额大于$1200的元组

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

- 例2：找出贷款大于$1200的贷款号

$$\Pi_{loan\text{-}number}(\sigma_{amount > 1200}(loan))$$

loan (<u>loan-number</u>, branch-name, amount)

查询示例

- 例3：找出有贷款或有帐户或两者兼有的所有客户姓名

$$\Pi_{customer\text{-}name}(borrower) \cup \Pi_{customer\text{-}name}(depositor)$$

- 例4：找出至少有一个贷款及一个账户的客户姓名

$$\Pi_{customer\text{-}name}(borrower) \cap \Pi_{customer\text{-}name}(depositor)$$

depositor (*customer-name*, *account-number*)

borrower (*customer-name*, *loan-number*)

查询示例

- 例5：找出在Perryridge 分支机构有贷款的顾客姓名

查询1：$\Pi_{customer\text{-}name}\ (\sigma_{branch\text{-}name=\text{“}Perryridge\text{”}}\ (\sigma_{borrower.loan\text{-}number\ =\ loan.loan\text{-}number}(borrower\ x\ loan)))$

查询2：$\Pi_{customer\text{-}name}\ (\sigma_{borrower.loan\text{-}number\ =\ loan.loan\text{-}number}\ (borrower\ x\ (\sigma_{branch\text{-}name=\text{“}Perryridge\text{”}}(loan))))$

查询2更好

loan (*loan-number*, *branch-name*, *amount*)
borrower (*customer-name*, *loan-number*)

查询示例

- 例6：找出在Perryridge分支机构有贷款，但在其他分支机构没有账号的顾客姓名

查询1：$\Pi_{customer\text{-}name}\ (\sigma_{branch\text{-}name\ =\ \text{“Perryridge”}}$

$(\sigma_{borrower.\ loan\text{-}number\ =\ loan.\ loan\text{-}number}(borrower\ \times\ loan)))$

$-\ \Pi_{customer\text{-}name}(depositor)$

查询2：$\Pi_{customer\text{-}name}\ (\sigma_{borrower.\ loan\text{-}number\ =\ loan.\ loan\text{-}number}$

$(borrower\ \times\ (\sigma_{branch\text{-}name=\ \text{“Perryridge”}}(loan))))\ -$

$\Pi_{customer\text{-}name}\ (depositor)$

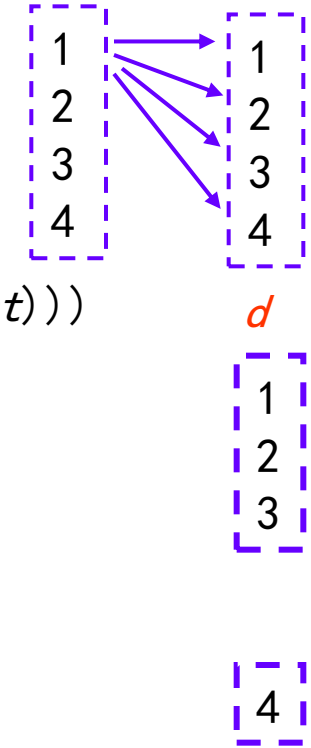
查询示例

- 例7：找出银行中最大的账户余额
  - 将*account*关系重命名为*d*
  - 第一步：找出由非最大余额构成的临时关系

$$\Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_{d}(account)))$$

  - 第二步：找出最大余额

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_{d}(account)))$$



附加运算

- 定义一些附加运算，它们虽不能增加关系代数的表达能力，但却可以简化一些常用的查询
  - 集合交
  - 自然连接
  - 除
  - 赋值

交运算

- 例：关系r

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- 交运算
  - r ∩ s：

A	B
α	2

交运算

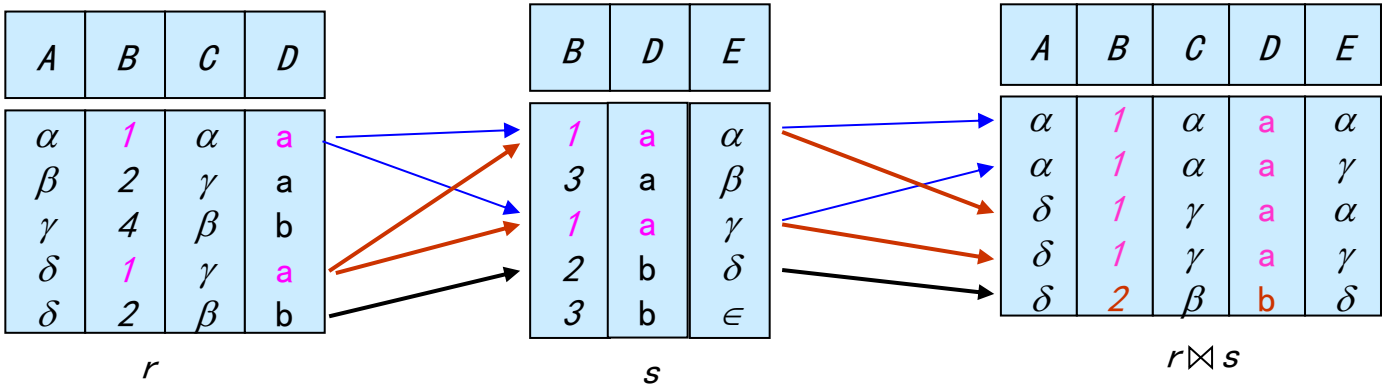
- 定义：$r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$
- 假设：
  - r和s同元
  - r和s的属性域是可兼容的
- $r \cap s = r - (r - s)$

自然连接

- r⋈s
- 例：R = (*A*, *B*, *C*, *D*)　　S = (*E*, *B*, *D*)
  - 关系r和s自然连接的结果模式为：(*A*, *B*, *C*, *D*, *E*)
  - $r \bowtie s = \prod_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$
- 设关系r和s分别代表模式R和S，那么r⋈s是对模式R∪S运算后的关系表示：
  - 考虑r的每一个元组t_r，s的每一个元组t_s
  - 如果t_r和t_s 在 R∩S 的公共属性下有相同的值，那么向结果集中插入元组t：
    - 元组t与t_r有相同的值
    - 元组t与t_s有相同的值

自然连接

- 例：关系r，s



注：

(1) r，s必须含有共同属性（名，域对应相同），

(2) 连接二个关系中同名属性值相等的元组

(3) 结果属性是二者属性集的并集，但消去重名属性。

theta连接

- theta连接：$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$
  - θ是模式 $R \cup S$ 属性上的谓词
- theta连接是自然连接的扩展

除运算

- r ÷ s 适用于包含了“对所有的”此类短语的查询
- 例:查询选修了所有课程的学生的学号

enrolled

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	80

÷

course

Cno
1
2
3

=

Sno
95001

$$\prod_{Sno,\ Cno}(enrolled) \div \prod_{Cno}(course)$$

除运算

- 设r和s分别代表模式R和S的关系
  - $R = (A_1, \cdots, A_m, B_1, \cdots, B_n)$
  - $S = (B_1, \cdots, B_n)$
  - r ÷ s 的结果代表模式 R－S 的关系：
    - $R - S = (A_1, \cdots, A_m)$
    - $r \div s = \{t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s(tu \in r)\}$

      注：商来自于$\prod_{R-S}(r)$，并且其元组t与s所有元组的拼接被r覆盖

除运算

- 例：关系r，s

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
∈	6
∈	1
β	2

r

B
1
2

s

$r \div s =$

A
α
β

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

除运算

- 例，关系r，s

A	B	C	D	E
α	a	α	a	1
α	b	γ	a	1
α	b	γ	b	1
β	a	γ	a	1
γ	a	γ	c	2
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	c	β	b	1

r

D	E
a	1
b	1

s

	A	B	C	D	E
1	α	a	α	a	1
2	α	b	γ	a	1
	α	b	γ	b	1
3	β	a	γ	a	1
	β	a	γ	b	3
4	γ	a	γ	c	2
	γ	a	γ	a	1
	γ	a	γ	b	1
5	γ	c	β	b	1

$r \div s =$

A	B	C
α	b	γ
γ	a	γ

除运算

R

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c1
a1	b2	c2
a2	b1	c2
a1	b2	c3
a1	b2	c4
a1	b1	c5

S

C
c1

Q

A	B
a1	b1
a2	b1
a1	b2

S

C
c1
c4
c2
c3

Q

A	B
a1	b2

S

B	C	D
b1	c1	d1
b2	c1	d2

Q

A
a1

$Q = R \div S$

S

C
c1
c2

Q

A	B
a1	b2
a2	b1

例:从SC表中查询至少选修1号课程和3号课程的学生学号

SC

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	80

$\div$

临时表K

Cno
1
3

$=$

Sno
95001

$\prod_{Sno,\ Cno}(sc) \div K$

除运算

- 性质
  - 若$q = r \div s$，则q 是满足$q \times s \subseteq r$的最大关系
- 用基本代数运算来定义除运算
  - 设r(R)和s(S)已知，且$S \subseteq R$：

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

  - 为什么：
    - $\prod_{R-S,S}(r)$ 重新排列r的属性顺序
    - $\prod_{R-S}(\prod_{R-S}(r) \times s) - \prod_{R-S,S}(r))$找出$\prod_{R-S}(r)$中的元组t，得到某些元组$u \in s$，$tu \notin r$

赋值运算

- 赋值运算(←)可以使复杂的查询表达变得简单
  - 使用赋值运算，可以把查询表达为一个顺序程序，该程序包括：
    - 一系列赋值
    - 一个其值被作为查询结果显示的表达式
  - 对关系代数查询而言，赋值必须是赋给一个临时关系变量
- 例：可以把$r \div s$写作，

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

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

$$result = temp1 - temp2$$

  - 将 ← 右侧的表达式的结果赋给 ← 左侧的关系变量，该关系变量可以在后续的表达式中使用

查询示例

- 例1：找出至少拥有一个“市区”和“住宅区”分支机构的账户的客户姓名

查询1：$\Pi_{CN}(\sigma_{BN = \text{“Downtown”}}(depositor \bowtie account)) \cap \Pi_{CN}(\sigma_{BN = \text{“Uptown”}}(depositor \bowtie account))$

其中，*CN*表示“*customer-name*”，*BN*表示“*branch-name*”

查询2：$\Pi_{customer\text{-}name,\ branch\text{-}name}(depositor \bowtie account) \div \rho_{temp(branch\text{-}name)}(\{(\text{“Downtown”}), (\text{“Uptown”})\})$

查询示例

- 例2：找出拥有布鲁克林市所有分支机构的帐户的客户姓名

$$\prod_{customer\text{-}name,\ branch\text{-}name}(depositor \bowtie account) \div \prod_{branch\text{-}name}(\sigma_{branch\text{-}city = \text{“}Brooklyn\text{”}}(branch))$$

- 例3：查询选修了全部课程的学生学号和姓名
  - 涉及表：课程信息*Course*(*cno*, *cname*, *pre-cno*, *score*)，选课信息*SC*(*sno*, *cno*, *grade*)，学生信息*Student*(*sno*, *sname*, *sex*, *age*)
  - 当涉及到求“全部”之类的查询，常用“除法”
  - 找出全部课程号：$\prod_{Cno}(Course)$
  - 找出选修了全部课程的学生的学号：
  - $\prod_{Sno,\ Cno}(SC) \div \prod_{Cno}(Course)$
  - 与*Student*表自然连接（连接条件*Sno*）获得学号、姓名

$$(\prod_{Sno,\ Cno}(SC) \div \prod_{Cno}(Course)) \bowtie \prod_{Sno,\ Sname}(Student)$$

总结

- 并、差、交为双目、等元运算
- 笛卡尔积，自然连接，除为双目运算
- 投影、选择为单运算对象
- 关系运算的优先级：
  - 投影
  - 选择
  - 笛卡尔积
  - 连接、除
  - 交
  - 并、差