ch6 形式化关系查询语言

6.1关系代数

基本运算

选择运算

括号里面是FROM哪个表，下标是谓词

选择(selelct)运算选出满足给定谓词的元组。我们用小写希腊字母 sigma(σ)来表示选择，而将谓词写作 σ 的下标。参数关系在 σ 后的括号中。因此，为了选择关系 *instructor* 中属于“物理(Physics)”系的那些元组，我们应写作：

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

投影运算

返回含有参数的关系
显示instructor表中的ID，name，salary作为一表结果

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

并运算

找到2009年秋季开设的所有课程ID

$$\Pi_{course\_id}(\sigma_{semester = \text{"Fall"} \wedge year = 2009}(section))$$

差运算

找到2009秋季开设但2010年没有开设的课程

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

笛卡尔积

找到物理系的教师以及他们教授的课程
第一步：
物理系的所有教师和课程做笛卡尔积

$$\sigma_{dept\_name = \text{"Physics"}}(instructor \times teaches)$$

第二步：

在笛卡尔积的结果中选择instructorID和teacherID相等的

$$\sigma_{instructor.ID = teaches.ID}(\sigma_{dept\_name = \text{"Physics"}}(instructor \times teaches))$$

就可以得到在 *instructor* × *teaches* 中那些只与物理系教师以及他们所教课程相关的元组。

最后，由于我们只需要物理系教师的名字，以及他们所教课程的 *course_id*，我们进行投影：

$$\Pi_{name, course\_id}(\sigma_{instructor.ID = teaches.ID}(\sigma_{dept\_name = \text{"Physics"}}(instructor \times teaches)))$$

更名运算

更名运算的另一形式如下。假设关系代数表达式 E 是 n 元的，则表达式

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

返回表达式 E 的结果，并赋给它名字 x，同时将各属性更名为 A_1，A_2，…，A_n。

其中A可以省略

找到大学里最高的工资

现在可以把非最高工资构成的临时关系写作：

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

通过这一表达式可以得到 *instructor* 中满足如下条件的那些工资：在关系 *instructor*（更名为 *d*）中还存在比它更高的工资。结果中包含了除最高工资以外的所有工资。图 6-11 表示这个关系。

图 6-11 子表

$(\sigma_{instructor.salary <}$

$\rho_d(instruc$

2. 步骤 2：查找大学里最高工资的查询可写作：

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

该查询的结果如图 6-12 所示。

salary
95000

图 6-12 大学里的最高工资

附加的关系代数运算

集合交运算

我们要定义的第一个附加的关系代数运算是**集合交**（intersection）（∩）。假设我们希望找出在 2009 年秋季和 2010 年春季都开设的课程。使用集合交运算，我们可以写为

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

此查询产生的关系如图 6-13 所示。

course_id
CS-101

自然连接运算

我们现在就可以给出自然连接的形式化定义。设 $r(R)$ 和 $s(S)$ 是两个关系。r 和 s 的**自然连接**（natural join）表示为 $r \bowtie s$，是模式 $R \cup S$ 上的一个关系，其形式化定义如下：

$$r \bowtie s = \Pi_{R \cup S}(\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \cdots \wedge r.A_n = s.A_n}(r \times s))$$

其中 $R \cap S = \{A_1, A_2, \cdots, A_n\}$。

赋值运算

有时通过给临时关系变量赋值的方法来写关系代数表达式会很方便。**赋值**(assignment)运算用←表示，与程序语言中的赋值类似。为了说明这个运算，我们来看自然连接运算的定义。我们可以把 $r \bowtie s$ 写作：

$$temp1 \leftarrow R \times S$$
$$temp2 \leftarrow \sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \cdots \wedge r.A_n = s.A_n}(temp1)$$
$$result = \Pi_{R \cup S}(temp2)$$

外连接运算

外连接与自然连接的关系：外连接允许包括空值的数据

左外连接(left outer join)(⟕)取出左侧关系中所有与右侧关系的任一元组都不匹配的元组，用空值填充所有来自右侧关系的属性，再把产生的元组加到自然连接的结果中。图 6-17 中，元组(58583，Califieri，History，62000，*null*，*null*，*null*，*null*)即是这样的元组。所有来自左侧关系的信息在左外连接结果中都得到保留。

右外连接(right outer join)(⟖)与左外连接相对称：用空值填充来自右侧关系的所有与左侧关系的任一元组都不匹配的元组，将结果加到自然连接的结果中。图 6-18 中，(58583，*null*，*null*，*null*，*null*，Califieri，History，62000)即是这样的元组。因此，所有来自右侧关系的信息在右外连接结果中都得到保留。

全外连接(full outer join)(⟗)既做左外连接又做右外连接，既填充左侧关系中与右侧关系的任一元组都不匹配的元组，又填充右侧关系中与左侧关系的任一元组都不匹配的元组，并把结果都加到连接的结果中。

扩展的关系代数运算

广义投影

第一个运算是**广义投影**(generalized-projection)，它通过允许在投影列表中使用算术运算和字符串函数等来对投影进行扩展。广义投影运算形式为：

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

其中 E 是任意关系代数表达式，而 F_1，F_2，…，F_n 中的每一个都是涉及常量以及 E 的模式中属性的算术表达式。最基本的情况下算术表达式可以仅仅是一个属性或常量。通常来说，在表达式中可以使用

聚集

当需要使用count sum average这类函数时

找出2010年春季学期有课的教师数（注意这里有distinct）

$$\mathcal{G}_{\text{count\_distinct}(ID)}(\sigma_{semester = \text{"Spring"} \wedge year = 2010}(teaches))$$

有时候我们希望对一组元组集合而不是单个元组集合执行聚集函数。作为示例，考虑查询“求出每个系的平均工资”。我们把此查询表达如下：

$$_{dept\_name}\mathcal{G}_{average(salary)}(instructor)$$

图 6-19 显示了通过 *dept_name* 属性对关系 *instructor* 进行分组得到的元组，这是计算查询结果的第一步。然后对每个组执行指定的聚集，查询结果如图 6-20 所示。

dept_name	*salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

6.2元组关系演算

查询示例

$$\exists t \in r(Q(t))$$

表示“关系 r 中存在元组 t 使谓词 $Q(t)$ 为真”。

用这种记法，我们可以将查询“找出工资大于 80 000 美元的所有教师的 *ID*”表述为：

$$\{t \mid \exists s \in instructor(t[ID] = s[ID] \wedge s[salary] > 80000)\}$$

我们可以这样来读上述表达式：“它是所有满足如下条件的元组 t 的集合：在关系 *instructor* 中存在元组 s 使 t 和 s 在属性 *ID* 上的值相等，且 s 在属性 *salary* 上的值大于 80 000 美元”。

6.3域关系演算

查询的例子

- 找出工资在 80 000 美元以上的教师的 *ID*、*name*、*dept_name* 和 *salary*：

$$\{< i,n,d,s > \mid < i,n,d,s > \in instructor \wedge s > 80000)\}$$

- 找出工资大于 80 000 美元的所有教师的姓名：

$$\{< i > \mid \exists n,d,s(< i,n,d,s > \in instructor \wedge s > 80000)\}$$