计算机科学 1998 Vol. 25№. 2

OQL 逻辑优化准则

OQL Logical Optimization Rules

钟 武 胡守仁

(国防科学技术大学计算机系 长沙 410073)

摘 要 Rewrite the formula, which is equivalent to the semantic of query sentence with form of select-from-where, by means of some optimizing rules. The final formula rewrited in terms of these rules possesses following characteristic: during the term of running, it can dynamically select the optimum loop nesting relation and corresponding form of the moving-out of loop invariants with computed result of atomic predicate to accomplish OQL queries.

关键词 OODB, Query optimization, OQL

1. 前言

对数据库查询的响应时间是数据库性能的重要指标。降低查询时间的关键是降低查询计算代价。因此，查询优化的研究工作分两个方向展开：一是代价估算模型的研究；二是优化手段的研究，其主要手段有：①代数变换。这类优化主要依赖于代数操作的等价性。但它有一个很大的缺点，即：对查询中的用户定义的对象操作方法显得无能为力；②利用方法的语义知识进行变换。由于对象行为在逻辑上和应用接口上具有抽象性和封装性，因此，优化又分为根据抽象的行为语义的优化和根据行为动作的具体实现的优化；③将查询从概念模式转换到具体实现的物理模式，以产生优化的执行计划。

我们以 ODMG93 为标准，为 C++ OOPL 环境扩充了对象永久化和查询永久对象等数据库功能。对对象库的查询访问可通过 OQL 语句完成。对含有 OQL 查询语句的 C++ 程序，均要通过预编译，将 OQL 查询语句变换成 C++ 代码。OQL 查询的逻辑优化就在此阶段进行。考虑到在 OQL 查询语句中，用户会频繁地使用自定义的对象间的比较方法，而方法的实现均较复杂且计算代价大，因此优化的目标是尽可能地减少计算代价。本文对计算代价的估算模型不做讨论，读者可根据自己所设计系统的特点自行确定。

为降低计算代价，本文以程式优化中的不变式外提思想为基础，并在此基础上做了如下的扩充：一是提出最佳循环嵌套关系，以最佳地适应不变式外提；二是原子谓词公式的判断尽早外提，以减少不必要的计算；三是变换产生的程式在运行时，能依查询比较中间结果，动态地选择最佳循环嵌套关系，而不是以一种循环嵌套关系及其相应的不变式外提形式来完成整个 OQL 查询过程。

2. OQL 等价程式及假设

本文将讨论 select-from-where 形式的查询语句的优化。OQL 语句的简要语法结构如下：

```
select select 处理表达式
from x1 in ex1, x2 in ex2, …, xn in exn
where P(A1, …, Am)
```

其中：①from 子句中的 ex_1 是计算结果为对象集合的 C++ 表达式；x_i 是 OQL 中定义的变量(以下称循环变量)，其值域是 ex_i 计算出来的对象集合；(i=1, …, n)；②where 子句中的 $P(A_1, \cdots, A_m)$ 是由原子谓词公式 $A_1, \cdots, A_m$ 构成的谓词公式。A_i 是计算结果为布尔型的 C++ 表达式。因此，它可含有用户自定义的计算代价高的方法。谓词公式中的逻辑运算符为“and”、“or”、“not”。(i=1, …, m)；③select 子句中的 select 处理表达式是 C++ 表达式。

与上述 OQL 语义等价的程式(1)为：

```
初始化：构造一个空集合对象 SET;
for(x1, …, xn)                                    (1)
if P(A1, …, Am)then *
```

其中，“*”表示“将 select 处理表达式产生的对象置于 SET 中”；“for$(x_1, \cdots, x_n)$”表征如下循环嵌套关系：

```
for x1 in ex1
  ⋮
for xn in exn
```

对于循环变量 x_i 和 $x_j(i<j)$，若 x_i 是表达式 ex_j 的变量参数，则称 x_j 对 x_i 有依赖关系。若存在循环变量 x_g，x_j 对 x_g 有依赖关系，x_g 对 x_i 有依赖关系，则 x_j 对 x_i 有依赖关系。x_j 对 x_i 的依赖关系决定了循环层 for x_j in ex_j 只能嵌入循环层 for x_i in ex_i，也即决定了它们之间的循环嵌套关系。

在以后的 OQL 查询优化的讨论中，程式(1)中的初始化部分将省略。另外，在讨论对程式(1)进行优化变换时，我们假设：

①对任意两个原子谓词公式 A_i 和 $A_j(i\neq j)$，有：A_i 先于 A_j 计算和 A_j 先于 A_i 计算是等价的。此假设保证：任意两个原子谓词公式之间的计算次序与它们的计算结果无关；

②如果 x_j 对 $x_i(i<j)$无依赖关系，则有：ex_i 先于 ex_j 计算和 ex_j 先于 ex_i 计算是等价的。此假设保证：当 x_j 不依赖 x_i 时，for $\langle x_j\rangle$ 嵌入 for$\langle x_i\rangle$与 for$\langle x_j\rangle$嵌入 for$\langle x_i\rangle$等价；

③如果 x_j 不是 A_i 的变量参数，则在计算次序上有：ex_j 先于 A_i 计算和 A_i 先于 ex_j 计算是等价的。此假设保证：在 A_i 外提到 for$\langle x_j\rangle$外时，A_i 和 ex_j 的计算结果与 A_i 嵌入 for$\langle x_j\rangle$时的计算结果一致。

3. 优化准则

准则 1 确定最佳循环嵌套关系，并在此循环嵌套关系上进行循环不变式外提。

所谓确定最佳循环嵌套关系是指：在满足由循环变量间的依赖关系所决定的循环嵌套关系的条件下，确定一种循环嵌套关系 G，使得：对任何一种可能的循环嵌套关系 H，通过在 H 上对原子谓词公式进行不变式外提的变换后，程式的计算代价不会低于在 G 上进行不变式外提变换后的计算代价。

例如：对于待变换的程式(2)，x_2 和 x_3 均依赖于 x_1；x_2 和 x_3 之间无依赖关系。则可能的循环嵌套关系只有：for $\langle x_1,x_2,x_3\rangle$、for$\langle x_1,x_3,x_2\rangle$。$A_1(x_1,x_2)$表示原子谓词公式 A_1 的计算要涉及到 x_1 和 x_2；$A_2(x_1,x_3)$的表征含义同 $A_1(x_1,x_2)$。对程式(2)进行不变式外提变换后，产生程式(3)：

```
for⟨x1,x2,x3⟩
if A1(x1,x2)and A2(x1,x3)then ※                    (2)
for ⟨x1,x2⟩
{T1=A1(x1,x2);
for⟨x3⟩
if T1 and A2(x1,x3)then ※}                         (3)
```

程式(3)的计算代价可大致估算为：

$$\text{cost}(ex_1)+\text{cyc}(x_1)*\text{cost}(ex_2)+\text{cyc}(x_1,x_2)*\text{cost}(A_1)+\text{cyc}(x_1,x_2)*\text{cost}(ex_3)+\text{cyc}(x_1,x_2,x_3)*\text{cost}(A_2)+\text{cyc}(x_1,x_2,x_3)*\text{rate}(A_1\text{ and }A_2)*\text{cost}(※) \quad ①$$

其中，函数 cost(expression)用于估算表达式 expression 的计算代价；函数 cyc(x)、cyc(x,y)、cyc(x,y,z)分别用于估算循环变量 x 所对应循环层的循环次数、循环变量 x 和 y 所对应双重循环层的循环次数、循环变量 x、y 和 z 所对应三重循环层的循环次数；函数 rate(P)用于估算谓词公式 P 取 True 值的概率。

对循环嵌套关系 for$\langle x_1,x_3,x_2\rangle$所对应的与程式(2)等价的程式进行不变式外提变换后，所生成的程式的计算代价可大致估算为：

$$\text{cost}(ex_1)+\text{cyc}(x_1)*\text{cost}(ex_3)+\text{cyc}(x_1,x_3)*\text{cost}(A_2)+\text{cyc}(x_1,x_3)*\text{cost}(ex_2)+\text{cyc}(x_1,x_3,x_2)*\text{cost}(A_1)+\text{cyc}(x_1,x_3,x_2)*\text{rate}(A_1\text{ and }A_2)*\text{cost}(※) \quad ②$$

根据估算表达式①和②的计算结果的比较，我们就能确定最适合不变式外提的最佳循环嵌套关系，亦即：若表达式①的计算结果不高于表达式②的计算结果，则 for$\langle x_1,x_2,x_3\rangle$是最佳循环嵌套关系。

由于 select 子句的计算只能处于循环嵌套层的最内层，因此对于循环嵌套关系 for$\langle x_1,\cdots,x_i,x_{i+1},\cdots,x_n\rangle$，其计算代价为：

$$\text{cyc}(x_1,\cdots,x_i,x_{i+1},\cdots,x_n)*\text{rate}(P)*\text{cost}(※) \quad ③$$

若 x_{i+1}不依赖 x_i，则对于循环嵌套关系 for $\langle x_1,\cdots,x_{i+1},x_i,\cdots,x_n\rangle$，其计算代价为：$\text{cyc}(x_1,\cdots x_{i+1},x_i,\cdots,x_n)*\text{rate}(P)*\text{cost}(※)$。由第 2 节中的假设 2，可得：$\text{cyc}(x_1,\cdots x_{i+1},x_i,\cdots,x_n)=\text{cyc}(x_1,\cdots,x_i,x_{i+1},\cdots,x_n)$。依此类推，对任意可能的循环嵌套关系，select 子句的计算代价与循环嵌套关系的变换无关，始终是③式的计算结果。因此，在估算程式的计算代价以确定最佳的循环嵌套关系时，不用再对 select 的计算代价进行估算，因而，在建立代价估算模型时，不用考虑函数 rate 的构造问题。

准则 2 对最靠循环外层的原子谓词公式尽早做出判断，化简谓词公式。

程式(4)和程式(5)的等价是准则 2 的出发点。对于程式(5)中的谓词公式 $P(A_1,\cdots,\text{True},\cdots,A_m)$和 $P(A_1,\cdots,\text{False},\cdots,A_m)$，由于分别存在常数项 True 和 False，因此可对两谓词公式做消除 True 值和 False 值的化简。例如：对于谓词公式 A_1 and$(A_2$ or $A_3)$，当 A_2 取 True 时，原谓词公式将简化为 A_1；当 A_2 取 False 时，原谓词公式将简化为 A_1 and A_3。

显然，当程式(5)中的谓词公式均简化后，所生成的新程式的计算代价不会高于程式(4)的计算代价。

```
for ⟨x1,…,xk⟩
{T=Ai,
for ⟨xk+1,…,xn⟩                                    (4)
if P(A1,…,T2,…,Am)then※}
for ⟨x1,…,xk⟩
if Ai then for ⟨xk+1,…,xn⟩
  if P(A1,…,True,…Am)then ※
  else for (xk+1,…,xn)
    if P(A1,…,False,…,Am)then ※                    (5)
```

准则 3 当最靠循环外层的原子谓词公式不止一个时，计算代价最小的原子谓词公式首先得到计算和判断。

准则 3 是对准则 2 的补充。例如：对于程式(6)，当 $cost(A_2)<cost(A_1)$时，选择先对 A_2 进行计算和判断，将生成程式(7)。

```
for ⟨x1,x2⟩
if A1(x1,x2)and A2(x1,x2)then ※                    (6)
for ⟨x1,x2⟩
if A2(x1,x2)then if A1(x1,x2)then ※                (7)
```

若选择先对 A_1 进行计算和判断，所生成程式的计算代价不会低于程式(7)的计算代价。

准则 4 反复利用准则 1、2、3 进行变换，消除外提的无用的原子谓词公式，直至不能再降低计算代价。

以程式(8)的优化转换为例，我们来讨论准则 4 是如何工作的。

```
for ⟨x1,x3,x3,x4⟩
if A1(x1,x2)and A2(x1,x3)or A3(x3,x4)then ※        (8)
```

利用准则 1 确定最佳循环嵌套关系(假设最佳循环嵌套关系为 for ⟨x_1,x_3,x_2,x_4⟩)，并将不变式外提后，产生程式(9)。

```
for ⟨x1,x3⟩
{T2=A2;
子程式}
子程式:for ⟨x2⟩                                    (9)
{T1=A1;
for ⟨x4⟩
if T1 and T2 or A3 then ※}
```

对于程式(9)中的子程式，其循环嵌套关系 for ⟨x_2,x_4⟩必是最佳的。因为：若它不是最佳的，则必存在 for ⟨x_4,x_2⟩是最佳的。因此，在 for⟨x_4,x_2⟩的基础上进行不变式外提变换后产生的程式的计算代价会低于程式(9)中的子程式的计算代价，这将导致 for ⟨x_1,x_3,x_4,x_2⟩比 for ⟨x_1,x_3,x_2,x_4⟩更佳这种矛盾性的结论。

对程式(9)利用准则 2 进行转化。由于最靠循环外层的原子谓词公式是 A_2，则对 A_2 尽早做出计算和判断。以 T_2 分别取 True 和 False 来简化谓词公式 T_1 and T_2 or A_3，产生程式(10)。

对程式(10)中的子程式 1，由于它的计算代价同程式(9)中的子程式的计算代价相同，因此无需再利用准则 1、2 来进行优化变换。

```
for ⟨x1,x3⟩
if A2 then 子程式 1
else 子程式 2
子程式 1:for ⟨x2⟩                                  (10)
      {T1=A1;
      for ⟨x4⟩
      if T1 or A3 then ※}
子程式 2:for ⟨x2⟩
      {T1=A1;
      for ⟨x4⟩
      if A3 then ※}
```

对程式(10)中的子程式 2，由于谓词公式已简化成 A_3，因此原子谓词公式 A_1 已无计算的必要。消除 A_1 后，子程式 2 变换成程式(11)。假设 x_4 不依赖 x_2，则对程式(11)利用准则 1、2 优化变换后，产生程式(12)。

```
for ⟨x2⟩
{for ⟨x4⟩
if A3 then ※}                                      (11)
for ⟨x4⟩
if A3 then for ⟨x2⟩
          ※                                        (12)
```

用程式(12)替换程式(10)中的子程式 2，就获得对程式(8)的最终的优化变换。从变换的过程来看：变换的每一步都是在前一步变换的基础上以减少计算代价为目标进行的；从变换产生的最终程式来看：程式在运行过程中，将随 A_2 的计算结果动态地选择最佳循环嵌套关系，当 A_2 为 True 时，选择 for ⟨x_2,x_4⟩；当 A_2 为 False 时，选择 for ⟨x_4,x_2⟩，即：程式在运行时，能依原子谓词公式的计算结果而动态地选择最佳循环嵌套关系完成 OQL 的计算。

参考文献

[1] The object database standard: ODMG-93, edited by R. G. G. Cattell, Morgan Kaufmann Publishers, 1994

[2] D. D. Staube and M. T. Ozsu, Query optimization and execution plan generation in object-oriented data management systems, IEEE trans. on knowledge and engineering Vol. 7, 1995

[3] D. D. Staube and M. T. Ozsu, Queries and query processing in object-oriented databse systems, ACM Trans. info. sys., 1990

[4] K. Aberer, G. Fischer, Semantic query optimization for methods in object-oriented database systems, IEEE 11th Intl. Conf. on Data Eng., 1995

[5] Z. Jiao and P. M. D. Gray, Optimization of methods in a navigational query language, Proc. 2nd Intl. Conf. on deductive and object-oriented databases, 1991

[6] R. S. G. Lanzelotte et al., Optimization of nonrecursive queries in OODBs, Proc. Intl. Conf. on deductive and object-oriented databases, 1990