面向对象数据库的查询优化*

吴胜利　王能斌

（东南大学计算机系　南京　210096）

摘要　查询优化一般分为逻辑和物理两层，但两者密切相关. 在一个具体系统的实现中，若把它们截然分开会影响优化效果，而把它们有机结合起来则更佳. 本文在联系代数的基础上探讨该问题，综合考虑了逻辑与物理 2 个方面，给出了面向对象数据库系统中的查询优化算法.

关键词　面向对象数据库，联系代数，查询优化.

查询处理与优化是面向对象数据库系统的重要部分，经过 10 余年的研究，已经取得了一些有意义的成果，如对象代数[1,2]、代数查询重写(Algebraic Query Rewriting)技术[3~5]、全局查询处理结构[5]、索引技术[6,7]等，已有一些商用的或原型系统的查询优化器[8,9]问世. 在初期对象代数的讨论较多，近期则较集中于代数查询重写技术.

查询优化可分成逻辑与物理两层. 传统作法是在逻辑层进行代数查询重写，此时只考虑代数表达式本身，而不涉及具体的物理数据库. 然后再对逻辑优化的代数表达式进行物理优化，例如可构造一查询代价(COST)模型，用启发式或穷举搜索算法找出代价较小者执行. 由于查询优化的逻辑与物理两层密切相关，在一个具体系统的实现中，把它们截然分开则优化效果不佳. 因此有必要综合考虑逻辑与物理优化. 文献[3]在讨论代数查询重写技术时，着重点虽仍为逻辑方面，但也同时考虑路径索引和对象聚集 2 种数据库物理信息. 本文除了路径索引和对象聚集外，还考虑反向属性[12]、类中对象的个数、类中对象选择谓词的选择率等，并考虑到谓词本身的复杂度，对谓词按复杂程度进行了分类.

为讨论方便，本文在面向对象数据库模式的基础上，考虑路径索引、对象聚集和反向属性，提出了增强数据库模式的概念.

面向用户的查询语言一般是描述性的，要将之变换为代数形式才适宜于优化. 本文在联系代数的基础上讨论优化方法，不涉及查询语言及查询语言向代数表达式的变换.

在联系代数表达式中，有 2 类最重要，一类为‘联系—选择—投影’复合操作；另一类为‘联系—选择—交—投影’复合操作，它们和关系模型中的‘选择—投影’、‘连接—选择—投影’复合操作具有同样的意义. 本文在综合考虑逻辑和上述物理信息的前提下，分别讨论其实现方法.

---

* 作者吴胜利，1963 年生，博士，主要研究领域为面向对象数据库系统. 王能斌，1929 年生，教授，博士导师，主要研究领域为数据库及信息系统.

本文通讯联系人：吴胜利，北京 100080，中国科学院软件研究所计算机科学开放研究实验室

本文 1996-01-22 收到修改稿

本文后面将'联系—选择—投影'复合操作简记为 ASP(associate_select_project),将'联系—选择—交—投影'简记为 ASIP(associate_select_intersect_project).

1 几点准备

1.1 联系代数

我们将复杂对象抽象为有向图,称作为联系式样(Association Pattern). 文献[10]*中对联系式样集合定义了联系、交、选择、投影等对象运算. 图 1 给出了 4 种运算的例子,所基于的数据库例子如图 2 所示. 详细讨论请参阅文献[1,10].

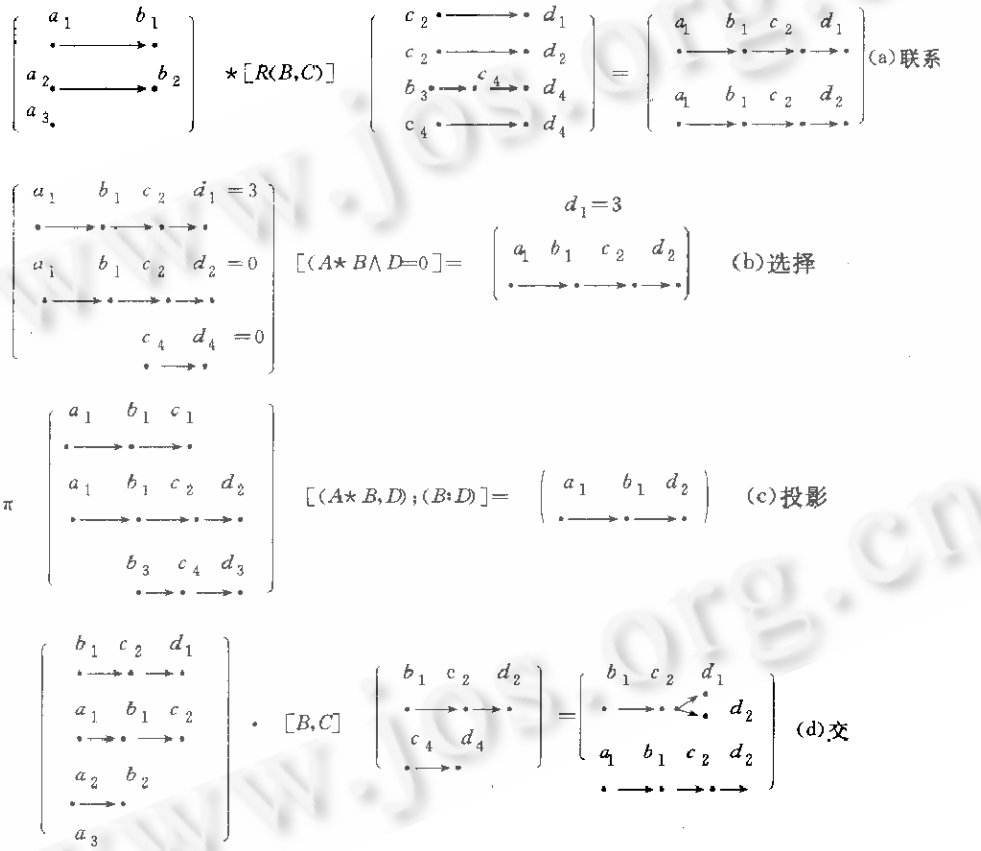

图1 联系代数的4种操作举例

在联系代数中,参加运算的操作数均为联系式样集合,集合中任一联系的操作数均为数据库对象联系图的子图. 图 2 中,A,B,C 和 D 均为模式中的类名,$a_i(1\leqslant i\leqslant 4)\in A$,$b_j(1\leqslant j\leqslant 3)\in B$,$c_k(1\leqslant k\leqslant 4)\in C$,$d_l(1\leqslant l\leqslant 4)\in D$ 均为联系式样. 例中联系操作将 2 个联系式样集合合并为一,具体作法是将两集合中满足条件的联系式样两两合并,条件 $R(B,C)$ 表示在数据库对象联系图中,b_j 到 c_k 有一有向边这样一个条件. 选择操作从联系式样集合中选出满足条件者,例中条件 $A*B\wedge D=0$ 表示所选联系式样要有从 a_i 到 b_j 的有向边且 $d_l=0$. 投影操作保留联系式样中某些特定的结点和边,而删除另一些. $[(A*B,D);(B:D)]$ 规定

* 注:文献[1]为基于无向图的联系代数,文献[10]将无向图改为有向图.

投影后的结果式样为 $a_i \cdot \xrightarrow{b_j} \cdot \rightarrow d_l$ 形式. 交操作亦对两集合中满足条件的联系式样两两合并,$[B,C]$表示两联系式样中有相同的子式样 $b_j \rightarrow c_k$ 才可合并.

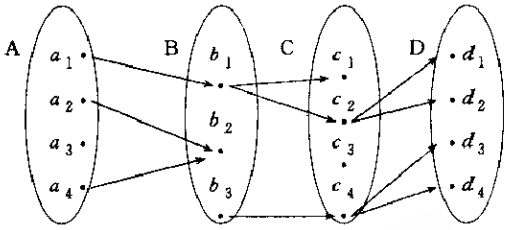

图2 数据库的对象联系图

本文涉及到的联系操作其第 2 个操作数均为单结点集合,这时和文献[9]中的实例化操作(Materialization)意义相同. 语义解释是为复杂对象的一属性(亦为复杂对象)找出其真实值. 如儿童 a 为一对象,其属性'父亲'是另一个对象,但在 a 中仅存放'父亲'对象的标识符,此时采用联系操作则为 a 找出其'父亲'对象.

1.2 数据库模式与增强模式

一个面向对象数据库包括模式和对象两层定义,它们均可用有向图表示. 由于模式有向图与对象有向图之间存在一对多对应关系,故我们用对象(联系式样)来说明数据库模式.

在一联系式样中,从结点 a 到结点 b 有一有向边,记作 $a \rightarrow b$,表示从 a 可快速访问 b,例如 a 对象的一属性值为 b,在 a 对象中存放了 b 对象的标识符,这样可很方便地从 a 访问 b. 但反过来从 b 访问 a 很困难,有可能需遍历 a 对象所属类的全部对象才能得知. 这样从 b 到 a 就没有有向边. 但如果从 b 到 a 定义了反向属性[11],或在 b 中建立了访问 a 的路径索引,或者 a,b 对象聚集存放,或者干脆用户亦定义 a 为 b 的一属性值,这时从 b 访问 a 也很容易,则从 b 到 a 也应有一条有向边. 若从 a 到 b 和从 b 到 a 均有有向边,且有同样的语义解释*,我们称 a,b 有双向边相连(记作 $a \leftrightarrow b$). 同样可定义路径和双向路径.

在数据库模式定义中,一般只包含用户定义的类及其相互间的关系. 现在 OODBMS 支持反向属性、聚集、索引等机制,有必要把这些信息增加到模式中去,我们把这种增加了此类信息的模式叫作增强模式. 因本文探讨的是非递归查询,在这种情况下,增强模式仍为树结构,但其中一些边变成了双向边,除此之外与原模式相同.

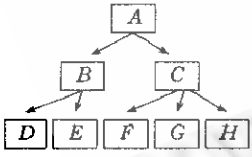

图3 例1模式定义

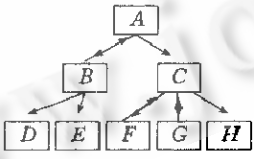

图4 例1增强模式

例 1:数据库模式定义如图 3 所示,现假定用户在 B 类中建立了访问 A 的路径索引,C,F,G 聚集存放,则其增强模式如图 4 所示.

为简便起见,本文后面将交替使用对象与联系式样这 2 个词表示相同的概念,也常使用模式中的结点名称表示类定义,或表示属于该类的所有联系式样的集合.

2 一般模式下的 ASP 与 ASIP 复合操作

为增强本文的可读性,我们先讨论在一般模式下(即未考虑路径索引、对象聚集和反向属性,此时模式呈 DAG 结构)ASP 与 ASIP 的优化方法,这实际上是下一节增强模式下 2

---

* 注:指 $a \rightarrow b$ 和 $b \rightarrow a$ 为同一联系的正反两面,如 $a \rightarrow b$ 为母子联系,$b \rightarrow a$ 为子母联系. 应将此与两类之间有多种联系的情形区分开.

类复合操作的特例.

2.1 ASP复合操作

假定初始时具有如下一般形式(投影暂未考虑):

$$\sigma(R_1 * R_2 * \cdots * R_m)[F] \tag{1}$$

这里 $R_1, R_2, \cdots, R_m$ 均为模式DAG中的单个结点,F 为选择谓词.

我们总可以把 F 改造成 $F_1 \wedge F_2 \wedge \cdots \wedge F_m \wedge F_t$,其中 F_1 中只涉及 R_1,F_2 只涉及 $R_2, \cdots$, F_m 只涉及 R_m,F_t 为任意形式. 这样,(1)式可写成如下形式:

$$\sigma(R_1 * R_2 * \cdots * R_m)[F_1 \wedge F_2 \wedge \cdots \wedge F_m \wedge F_t] \tag{2}$$

直接对(2)式进行运算不一定好,因为通过多次联系运算拼装而成的复合对象集合,很可能只有少数符合谓词条件而成为最终结果,这样许多工作就白做了. 对(2)式的优化思路是尽可能避免用联系操作拼装无用的复杂对象.

(2)式把选择谓词 F 分解成 $m+1$ 项的合取,这样我们就可以将选择分解为 m 个,与联系操作交替进行,这就可用下面的(3)式表示:

$$\sigma_m(\cdots(\sigma_2(\sigma_1(R_1)[F_1]) * (R_2)[F_2]) * \cdots * (R_m))[F_m \wedge F_t] \tag{3}$$

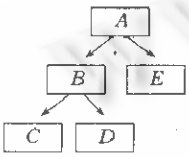

图5 模式有向图

(3)式的运算过程很有规律,除 σ_1 和 σ_m 外,其余均是联系操作后跟相应的选择操作,并且联系操作的第2个操作数是单结点的联系式样,其后的选择中谓词也只跟这个单结点的联系式样有关. 如果我们确定用(3)式的结构来进行联系—选择复合操作的话,剩下的问题是如何确定 $R_1, R_2, \cdots, R_m$ 的先后运算次序. 因为对一特定模式而言,存在多种运算次序,它们均能拼装成功所有的复杂对象. 如图5所示模式,存在8种次序,它们是: $ABCDE$, $ABCED$, $ABDCE$, $ABDEC$, $ABECD$, $ABEDC$, $AEBCD$ 和 $AEBDC$. 形成运算次序所应遵循的准则是:父结点一定要先于其子结点出现. 根结点(图5中 A 为根结点)总是排在首位.

假定我们知道任一类 R_i 中对象个数 N_i,类 R_i 中对象满足谓词 F_i 的选择率 S_i(这往往可通过对象索引得出),则我们可得到一较佳运算次序,由下面的算法1给出.

算法1. 确定 $R_1, R_2, \cdots, R_m$ 的运算次序,其中 R_1 为根结点.

输入:模式有向图,并包含 $N_i, S_i (1 \leqslant i \leqslant m)$ 等统计信息

输出:$R_1, R_2, \ldots, R_m$ 的一种排序

步骤:1. $T=\{R_1\}$; $R=\{\}$;
2. for $i=1$ to m do
3. 从 T 中挑选出 $N_i \times S_i$ 值最小的结点 R_i 输出;
4. $R=R \cup \{R_i\}$;
5. for $follow(T)$ 中的每一个元素 R_j
6. if $R_j \notin R$ then $T=T \cup \{R_j\}$;
7. end of for;
8. end of for;
9. end.

其中 $follow(T)$ 为 T 中结点有有向边指向的结点集合,亦即对任一 $R_i \in follow(T)$,一定存在一 $R_j \in T$,使 $R_i \rightarrow R_j$ 成立.

算法 1 主要包括 1 个两层嵌套循环，第 2～8 行为外层循环，执行 m 次，第 5～7 行为内层循环. 由于第 5 行的执行次数与模式有向图中的有向边数相等，故至多执行 $1/2m(m-1)$ 次. 采用散列方法可使第 6 行用 $O(1)$ 时间完成，所以整个算法的复杂性不超过 $O(m^2)$.

根据以上算法的输出可得一新的结点序列 $R_1, R_2', \cdots, R_m'$. 以此分别代替(3)中的 $R_1, R_2, \cdots, R_m$，并改动相应的选择运算，可望大大提高(3)式的运算效率.

在(3)式中，当遇到一新的结点时，先做联系操作，再做相应的选择操作. 另一种作法是先做选择操作，再做联系操作，这可由下面的(4)式描述：

$$\sigma_{m+1}((\sigma_1(R_1)[F_1]) * (\sigma_2(R_2)[F_2]) * \cdots * (\sigma_m(R_m)[F_m]))[F_t] \qquad (4)$$

我们规定运算自左至右进行(注意联系操作满足结合律). $R_1, R_2, \cdots, R_m$ 运算次序的确定同样可由算法 1 给出.

(3)式与(4)式中的选择运算在形式上稍有不同. 在(3)式中 σ_i 是对 $R_1 * R_2 \cdots * R_i (1 \leqslant i \leqslant m)$ 进行的，而(4)式中 σ_i 仅对 R_i 进行，但这 2 种形式之间的转换是很方便的.

在上述讨论中，我们对谓词本身没有进行什么考虑. 实际上，由于面向对象模型能支持一些复杂的应用，且可由用户定义操作或方法[11]，不同谓词的复杂程度差异很大，简单的只涉及一、二个比较操作，复杂的可包含计算量庞大的用户定义操作，如在 CAD、地理信息系统等应用中. 所以有必要考虑谓词本身的复杂性. 为便于进一步优化，将谓词分作 2 类：一般谓词和费时谓词. 对于费时谓词，我们尽量避免或只对尽可能少的对象进行.

如何划分一般谓词与复杂谓词值得考虑. 因为费时谓词一般都是由于其中费时的用户定义操作引起的，故可由划分用户定义操作是否费时来代替前者. 此外，查询语句中一个特定的谓词只在语句中出现一次，很少重现. 而用户定义的操作可反复应用，测定其费时与否更有意义. 我们规定一个谓词若含有费时的用户定义操作，则为费时谓词，否则为一般谓词.

下面讨论划分一般与费时用户定义操作的方法. 最简单的方法是由用户说明，但缺点是主观随意性较大，也可考虑采用静态分析方法，例如可根据相应目标程序的大小来决定. 还有一种办法是根据实际执行情况来确定. 例如可先设定所有的用户定义操作均为一般谓词，在经过一次(或数次)运行后，若其(平均)执行时间超过某个阈值，则认为其为费时的用户定义操作. 所有的用户定义操作均根据其以前的执行情况动态调整.

还有一点要说明的是，前面的讨论假定输出结果是一个个完整的复杂对象，包含其所有的组成成分，实际上往往并非如此，这就需要对结果做投影操作. 但若用联系操作拼装好整个对象后投影去掉一些分量，未免不划算，解决办法是对那些无需输出、且在后面的运算中亦无用的分量，不用联系操作去生成它们. 这只需要在算法 1 的第 6 行，将条件 $R_j \notin R$ 改成 $(R_j \notin R) \wedge (useful(R_j) = true)$ 即可.

实现函数 $useful(R_j)$ 要用到查询条件有向图和查询结果有向图，它们均是数据库模式结构图的子图，分别记载查询条件和结果所形成的有向图. 它们可从联系代数表达式方便地求出. 下面举一例说明：

例 2：设数据库模式结构如图 5 所示，对于下列查询表达式

$$\pi(\sigma(A * B * C * D * E)[A.B.C = \text{'}c\text{'} \wedge (A.E = \text{'}f\text{'}]))[(A, D, E)]$$

其查询条件有向图和结果有向图分别如图 6 和图 7 所示.

$useful(R_j)$ 只需简单地访问上述 2 个有向图. 若 R_j 至少是其中 1 个有向图中的结点，则

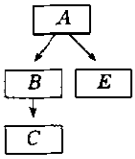

图6 例2的查询条件有向图

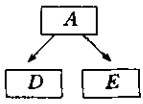

图7 例2的查询结果有向图

R_j 有用，需进行相应的联系操作，否则无需进行.

综合上述讨论，我们可得到一较完整的算法，将联系代数表达式转换成等价的优化形式.

2.2 ASIP 复合操作

在文献[1,10]中提供了交操作，它和关系模型中的连接操作有同样的意义.

假设 ASIP 复合操作具有如下形式(暂不考虑投影操作)：

$$(\sigma(R_1)[F_1]\cdot\sigma(R_2)[F_2])[F] \tag{5}$$

其中 $\sigma(R_1)[F_1]=\sigma(R_{11}*R_{12}*\cdots*R_{1m})[F_{11}\wedge F_{12}\wedge\cdots\wedge F_{1m}\wedge F_{1t}]$，$\sigma(R_2)[F_2]=\sigma(R_{21}*R_{22}*\cdots*R_{2m})[F_{21}\wedge F_{22}\wedge\cdots\wedge F_{2n}\wedge F_{2t}]$. F 是交操作的条件. F 中要涉及 R_1 和 R_2 中的结点，设 $\cdot[F_1]$是 F 中涉及 R_1 的结点集合，$\cdot[F2]$是 F 中涉及 R_2 的结点集合.

一般来说，交操作比联系和选择操作更费事，所以我们按 2.1 节讨论的方法先求$\sigma(R_1)[F_1]$和 $\sigma(R_2)[F_2]$，然后再求这 2 个联系式样的交.

在求 $\sigma(R_1)[F_1]$和 $\sigma(R_2)[F_2]$的过程中，应注意函数 $useful(R_j)$要稍作修改，即需新定义交操作条件有向图，其所含结点分别为 $\cdot[F_1]$与 $\cdot[F_2]$，作用与另 2 种有向图相同.

2.1 节我们将谓词分成 2 类，这里再进一步分成 3 类：一般谓词、费时谓词和极费时谓词. 在进行联系—选择操作时，我们先做具有一般谓词的选择，然后做具有费时谓词的选择，而具有极费时谓词的选择则放到交操作之后进行. 这总是可以做到的，考虑下面 2 种情形：①极费时谓词所在的结点未被以后的联系或交运用，将涉及该谓词的联系、选择操作放在交操作之后进行没有任何问题. ②极费时谓词所在的结点要被以后的联系或交运用，用永真谓词暂时替代极费时谓词，待做完交操作后再用极费时谓词做相应的选择操作. 还有一些问题与 2.1 节的相似，不再赘述.

3 增强模式下的复合操作

在讨论之前，先做一点准备.

定义 1. R 是增强模式中一结点，R 的双向联系集合 $birelation(R)$为所有与 R 有双向边相连的结点集合. 亦即对任一 $R_i\in birelation(R)$，即有 $R\leftrightarrow R_i$ 成立.

定义 2. R_1 和 R_2 是增强模式中两结点，且下列诸条件中必有一条成立：

a) $R_1\rightarrow R_2$; b) $R_2\rightarrow R_1$; c) $R_1\leftrightarrow R_2$

定义反转操作如下：

$\nabla(R_1*R_2)=R_2*R_1$; $\nabla(R_2*R_1)=R_1*R_2$

在增强模式下 ASP 复合操作的优化思想是这样的：根据上一节的(4)式进行计算，当我们要选择下一个结点时，设 R_i 是其中可供选择的一个. 但此时不象上一节那样，仅仅考虑 R_i，而且考虑 $birelation(R_i)$中的每一个结点. 如发现 $R_j\in birelation(R_i)$是最佳选择，则先做 $R_j'=\sigma_j(R_j)[F_j]$，其次做 $R_{ij}=\nabla(R_j'*R_i)$，再做 $\sigma(R_{ij})[F_i]$，最后和已得到的结果做联系运算，这样又可进行下一步的结果选择. 象这样选择面进一步放宽，优化效果更佳.

在此基础上，ASIP 复合操作亦可有效实现，方法与上一节相似.

4 小　结

本文从联系、选择和交这 3 种操作的关系入手，对面向对象模型中 2 类重要的复合操作 ASP 和 ASIP 进行了探讨，给出了实现算法. 由于该问题是面向对象模型中查询优化的关键，妥善解决该问题有助于全面解决面向对象数据库中的查询优化问题.

本文的主要创新之处在于：

a）指出 ASP 和 ASIP 是面向对象模型中查询优化的关键；

b）在实现 ASP 和 ASIP 的方法中本着一般操作尽可能先做、费时操作后做、无用操作要避免的指导思想，采用了选择与联系交替进行、交后做、投影贯穿始终的策略；

c）将条件谓词分成 3 类，可改变 b）中策略的应用，达到更佳的优化效果；

d）提出了增强模式的概念，并在增强模式下给出了 ASP 和 ASIP 的实现方法.

本文所提出的方法在绝大多数情况下均能得到较佳的结果，算法复杂性也较低，但在有些情况下不是最佳结果. 实际上，尽管在本文所讨论的范围而言，如不采用穷举搜索策略比较所有可行方案，亦不能保证在所有情况下均能得到最佳方案. 作为对优化查询表达式和查询处理两者的均衡考虑，本文所提出的方法具有较大的实用价值.

参考文献

1 Stanley Y W Su *et al*. Association algebra: a mathematical foundation for object-oriented databases. IEEE Transaction on Knowledge and Data Engineering, 1993, **5(5)**: 775～798.

2 Vandenberg S L, DeWitt D J. Algebraic support for complex objects with arrays, identity, and inheritance. Proceedings of ACM SIGMOD Conference, 1991. 158～167.

3 Cluet S, Delobel C. A global framework for the optimization of object-oriented queries. Proceedings of ACM SIGMOD Conference, 1992. 383～392.

4 Steenhagen H J *et al*. From nested-loop to join queries in OODB. Proceedings of 20th VLDB Conference, 1994. 619～629.

5 Straube D D, Ozsu M T. Queries and query processing in object-oriented database systems, ACM Transaction on Information System. 1990, **8(4)**: 387～430.

6 Kemper A, Moerkotte G. Advanced query processing in object bases using access support relations. Proceedings of 16th VLDB Conference, 1990. 290～301.

7 Sreenath B, Seshadri S. The hcC-tree: an effective index structure for object oriented database. Proceedings of 20th VLDB Conference, 1994. 203～213.

8 Orenstain J *et al*. Query processing in the object store database system. Proceedings of ACM SIGMOD Conference, 1992. 403～412.

9 Blakeley J A *et al*. Experience building the open OODB query optimation. Proceedings of ACM SIGMOD Conference, 1993. 287～296.

10 吴胜利，王能斌. 面向对象数据库中基于有向图的联系代数. 计算机学报，1997，**20(1)**：58～67.

11 Levy A Y *et al*. Query optimation by predicate move-around. Proceedings of 20th VLDB Conference, 1994. 96～107.

12 吴胜利，王能斌. 一种支持对象联系的有效方法. 计算机研究与发展，1995，**32(5)**：8～12.

QUERY OPTIMIZATION IN OBJECT-ORIENTED DATABASE SYSTEMS

WU Shengli WANG Nengbin

(*Department of Computer Science Southeast University Nanjing* 210096)

Abstract Query optimization traditionally involves two deeply connected levels that are qualified as logical and physical. In the implementation of a concrete system, dividing it in two completely is not a good way. Considering the problem comprehensively is much better, the authors discuss this in the paper based on association algebra, and give query optimization algorithms in object-oriented database system which taking into account both logical and physical aspects.

Key words Object-oriented database, association algebra, query optimization.