内嵌式数据库 引擎 优化 数据库 (20) 数据查询

计算机科学1998Vol. 25№. 6

86-89

内嵌式数据库引擎实现中的优化方法

The Optimization Methods of a Built-in Database Engine Implementation

李 磊 TP311.13

（中山大学计算机软件研究所 广州 510275）

摘 要 The implementation of a tight coupling PROLOG-DBMS is rather difficult. The major problem is to realize a simple but efficient built-in database engine. In this paper, we discuss some optimization algorithms for efficiently evaluating database queries. In the same time, the implementation of such database engine can be simplified by using the algorithms.

关键词 Database engine, Optimization algorithms, DBMS, PROLOG

近年来，PROLOG 获得了越来越广泛的应用，进化也很快。但是人们在用 PROLOG 开发实际应用时常常希望访问外部数据库，特别是 XBASE 数据库中的数据，为此 WIN-PROLOG 作出了努力。虽然 WIN-PROLOG 可以通过 ODBC 访问符合 ODBC 标准的数据库系统，但是存在如下问题：

1. 访问数据库是不透明的；

2. 效率不高。

如果能够为 PROLOG 研制一个内嵌式(BUILT-IN)数据库引擎，PROLOG 可以直接通过它透明和有效地访问外部数据库数据(XBASE)，显然是有意义的。

本文仅研究这样的数据库引擎实现中的查询公式求值优化技术。

这一数据库引擎应该具有如下性质：

* 要能访问某种有一定使用广度的数据库；

* 系统要同 PROLOG 提供的编译条件相容；

* 系统应该占用尽量少的资源(主要资源应留给 PROLOG 系统和应用程序使用)；

* 在占用尽量少的资源的情况下，应尽量提高数据库引擎的求值效率(在此条件下优化便显得十分重要)。

我们实现了一个满足上述性质的数据库引擎。由于我们采用了部分编译和基于逻辑的优化技术，因此提高了数据库引擎的求值效率而且简化了它的实现，达到了上述要求。

1 预备知识

对数据库的非聚类查询可以用逻辑表示。我们常称作逻辑查询公式，或简称为查询公式。

查询公式的求值效率是数据库系统的重要指标。因此查询公式的高效求值技术是数据库系统实现中的关键技术。查询公式求值器也是数据库系统引擎的核心部分。众所周知，逆波兰表示表达式求值算法是表达式求值的最经典算法。查询公式是一种特别的表达式，采用逆波兰表示表达式求值算法来对查询公式求值是十分自然的。因此目前绝大多数的数据库引擎中的查询公式求值采用了这种算法。由于在通常情况下，查询公式只有到程序运行时才能确定，因此查询公式的求值事实上是解释执行的。本文提出的基于范式的查询公式求值和优化算法采用了部分编译和基于逻辑的优化技术，因此不但具有较高的求值效率并且简化了求值器的实现算法。

在数据库关系模型中的五个基本运算是：交(I)、并(Y)、投影(π)、选择(σR)、和迪卡儿积(X)，其中最基本和最重要的是选择操作(σF)。选择操作中的 F 就是本文所述的查询公式。它对应于 SQL 中的 WHERE 子句。

查询公式是一类很特别的一阶逻辑公式，下面我们给出定义并讨论它们的性质。

项 常数，变量(字段名称)。

原子公式 $X\ \theta\ Y$，其中 X，Y 是项且 $\theta \in \{=, \neq, <, >, <=, >=\}$。

公式 原子公式：

W∧V,W∨V,和¬V是公式,如果W和V是公式。

性质1:设F是查询公式,事实上F是封闭的并且是由前束存在量词约束的。

例如:设F为X=a∧Y>100∧Y<1000,

事实上F表示的逻辑含义是

∃X∃Y(X=a∧Y>100∧Y<1000).

性质2:设F是查询公式,事实上存在与F等价的查询公式F',其中F'中不存在¬。

证明:把F中任意原子公式¬(XθY)转化为Xθ'Y,其中如果θ分别是=,≠,<,>,<=,>=,则θ'分别转换成≠,=,>=,<=,>,<。显然F与F'是等价的。

性质3:设F是查询公式,F有一等价的范式F'。

证明:设F是查询公式,由性质1知F的全部量词(存在量词)前置,因此对F的非量词部分做任何变换时都与变量无关,因此可以把不同的原子公式等同于不同的命题。根据命题逻辑范式定理有,任意F具有下面形式的等价范式F':

(A11∧…∧A1n)∨(A21∧…∧A2n)∨…∨(Ak1∧…∧A1nk)(析取范式)或

(A11∧…∧A1n)∧(A21∨…∨A2n)∧…∧(Ak1∨…∨A1nk)(合取范式)。

考虑实际应用的习惯性,我们选取合取范式作为F'。

事实上存在着十分简单的方法把任意命题公式翻译成等价的命题范式。

总结上述性质,我们可以得出结论:存在着简单和机械的算法把任意一个查询公式翻译成NOT-free合取范式。

2 查询公式的集合表示

表达式表示成逆波兰表示有两个主要目的:

* 去掉括号;

* 按照子表达式的求值顺序排列运算符和运算数的顺序。

由于合取范式具有固定的形式且具有固定的求值顺序,因此采取逆波兰表示已无意义。

假定所有的查询公式都已化为合取范式,例如:

S1∧S2∧…∧Sn. (1)

其中Si为:

Ai1∨Ai2∨…∨Aini. (2)

我们把(1)和(2)分别叫做合取式和析取子式。

事实上,在范式中的Si(或Aij)的顺序是没有关系的,因此我们可用集合表达合取式和析取子式。例如:

合取式

S1∧S2∧…∧Sn表示为{S1,S2,…,Sn};

析取子式

Ai1∨Ai2∨…∨Aini表示为{Ai1,Ai2,…,Aini};

查询公式

(A11∨…∨A1n)∧(A21∨…∨A2n)∧…∧(Ak1∨…∨A1nk)

表示为

{{A11,…,A1n},{A21,…,A2n},…,{Ak1,…,A1nk}}。

我们称上述表示为查询公式的集合表示。

3 求值算法

假定查询公式是集合表示的。显然查询公式的求值顺序是:原子公式,析取子式,最后是合取式。下面我们分别叙述其求值。

3.1 原子公式求值

在查询公式中原子公式是XθY,明显XθY是一个一阶谓词,但是是一个特别的谓词。

在一阶逻辑中为了给每个N元谓词一个解释(模型论),我们要定义一个Dn->{F,T}上的映射,或说定义一个N元关系Rn。如果某N元组rn∈Rn,则rn映射T,否则F。

XθY是一个二元谓词,我们可以给予XθY一个解释,或者说为XθY定义一个二元关系Rθ:如果任意二元组<x,y>,x,y∈D(解释空间)并且x,y满足θ关系,则(x,y)∈Rθ。事实上我们是无法显式构造Rθ,因为D常常是无限的。

在数据库中是不保留Rθ的,原子公式,XθY是由系统求值(通常叫内部谓词)。XθY的求值为:

$$A\_val(X\theta Y)=\begin{cases}1,\text{如果X与Y满足}\theta\text{关系}\\0,\text{否则}\end{cases}$$

其中X与Y满足θ关系是由系统自动判别的。

3.2 析取子式求值

设S,{Ai1,Ai2,…,Aini}是某析取子式的集合表示,S的求值为:

$$D\_val(S)=\begin{cases}1, & \text{如果有原子公式 Aij}\in\{Ai1,Ai2,\cdots,Aini\}\text{且 A\_val(Aij)=1}\\0, & \text{否则}\end{cases}$$

3.3 合取公式求值

设 F,{S1,S2,…,Sn}是一合取式的集合表示,F 的求值为:

C_val(F)=

1, 如果所有的析取子式 Si∈{S1,S2,…,Sn}且 D_val(Si)=1

0, 否则

3.4 查询公式的求值算法

设 F,{S1,S2,…,Sn}是一查询公式的集合表示,Si,{Ai1,Ai2,…,Aini}是析取子式集合表示,n 表示 F 中析取子式的个数,ni 表示每一析取子式中原子公式的个数,则 F 的求值算法可用类 C 语言表示如下:

```
boolean C_val(F,n)
        {
          for(i=1;i<=n;i++)
           if(D_val(Si,ni)==0) return 0;
          return 1;
        }
boolean D_val(Si,ni)
        {
          for(i=1;i<=ni;i++)
           if(A_val(Ai)==1) return 1;
          return 0;
        }
boolean A_val(A)
        {
           if(Xθ Y)==1) return 1
          else return 0;
        }
```

其中 C_val D_val A_val 函数分别完成合取公式,析取子式和原子公式的求值;原子公式 Xθ Y 的求值是由系统完成的。

4 优化

4.1 求值顺序优化

我们知道在逻辑演算中有如下性质:

性质1:0∧A1∧A2∧…∧An=0

性质2:1∨A1∨A2∨…∨An=1

假定 F,{S1,S2,…,Sn}是查询公式的合取集合表示并且假定 Si 集合中仅有一个元素,明显根据性质1有

C_val(F,n)=0 当且仅当 ∃Si(Si∈F ∧ A_val(Si)=0)。

或者说,在 F 中只要有一个 Si,它的析取子式求值为零则 F 的合取子式求值为零。意思是当如果已知存在某析取子式求值为零,则余下的析取子式求值是不必要的。根据这个性质,如果把最容易求值为零的析取子式(原子公式)优先求值,则余下的析取子式的求值可以省略,从而达到优化的目的。

我们认为原子公式 X θ Y 求值为零可能性是由 θ 确定的并且大小是如下排列的:

=,<(或>),<=(或>=),≠.

同理,假定 S,{A1,A2,…,An}是析取子式的集合表示,根据性质2,显然如果把最容易求值为1的原子式优先求值,则余下的原子式求值可以省略,同样可以达到优化的目的。

显然,原子公式 X θ Y 求值为一可能性大小是按为零可能性大小的反序排列的。

4.2 相关优化

假定 θ 是定义在线性集 R 上的关系,不妨假定 R 是整数集,通常情况下查询公式为 a<=X ∧X<=b,亦 a<=X<=b,其中 a,b∈R,X 是字段名字(可视为变量)。我们把 a<=X<=b 表示为[a,b]并叫做闭区间。

事实上,如果令 θ∈{=,<=,>=},则所有的原子公式都可定义唯一的闭区间(称为原子区间),因为:

X=a [a,a];

X<=a [-MAX,a];

X>=a [a,MAX],

其中 MAX 表示机器表示的最大整数。

对于特例 a<=X∧X<=b 直接表示成[a,b]并视为原子区间。

如果把原子区间用 ∧,∨ 相连,我们就得到了查询公式的区间表示。例如:

a<=X∧X<=b∧X<=c∧X<=d 的区间表示为[a,b]∧[c,d]

如果把区间视为集合,则任意两个集合 R1,R2 之间存在下述关系:

1. 无关 如果 R1∩R2=∅

2. 包含 如果 R1⊇R2

3. 相等 如果 R1⊇R2∧R2⊇R1

4. 相交 如果 R1∩R2≠∅∧R1和 R2不相等

事实上,设 R1,R2是区间,它们之间的上述关系在语法级上就可判别,因为:

假定 R1=[a,b],R2=[c,d],并且 a<=c

R1,R2无关 当且仅当 b<c

R1,R2包含 当且仅当 c>=a∧d<=b

R1,R2相等 当且仅当 c=a∧b=d

R1,R2相交 当且仅当 a<c∧c<=b

明显,如果 R1,R2 无关,则需满足 a<=X∧X<=b∧X<=c∧X<=d(无优化可言);

如果 R1,R2 包含,则只需满足 a<=X∧X<=b;

如果 R1,R2 相等,则只需满足 a<=X∧X<=b(或 X<=c∧X<=d);

如果 R1,R2 相交,则只需满足 c<=X∧X<=b。

不难看出,利用区间之间的包含,相交和相等关系可以使得查询公式化简并且在语法级上存在简单的化简算法。

把本节讨论的结果扩充到其它线性序集上是没有困难的。

结束语 我们采用本文所述算法实现一个数据库引擎,采用此引擎实现了一个数据库系统并命名为 eBASE。

考虑 xBASE 是目前应用最广泛的数据库系统,eBASE 同 xBASE 数据一级兼容,因此 eBASE 属于 xBASE 家族。

eBASE 同 C 语言是无缝耦合的,是一个全 C 语言数据库系统。由于许多系统都提供 C 语言接口,因此这意味着 eBASE 可以同这些系统实现无缝耦合。

由于在 eBASE 的实现中采用本文所述优化算法,因此它的内核很小,仅70-80KB 左右。但它的逻辑查询速度较快,尤其是在模糊查询方面。

实验还证明此系统在网络环境下性能特别优良。

目前 eBASE 已经实现了网络和 WINDOWS 版本,并且实现了同 Visual BASIC,AUTO_CAD 等系统的连接。同时,我们已经采用此系统为国内外客户实现了十余个应用系统,取得了良好的效果。

由于 eBASE 完全内嵌到了 PROLOG 系统中,因此 PROLOG 程序员完全可以通过 PROLOG 程序透明地访问外部数据库中的数据,这给 PROLOG 程序员带来极大的方便。目前此系统已在加拿大 NT 的合作项目中采用,效果良好。

参 考 文 献

[1] D. Ullman, "Principle of Database and Knowledge-base Systems", Computer Science Press, 1988

[2] J. W. Lloyd, "Foundation of logic Programming", Springer-Verlag, 1984

[3] S. Ceri, et al., "Logic Programming and Database", Springer-Verlag, 1990

[4] J. Bocca, "On the evaluation strategy of Educe", Proc. ACM Singmod, Washington, May 1986

[5] S. Ceri, et al., "Interface relational database and PROLOG efficiency", raporto interno No 85-23

[6] 李磊,等,"PROLOG-DBMS 实现中的子句间优化技术",软件学报,1995年第三期

[7] 周龙骧,"数据库实现技术"中国地质出版社,1990

[8] 李磊,"PROLOG—DBMS 实现方法",计算机学报,92年第三期

[9] 施伯乐,"KBASE-P:一个知识库程序设计",软件学报,1995 07 544-550

(上接第69页)

结束语 本文介绍了 PSOLA 算法的原理,给出了 TDPSOLA 实现的基本步骤,计算基音曲线和标注基音标记方法,虽然 TDPSOLA 方法可以改变语音的某些超音段性质,但是音节单元的性质在不同的自然语流中有很大的不同。我们采取了针对音节在词组和句子的位置信息,对常用音节再分类,增加音库中音节单元的个数。在抽取参考词组时,考虑该词组在语料中出现的频次,应用高频词组作为参考词组。这样处理,系统的合成质量在概率角度就得到了一定的提高。

参 考 文 献

[1] 张家录、吕士南等,"汉语文语转换的研究",信号处理,1989年第一期

[2] 倪宏、李昌立、莫福源等,"汉语大词汇量合成系统的研究",第六届全国语音图像通信信号处理学术论文会议集,1993

[3] 初敏,"高清晰度高自然度汉语文语转换系统的研究",中国科学院声学研究所博士论文,1995

[4] Cai Lianhong, Zhao Qiaofeng, Wang Yong, "Research of Prosody Modification Based on PSOLA in Chinese TTS", 1995

[5] 杨顺安,"语音合成与语音学研究",中文信息处理,1992

[6] Robert Edward Donovan, "Trainable Speech Synthesis", The Dissertation of the University of Cambridge, 1996

[7] 杨行峻、迟惠生,《语音信号数字处理》,电子工业出版社,1995