计算机科学1997 Vol.24 №.1

关系对象数据库管理系统 POSTGRES

昌月楼 杨 利
（国防科技大学计算机学院 长沙410073）

摘 要 本文叙述了关系对象数据库管理系统 POSTGRES 的历史、现状、及其特点，并指出了它存在的问题。

关键词 数据库，面向对象，数据库管理系统，继承性。

关系模型数据库(RDB)具有许多众所周知的优点，在数据库行业中应用逾来逾广且常盛不衰。然而，随着应用面的不断拓宽和应用需求的复杂化，诸如 MM、CAD、GIS 之类典型的复杂应用使得单纯以 Table 形式表现和处理数据的 RDB 在管理复杂对象时显得有些笨拙或无能为力。OOL 的出现启发业界人士另辟溪径，出现了为数不多的 OODBMS，如 ONTOS、O_2等。然而，纯 OODB 模型的数据库在广泛的应用领域中又显得不直观。象 MIS、POS、OA 等领域中人们对 Table 的"感情"还是深一些。这样，在兼顾广泛应用和特殊应用方面业界人士左右为难。

从头开始研制一个新的完善的 OODBMS 是很难的。首先是 OO 模型缺乏象关系模型那样坚实的理论基础，而且应用不很广，有些"费力不讨好"之憾。于是，有些人又提出一个思路：在底层仍使用 RDBMS，在应用上使用 OOL，例如，C++。这种办法是可行的，且不费力，但它是一种"治表不治里"的权宜之计。比较好的办法是在原 RDBMS 的基础上增加一些 OO 功能，如继承性、多态性、属性非原子性等。这样一种"两栖"DBMS 既能兼顾 RDB 和 OODB 的优点，又能减轻应用开发人员的负担。Postgres 就是这样一个具有 RDB 和 OODB 功能的关系对象数据库管理系统。

1 Postgres 的发展历史及现状

Postgres 是由 M. Stonebraker 教授领导的小组在 U. C. at Berkeley 研制成功的数据库原形系统，得到 DARPA、ARO、NSF 以及 ELS 公司的赞助。该系统于1986年开始实现。1989年第一版问世。该小组在征求了各方意见后，重新设计了其中的规则系统，次年推出第二版。第三版出现于1991年，它增加了对多存储管理器的支持，改进了查询执行器。在第4.2版中，该小组主要在系统的可移植性和可靠性方面作了努力。

Postgres 的最后版本是 Postgres95，是在 V4.2 的基础上发展而来的，它的改进主要体现在：用以 SQL-92为基础的 SQL 取代了原来的 POSTQUEL；源代码全部使用 ANSI C，从而代码量减少1/4；一系列的性能改进使它的运行速度比 V4.2提高30-50%（基于 Wisconsin Benchmark）；增加了基于 Tcl 的用户接口等等。

Postgres 出现以后在许多研究部门和生产单位得到了应用，诸如财务数据分析系统，喷气式引擎性能监测包、行星跟踪数据库、药品信息系统、地理信息系统；在教学中的应用更是屡见不鲜。Postgres 的商品化由 Illustra Information Technology 公司(已被 Informix 公司收购)完成。

下面是 Postgres 能运行的工作平台。

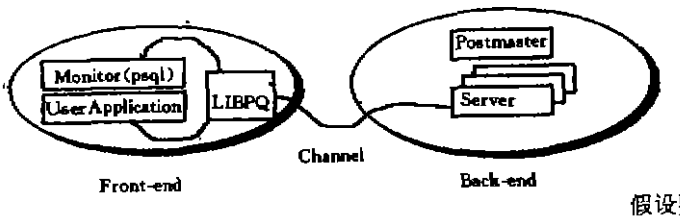
硬件	操作系统
DECstation 3000 (Alpha AXP)	OSF/1.2.1.3.0
DECstation 5000(MIPS)	ULTRIX4.4
Sun 4(SPARC)	Sun OS 4.1.3,4.1.3.UI, Solaris2.4
HP 9000/700, 800 (PA-RISC)	HP-UX9.00,9.01,9.03
Intel(X86)	Linux1.2.8ELF
IBM RS/6000	AIX3.2.5
Intel(X86)	NetBSD1.0

此外，还有一些平台的支持正在开发中，例如 Intel X86的 NT，SGI 的 IRIX5.0。

2 Postgres 的软件结构

Postgres 是一个 Client/Server 方式的 DBMS，其 C/S 通讯用的是 TCP/IP 协议，用户在前台既可

以通过 Monitor(或 psql)交互式地访问后台 Server，也可以使用 C 程序通过其 API(LIBPQ)嵌入式访问后台，其软件结构大致如下：



其中，Channel 首先由 Postmaster 建立，LIBPQ 与 Server 连接上以后 Postmaster“脱钩”，LIBPQ 与 Server 直接通讯。Postmaster 相当于 Deamon 进程。LIBPQ 允许一个应用程序连接多个 Server。

3 Postgres 的主要特点

3.1 Postgres SQL 的特点

Postgres SQL 除了一般 RDB SQL 的功能以外，还具有其本身的特点，它包括继承性、时序性和属性非原子性。

3.1.1 继承性 Postgres 引入 OO 概念，将一个 Table 视为一个 class，将 Table 中的每一个 row 视为一个 instance。class 具有继承性。在 Postgres SQL 中，可用如下方式定义继承性：

```
CREATE TABLE tbname(attr1 type1{,attri typei …})
    INHERITS(tbname1{,tbnamei…}).
```

定义继承性以后，查询要求既可以针对某一 class，也可以针对某一 class 及其后代。下面是关于继承性的一个例子(其中 text 是 Postgres 特有的数据类型)。

```
CREATE TABLE empl(no char, name text, age int,salary float);
CREATE TABLE mngr(position text) INHERITS (empl);
```

上面，class mngr 继承了 class empl 的属性，加上它自己的属性共5个属性。在 Postgres SQL 中，对一个 class 的查询或更新若要包括其后代，只要在它的后面加“*”就行了。例如：

```
SELECT * FROM empl* WHERE age>30;
```

查询结果包括了 empl 和 mngr 两个 class 中满足条件的所有 instance。

3.1.2 时序性 Postgres 具有时序概念。例如，若要查询“刘三本”在时间段[t1,t2]内的工资历史，可用如下句子表达：

```
SELECT name, salary FROM `empl* [t1, t2] WHERE name=“刘三本”;
```

['epoch', 'now'](或[,])表示系统初始时间至当前。对于 UNIX 系统，系统初始时间为1970.1.10:00:00GMT。

3.1.3 属性非原子性 RDB 的1NF 就是指属性的原子性限制。这一限制给处理一些复杂问题带来诸多不便。Postgres 突破了这一限制，它的属性可以是数组类型或集合类型。下面以数组类型为例加以说明。

假设要记录学生4年中各科成绩，在 Postgres 中建立该数据库就很简洁了。

```
CREATE TABLE std_score(no char, name text,
    score int[ ][ ]);
```

若用 score[1:1]、score[1:2]、score[1:3]…分别表示学生第一年的语文、数学、外语…成绩，余此类推，要查询第二年的数学成绩有进步的学生时，查询语句也很简单。

```
SELECT * FROM std_score WHERE score[2:2]>
  score[1:2];
```

插入语句可以表示如下(举例)：

```
INSERTINTO std-score (no, name, {score[1:1],
  score[1:2],score[1:3]})
  VALUE('9560101', '武力'{78,82,89});
```

3.2 Postgres SQL 的可扩充性

可扩充性基于 Postgres 的丰富的 DD 内容以及它的动态装载功能。Postgres 的 DD 中除了通常 DD 中的内容以外还包括类型、函数以及存取方法等信息；动态装载是指系统运行时按需要自动将事先指定的目标码文件(.o 文件或共享文件)装载到 Server 中。目标码文件中包括 Postgres SQL 扩充功能。在 Postgres 中，老练的用户可以扩充函数、类型、运算符和聚类。

3.2.1 定义函数 在 Postgres 中用户可用 SQL 语言或 C 语言定义新函数，函数的参数可以是基本类型或复合类型。基本类型指系统固有类型或用户定义类型，用户创建的一个 class 就是一个复合类型。Postgres 定义函数的语法如下：

```
CREATE FUNCTION func_name ([type1{, typei …}])RETURN type_r
    AS {'objectfilename' | 'sql-quries'} LANGUAGE{'c'|'sql'};
```

举例(在 Postgres SQL 中，SELECT 语句可以不带 FROM 子句)。

··用 SQL 语言定义基于复合类型(例如 empl)的函数，该函数计算 empl 中职员工资使之翻倍。

```
CREATE FUNCTION double_salary (empl) RETURN int
    AS 'SELECT' $.salary* 2 AS salary;' LANGUAGE 'sql';
```

引用：SELECT name，double_salary(empl)AS dream FROM empl WHERE empl.age>30；

··用C语言定义基于基本类型的函数，该函数用第二个串替换第一个串中的一部分，从某位置开始，假设C语言目标文件为sample1.0，则该函数定义如下：

```
CREATE FUNCTION replace(char,char,int)RETURN char
  AS '/usr/local/obj/sampleI.0' LANGUAGE 'c';
```

引用：newstring=replace('old string','new',I)

3.2.2 定义类型 定义类型之前必须先定义相关函数。最简单的定义类型的语法如下：

```
CREATE TYPE type_name(
  internallength={number|variable},
  input=input_function,
  output=output_function);
```

其中，internallength是被定义类型的内部长度，函数input_function将被定义类型从外部(字符串)形式转换成系统的内部形式；函数output_function则反之。例如，要将(x,y)定义成复数时可用如下步骤完成(假设cmplxin.c和cmplxout.c分别为输入输出函数)：

```
CREATE FUNCTION complex_in(opaque)RETURN complex
  AS '/usr/local/obj/cmplxin.o' LANGUAGE 'c';
CREATE FUNCTION complex_out(opaque)RETURN opaque
  AS '/usr/local/obj/cmplxout.o' LANGUAGE 'c';
CREATE TYPE complex(internallength=16,input=complex_in,output=complex_out);
```

其中，opaque是Postgres特有类型，此后，complex就可以作为复数类型定义别的变量了。

3.2.3 定义运算符 Postgres的这种功能类似于OOL的运算符重载，支持左元运算符、右元运算符和二元运算符的定义。最简单的二元运算符定义的语法如下：

```
CREATE OPERATOR operator_name(
  leftarg=typeI,
  rightarg=type2,
  procedure=func_name,
  commutator=com_op);
```

其中，若二元运算符是不可交换的，则commutator一项可省。与定义类型一样，也必须先定义函数。假定cmplxadd.c是实际操作的函数，于是下面的语句可用来定义复数加法操作符：

```
CREATE FUNCTION complex_add(complex,complex)RETURN complex
  AS '/usr/local/obj/cmplxadd.o' LANGUAGE 'c';
CREATE OPERATOR+(
  leftarg=complex,rightarg=complex,
  procedure=complex_add,commutator=+);
```

这样，"+"号就具有复数相加的意义了。

3.2.4 定义聚类 Postgres SQL定义聚类的语法如下：

```
CREATE AGGREGATE agg_name[AS](
  [sfunc 1=state_transition_functionI,
  basetype=date_type,
  stypel=sfuncl_return_type]
  [,sfunc2=state_transition_function2,
  stype2=sfunc2-return type]
  [,finalfunc=final_function]
  [,initcondl=initial_condition1]
  [,initcond2=initial_condition2]);
```

聚类的定义是基于状态转换的。sfuncl在状态转换(元组的增减)时跟踪某一属性值的变化；而sfunc2只跟踪状态内部变化，不考虑具体属性值。initcondl和initcond2分别为这两个函数的初始条件。下面是求平均值的一个例子('--'为注解标志)。

```
CREATE AGGREGATE my_average(
  sfuncl=intpl,--求和
  basetype=int,stypel=int,
  sfunc2=intinc,--计数
  stype2=int,
  finalfunc=intdiv,--除法
  initcondl='0',initcond2='0')
```

上面三个模块可以任意编制，从而可以定义各种不同的求均值函数。

3.3 其它特点

下面的特点涉及到更深的问题，限于篇幅不作细述。

A)Postgres还可以将二级索引(B-tree,R-tree,hash)引进上述新定义的类型或运算符中，从而提高系统性能。这牵涉到对DD的访问和修改。

B)Postgres引进了大对象功能，从而能处理多媒体数据。一般数据的元组是不能跨页的，一页仅有8,192字节，所以若不引进大对象功能，则不能处理多媒体信息。在大对象接口中，对用户数据的访问是基于UNIX文件的。Postgres的API提供了许多访问和处理大对象的函数，诸如打开、定位、读、写、关闭等。

C)Postgres引进了R-tree索引和提供了特殊数据类型(如polygon)，从而可以用来描述空间数据，支持GIS应用。

结束语 Postgres95引入了class概念，提供了class的继承性、属性非原子性和运算符重载功能，这些都是OODBMS所具有的；同时它又保持了

计算机科学1997Vol.24№.1

采用软件构件技术开发领域应用软件*

耿刚勇　　仲萃豪
（中科院软件所　北京 100080）　（中科院软件所）　（国家软件工程中心）

摘　要　In the procedure of developing tobacco MIS, The idea of reuse which run through the whole process of component development is used. This paper gives some rules of component drawing and making. A mechanism for component designing, integrating and management is given. This method can be used to develop software of other domain.

关键词　Reuse, Component library, Software bus, Domain

烟草行业是属国家专卖的一个特殊领域，国家烟草专卖总局决定在此行业推广使用 MIS 系统，开发领域的应用软件。由于各省烟草公司之间及下设烟草分公司和烟草厂商之间在管理上有一定的共性，但也有其特殊性，管理体制和经营方式不尽相同，所以使用同一 MIS 是行不通的，为此，我们采用群体构件库开发技术，针对其具体操作方式，用已有构件集成一个系统。这样就可以解决应用软件开发效率低、重复劳动多、周期长、适应性差等长期困扰开发人员的难题。

重用的思想在计算机软件领域得到了最广泛的公认，这一事实揭示了未来应用软件的开发趋势，"八五"时期我们已提出过一套开发方法，并取得了良好的效果[1,2]。近年，国际上兴起了可重用和可互操作的构件软件研究的高潮，更向软件提供商昭示了一个对支持重用的开发环境和方法及其辅助工具渴求的巨大市场，世界上最负盛名的计算机软件公司纷纷加入到这场角逐中。

国际上象 CORBA[4]，OpenDoc，OLE2等标准和技术以及 PowerBuilder，Delphi 等界面构件开发工具的出现，主要是解决编程阶段的一些可重用问题。现在软件工程研究领域一个重要课题是研制出大众化的实用的领域构件化软件开发方法和技术。软件工程作为一门工程的学问，首要任务是使软件开发越来越工程化，普及化，使众多的"工匠"能迅速高效地垒筑软件系统。软件工程的价值在于应用。只要成功地支撑应用开发才能延续和发挥其内涵价值。本文利用国际上的成果和我们的开发经验，确定出针对领域构件应用开发的方法、原理、模型和机制。

1　构件概念模型

1.1　构件的定义

软件构件是可重用的软件单元，可以被用来构造其它软件。它可以是被封装的对象类、一些功能模

RDBMS 的所有功能。所以可以说它是一个关系对象数据库管理系统。它的丰富的 DD 内容和灵活的 DD 结构使它具有很好的可扩充性，包括定义函数、类型、运算符和聚类。这些特点是许多 DBMS 所没有的。它的 Client/Server 结构、大对象接口以及空间数据描述和处理功能正符合目前迅速发展的信息系统对 DBMS 的要求。目前，Postgres95不支持子查询，这是它的美中不足。尽管这一功能可以通过用户定义函数来实现，但毕竟不直观。另一缺点是，由于它未广泛商品化，所以前端工具不丰富。Postgres95是一个很有生命力的 DBMS，它的进一步完善和商品化是指日可待的。

主要参考

[1]Andrew Yu et al., The POSTGRES95 User Manual, Beta Release0.03(7/19/95)
[2]L. Rowe et al., The POSTGRES Data Model, Proc. 1987 VLDB Conf., Brighton,England,Sept., 1987
[3]L. Rowe et al., The Design of POSTGRES, Proc. 1986 ACM-SIGMOD Conf. on Management of Data, Washington,DC,May,1986
[4]M. Stonebraker et al., The Implementation of POSTGRES, IEEE Trans. on Knowledge and Data Engineering, 2(1), March, 1990
[5]A. Davison, Object Oriented Databases in Polka, UNICOM, Applied Information Technology, 13, 1992

*)本文得到国家软件工程重点项目，国家基金和北京基金的支持。