Application & Research of Post-relational Database Caché

后关系型数据库Caché的应用研究

张金辉　刘仲英　（同济大学 经济与管理学院　200092）

摘要：本文首先简要介绍Caché的技术背景和概况；而后说明Caché的三大特色：优异的数据表现力、对象化的数据库、集成的WEB应用；最后将给出一个应用实例。

关键词：面向对象　数据库　后关系型数据库　Caché

1 概述

Caché是由Intersystems公司开发的一款后关系型数据库。它所基于的层次型数据库技术Mu mps (Massachusetts General Hospital Utility Multi-Programming System)在上世纪60年代发源于医学领域。Mumps既是一种高级程序设计语言，跟C及Pascal语言一样古老、一样优秀；又是一个高效灵活的数据库系统。由于层次型数据库技术的固有特点以及Mumps技术特有的字符串下标技术，使得其在表达复杂结构的信息方面有着异常灵活、高效的性能优势。正是由于其具有独特性能，因而一直在美国和其他国家及地区得到重视和应用，尤其在发展HIS医院信息系统上，扮演了举足轻重的关键角色。MUMPS技术多年来得到不断丰富和发展，不仅被定为美国国家标准，而且从1992年起被批准为ISO国际标准。

Caché还有一个亮点就是集成了WEB应用。因为Caché的语言不仅是数据的操纵语言，它也是应用程序的设计语言。开发人员可以在数据库环境内直接编写WEB应用程序。由于WEB应用程序可以直接操纵数据库内数据，开发基于BS架构的数据库应用系统非常灵活方便。

2 Mumps的特性

2.1 Mumps优异的数据表现力

在MUMPS中，数据是被保存到GLOBAL中的，类似于关系型数据库中的表。GLOBAL也是数组的一种，但它对应在MUMPS管理的磁盘空间中，因此它的数据可以永久保存，并且所有Mumps进程都可以访问。由此可见比一般意义的全局内存变量更具“全局性”，因此把它称为磁盘变量，以区别于保存在内存中的数组变量。磁盘变量的操作与功能和内存变量完全一样，仅仅是被保存的位置以及永久性不一样而已。磁盘变量用"^"标记出来，如上文的a是内存变量，将a换成^a则为磁盘变量，上面的数据将被保存到磁盘中，形成记录。因此Mumps技术能很容易在磁盘中构造出各种类型的数据结构。如：堆栈、队列、链表、二叉树、二维表、图以及对象结构的多维记录等，而且操作也非常灵活。

2.2 扩展Mumps语言独特的对象化技术

数据结构与数据行为封装在一起才能构成对象。Mumps数据库的树型多维数组虽然可以轻松构造各种数据结构，但却无法与相对独立的Mumps语言封装在一起。因此Caché对Mumps语言进行了扩展，构建面向对象的Mumps语言，称之为Caché Object Script。在Caché Object Script中可以定义类，构建对象，封装数据结构与数据行为形成实体概念。考虑到Caché Object Script与标准Mumps语言的兼容和可移植性，Caché Object Script写的程序不能直接运行，必须通过Caché编译器将其编译成标准Mumps语言，而后才能被执行。反过来就是说标准Mumps语言可以通过适当的宏构造来实现对象化技术，由此可见Mumps语言的强大表达能力。

Caché编译器还让Caché Object Script更靠近自然语言，与常见的高级程序设计语言风格一致。比如，由于Mumps是解释型程序语言，为了提高数据库系统执行效率，对语法有着非常严格的约束要求，多一个空格，少一个TAB都会引起程序的出错与挂起，现在编译器在编译时可以在Caché Object Script中识别，并兼容这些难免的人为失误，转化成严格语法约束的标准Mumps程序。在Mumps中基本上没有数据类型，所有数据都是字符串。而在Caché Object Script中则可以通过适当的标记让编译器插入宏验证来实现一些基本的数据类

型。当然，而后可用对象与类来构造更复杂的数据类型，封装数据行为，就如同C++一样。

然而如果需要超越进程空间的限制，将数据存储到磁盘中去，供其他进程共享，也就是数据库技术的基本功能，一般的面向对象程序设计语言就要颇费周折了，不是打开设备/文件，就是与数据库服务器连接，另外用数据操作语言操纵数据库服务器中的数据。Caché则不同，其自身就是一个高效的数据库系统，Caché Object Script可以直接操纵该数据库中的数据。并且Caché利用类和对象模拟关系型数据库的关系表，实现对数据库中数据的对象化操作。下面是一个简化了结构的例子。

树型多维数组构造的关系表与索引：

```
^PersonD(1)= "S001^张三^28^M"
^PersonD(2)= "S002^李四^24^M"
^PersonD(3)= "S003^赵六^22^F"
^PersonI("Name", "张三",1)= ""
^PersonI("Name", "李四",2)= ""
^PersonI("Name", "赵六",3)= ""
^PersonI("Sex", "M",1)= ""
^PersonI("Sex", "M",2)= ""
^PersonI("Sex", "F",3)= ""
```

其中^PersonD是表的磁盘变量，^PersonI是索引的磁盘变量，索引有两个，分别是Name和Sex。表的磁盘变量第一个下标为记录号，数组元素内容被"^"分割成4片，作为四个字段。该数据结构被Caché Object Script代码封装，形成一个与一般关系型数据库中的表(Table)类似的对象化结构，在Caché中被定义为Persistent类的派生类，不妨称之为P型类。该类不但包含有各种方法函数，如：%New()、%Delete()、%Save()等，以及被定义为属性变量的字段定义，其还在磁盘内保存有表中的数据。因此该类不只是包含表结构，而是包含一个完整的表。当然它还含有各种方法函数，比通常的表功能要强大很多。基于P型类所创建的对象包含一个内存变量结构，保存有一条该表的记录，也就是当前记录。可以视之为是一个带有对记录操作方法集的游标对象，该对象的属性值对应着当前记录的字段值。

至此还不够。为与一般的RDBMS系统兼容。Caché内还构造了SQL Gate Way，将P型类映射成SQL可以操纵的关系表，提供用户基于SQL语言的数据操纵界面。并且还集成了JDBC和ODBC接口，方便JAVA或微软程序员编写的客户端程序与该数据库连接，也方便大多数RDBMS系统与该数据库交互数据。

更进一步，Caché还集成了CORBA类，使得Caché中的方法函数能够被远程跨平台地调用。这样，CORBA接口函数灵活而完全地封装了数据库中的数据及其行为，实现了数据库系统灵活性与安全性的双赢。

3 集成WEB应用方便地实现BS架构

Caché数据库系统开发平台主要支持的就是基于BS架构的WEB应用。在Caché中集成有WEB应用的引擎，它接受WEB服务器转递的浏览器请求，调用对应的脚本程序，生成WEB页面返回给WEB服务器再转达给用户的浏览器。Caché甚至还内建有一个功能相对简单的WEB服务器，用户浏览器可以直接通过1972端口访问该内建的WEB服务器，获得脚本程序生成的WEB页面。在Caché平台中，Caché Object Script自然就被用作WEB服务器端的脚本语言。最基本的就是用Caché Object Script语言直接操作，输出HTML文本到浏览器。不过这样实现WEB应用显然过于烦琐，开发基于WEB的应用程序不太方便。因此Caché向用户提供了另外一条更便捷的途径--Caché Server Page，即CSP。

如同ASP一样，CSP定义了一套为数不多的特有标记，如：<server>、<csp:>等，通过它们将Caché Object Script脚本或宏嵌入到HTML文本中去形成CSP页面文档(*.csp)，即Caché的动态服务器主页。因而CSP页面可以通过脚本，很灵活地操控数据库中数据。不过

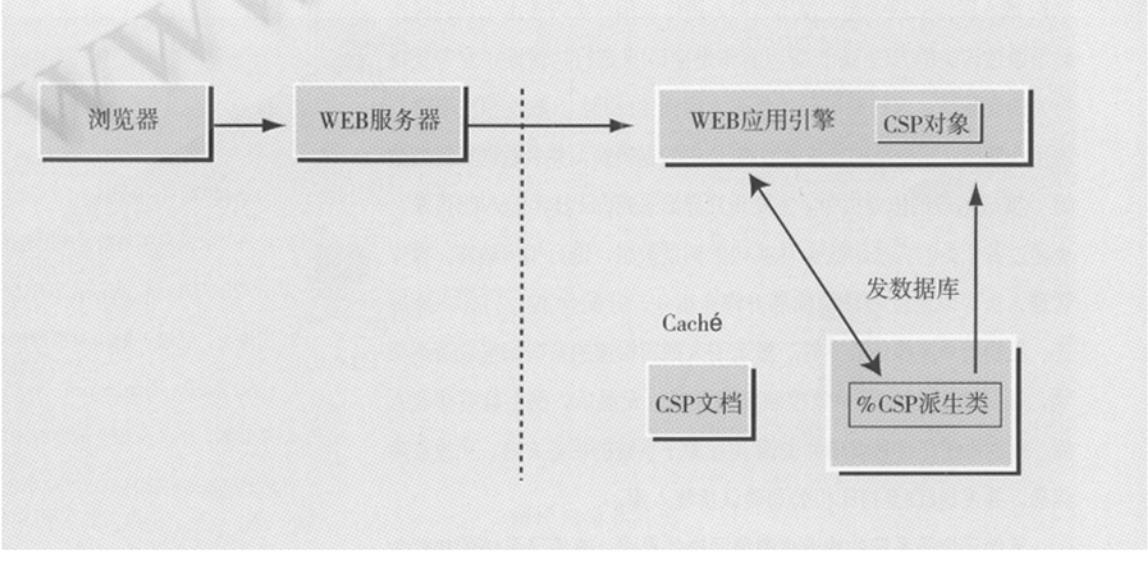

图1 CSP页面的运作示意图

Caché并不会直接解释执行CSP页面，而是将其融进Caché ObjectScript的对象中，为每一个CSP页面文档编译生成一个与之对应的%CSP派生类，该类中将会包含有生成相应HTML文本的方法函数。当浏览器发出CSP页面请求时，引擎就会识别所请求的页面，找到其对应的类创建对象，而后用事件驱动对象中相应的方法函数，生成页面回送浏览器。如图1所示。

更有魅力的是Caché将APPLET的远程调用技术揉进了其对象化的CSP页面。作为%CSP的派生类，CSP页面可以用标记<script language=cache runat=server method=Fun1 ...>来定义其自己的方法。而Caché在编译时，会自动在该方法的调用处插入JavaScript的APPLET例程，使得该方法很容易地被HTML的客户端脚本所调用。这很大程度上减轻了程序员的负担，程序员可以不必去关心APPLET，不懂APPLET也没问题。由于不提交页面，浏览器端就可以和服务器端通过远程调用交互数据，这大大减轻了浏览器和服务器之间的数据传输负荷，非常适合开发基于BS架构的事务处理系统。

4 典型例子--某品牌打印机配送仓库作业管理系统

该配送仓库共有五层，约20,000平方米，是个区域性的大型打印机配送中心。其中3到5层为库房，每层约有1,500个货架分别存储有各种型号的打印机及其配件。二楼是分检打包中心和配送作业管理中心，接受送货指令，配齐货物打成货包。底层则是收货、出货及退货的堆场和装卸作业台。因此该系统主要被分为四个相对独立又紧密协作的子系统：入库、在库、配货与出库。

我们详细考察一下入库子系统。首先入库作业指令被表示成树型数组的多维记录，免去了关系型数据库大量关系表的连接操作，而直接操纵数据，大大提高了数据库的操作效率，也使得WEB有良好的响应性能，很适合此类事务处理系统的要求。而后用对象封装了多维记录构成入库作业指令类，使所有对入库作业指令类内多维记录的操作必须通过该类的方法(或也可以看作是接口)来进行，保证了业务逻辑行为的完整性。当打印机公司发来入库指令EDI时，触发相应事件，调用入库作业指令类的方法函数将EDI中数据转换，并装载进其树型数组，变成对应的作业指令。主要用户界面采用CSP技术的WEB页面，也通过各个类的方法函数操纵类内的树型数组，进行相应动作。操作管理人员可以通过WEB页面展开作业指令，分配仓位，打印入库标签、货物清单及操作指令等。搬运工人则用相连的条码扫描仪清点货物，在系统通过内部对象控制条码仪确认无误后，贴上标签接收入库。同时系统还会触发事件去调用在库子系统的相应方法，更改在库信息，并发送EDI至打印机公司确认货物入库。

其他三个子系统的构造也遵循同样的思路。在库子系统的功能主要包括：接受入库、分检打包及出库子系统的调用请求，修改库存和在库状态；监视当前实际库存情况，查询并跟踪库存货物信息与仓位状况；当然打印机公司也可通过特定的WEB访问，远程查询其在库的信息；库存盘点后，进行必要的信息更改，以保证其真实有效地反映库存情况。配货子系统接收打印机公司发来EDI配送指令，协助操作管理人员分配库存货物，生成包括分检、打包等在内的操作指令，打印相关文档与标签，并用对象技术控制电子作业指示牌的显示。操作工人则根据指令和电子作业指示牌，在库内取出打印机和配件，在包装作业台打成包裹，贴上标签，后传送往底层货物堆场。途中经自动条码扫描仪确认后，系统也调用在库子系统的相应方法，更改在库状态信息，同时生成出库指令，发往出库子系统。出库子系统则根据出库指令将包裹拼装成车，用动态CSP页面在大型电子屏中滚动显示装车指示，同时打印装车指令和送货单据。搬运工人用扫描仪核对货物及其标签，并将货物搬运入车内，经司机确认无误后发车。同时出库子系统调用在库子系统相应方法，更改在库信息，并发送EDI给打印机公司通知配送作业完成。至此完成该系统的一个业务流程。

5 结束语

Caché利用Mumps技术可以近乎于底层地，直接操纵数据库数据；而用对象化技术又可以根据业务逻辑需要封装数据库中的数据。这为程序员开发高效、稳健的事务处理系统提供了最大可能。Caché的CSP技术也尽可能地通过自身的集成，并用APPLET优化了页面的响应性能。在上面的配送作业管理系统中CSP页面就完全满足了性能要求。如果有更高的性能要求，可以采用CORBA技术连接客户端应用界面。无论如何，在用关系型数据库构建系统感到“山穷水复”的时候，试试Caché或许就会有“柳暗花明”的意外收获。

参考文献

1 Intersystems Corporation,《Caché Tutorials》, 21 August 2000。

2 於丹，吹响后关系数据库时代的号角，http://www.pcworld.com.cn/2000/back_issues/ 2012/1231.asp。

3 王继中，稳居于Internet医疗信息网络前沿的Caché e-DBMS，http://www.cmia.org.cn /xsyj5/5qi/wenju.htm。

4 马文赛，Mumps以空间换时间，http://www.ccidnet.com/tech/guide/2001/05/09/58_21 13.html。