第 30 卷 增刊　　电 网 技 术　　Vol.30 Supplement
2006 年 10 月　　Power System Technology　　Oct. 2006

文章编号：1000-3673（2006）S2-0171-05　中图分类号：TM73　文献标识码：A　学科代码：470·40

开放实时数据库及其在调度自动化系统中的应用

周 宁，丁 琦

（重庆市电力调度（交易）中心 重庆市 400014）

Open Real-time Database and it's Application in Dispatching Automation Systems

ZHOU Ning, DING Qi

（Chongqing Power Dispatch Center,Chongqing 400014, China）

ABSTRACT: Traditional databases such as ORACLE, SQL SERVER have powerful data processing functions, and become a standard component of automation systems. However, such database use hard disk as its storage media with low speed. Most dispatching automation systems use both real-time and traditional databases to achieve high speed. Today, most real-time databases are developed by automation system providers, the interfaces are non-standard. So, it's hard to develop and data interchange. Especially, function of data synchronization among different nodes in a automation system is also developed by the providers, which are not reliable, this may reduce the stability of the systems. This article lay out a scheme of open real-time database based on memory. The database supports standard SQL language and data replication among different nodes. It also supports transaction to make sure the data consistency. This article describes the implementation technology of data I/O from memory and the arithmetic of data synchronization between real-time and traditional database, and make real-time and traditional database cooperate together. Technology developed by this article will help to reduce the development investment, improve the system's reliability, make progress in the standardization of real-time data processing in dispatching automation systems and perhaps will be widely used.

KEY WORDS: real-time database; application; dispatching automation systems

摘要：传统的历史数据库如 ORACLE、SQL SERVER 等具有强大的数据处理能力，但这类数据库以硬盘为存取介质，速度较慢，难以满足实时数据处理的要求。调度自动化系统大多采用实时库和历史库相结合的方式，将实时性要求较高的任务交给实时库处理。目前的实时数据库大多由调度自动化系统厂商自行开发，采用各自的专用接口，这给系统的开发和实时数据共享造成困难。特别地，调度自动化系统各节点的实时数据同步功能也由厂商自行开发，可靠性低，从而影响自动化系统的稳定运行。本文提出一种开放的常驻内存实时数据库，可采用标准的 SQL 语言进行访问，并支持实时库之间以数据复制的方式进行同步。该实时库还支持事务，从而保证实时数据的一致性。本文描述了以内存为介质进行数据存取的实现技术，并提出了实时库与历史库之间进行同步的优化算法，从而把实时库与历史库有机结合起来。本文提出的技术有助于降低开发成本，提高系统可靠性，在调度自动化系统实时数据处理的标准化方面前进了一步，具有较大的推广价值。

关键词：实时数据库；应用；调度自动化系统

0 引言

传统的历史数据库如 ORACLE、SQL SERVER 等具有强大的数据处理能力，但这类数据库以硬盘为存取介质，速度较慢，难以满足实时数据处理的要求。调度自动化系统大多采用实时库和历史库相结合的方式，将实时性要求较高的任务交给实时库处理。目前的实时数据库大多由调度自动化系统厂商自行开发，采用各自的专用接口，这给系统的开发和实时数据共享造成困难。本文提出一种开放的常驻内存实时数据库，可采用标准的 SQL 语言进行访问。文章介绍了实时库的功能设计、体系结构、实现技术及其应用。

1 实时数据库的功能设计

实时数据库被设计成与历史数据库类似的系统，就功能而言，则实现历史数据库的一个子集，具体如下：

（1）关系数据库模型

在关系模型中，用户数据存储为表中的行（记录）。行被定义为一系列的列（数据字段），每个字段都有一个指定的数据类型。可以显式或隐式地创建索引，以便加快对表执行的基于键的搜索或值范

围搜索。在表中，可以指定由一列或多列组成的主键。主键是行的唯一标识符，还可以为每个表指定一个或多个外键，每个外键都与另一表的主键关联。外键建立两个表之间的父子关系。

物化视图是可以定义的可选结构。物化视图是源自常规（基）表的只读表。它可以包含一个或多个表中的部分或所有数据，并可以包含经计算得到的值（如总数和平均数）。

数据管理操作使用行业标准的结构化查询语言(SQL) 表示。主要操作包括 SELECT（读）、UPDATE、INSERT 和 DELETE。还支持关系联接操作（将多个表中的列组合为一个结果集），并且支持大多数在基准 SQL92 标准中定义的 SQL 选项。

（2）锁定和隔离

实时数据库使用锁来防止用户更改其他用户当前正在读取或更改的数据。锁是在数据存储区、表或行等级别置于数据存储区对象中的“保留地”。

（3）持久性和并发性

实时数据库应符合数据管理系统的“ACID”属性：原子性、一致性、孤立性和持久性。这些属性确保在多用户系统中，每个事务在运行时就好像此时只有它运行一样，并且系统可以确保不丧失已提交事务的效果。这些属性是数据管理器所需的最严格的属性。

（4）查询处理

大多数专门为获得高性能设计的产品都需要使用专有的、“隐蔽”的 API。实时数据库则不用这样，它的主要目标始终是采用相关的开放式标准。SQL（结构化查询语言）、JDBC（Java 数据库连接）、ODBC（开放式数据库连接）以及 JMS（Java 消息服务）等.从而简化应用软件开发。

（5）数据复制

所谓的复制就是在数据库之间复制数据。Oracle TimesTen 复制主要是为了使任务关键的应用程序能够在尽可能不影响性能的情况下连续使用数据。除了在故障恢复中的重要作用外，复制在多个数据库之间分配用户负载从而实现最高性能和简化在线升级和维护方面也很有用。

（6）高速缓存

在调度自动化系统中，实时数据库通常与基于磁盘的历史数据库部署在一起，当需要实时捕获或处理数据时，将使用实时数据库。当数据转换到非实时状态，将把信息从实时数据库传到历史数据库。

2 实时数据库的体系结构

本节讨论实时数据库的体系结构。通常，实时数据库包含以下组件：

（1）应用程序层共享库

实施 SQL 操作的例程以及相关函数包含在一组共享库中，开发人员将这些库链接到应用程序并作为应用程序进程的一部分执行。对于广泛使用的 WINDOWS 系统，应用程序层共享库以动态链结库（*.DLL）的形式存在。

（2）内存中数据结构

内存中数据库在操作系统的共享内存段中维护，它包含所有用户数据、索引、系统目录、日志缓冲区、锁表和临时空间。多个应用程序可以共享一个数据存储区，而单个程序可以访问同一系统上的多个数据存储区。

（3）系统进程

后台进程在系统级别为启动、关闭和应用程序故障检测提供服务，并在数据存储区级别为加载、检查点和死锁处理提供服务。

（4）管理程序

用户、脚本或应用程序显式调用实用程序来执行交互式 SQL、批量复制、备份/恢复、数据存储区移植和系统监视等服务。

（5）检查点和日志文件

如果需要恢复数据存储区，将把磁盘上的数据存储区检查点 与仍位于日志文件中的已完成事务合并在一起。检查点和日志文件使用普通的磁盘文件系统。在有限情况下，可以将实时数据库配置为无磁盘操作。

3 实时数据库的实现技术

实时数据库技术通过更改有关数据在运行时所在位置的假设，提供了引人注目的性能。通过在内存中管理所有数据 并针对该环境进行优化，内存中数据库技术可以更高效地发挥作用，从而显著显著提高响应性和吞吐量。

但这种性能的核心是什么？应用程序能否只通过将其所有 RDBMS 数据置于主内存中而获得同一结果？

历史数据库执行的大部分工作都是基于数据主要位于磁盘中这一假设完成的。优化算法、缓冲池管理以及索引化检索技术都有利于此基本假设。而实时数据库知道数据位于主内存中，因此可以更

直接地路由数据，从而降低了代码路径长度并简化了算法和结构。 通过在内存中管理所有数据并对该环境进行优化，内存中数据库技术可以更高效地发挥作用，从而显著提高了性能。

在比较这些体系结构时，有一些重要的差别清楚地说明了如何实现数倍的性能改进。下面就查询优化算法、缓冲池管理和索引结构等重要和关键性的差别进行阐述。

（1）查询优化

基于磁盘的系统与基于内存的系统使用不同的查询优化算法。历史数据库优化决策依赖于数据主要位于磁盘上这一假设。在动态的运行时环境中（数据在任何给定时刻都可能位于磁盘上或缓存在主内存中），基于磁盘的系统必须假设最差的情形。由于磁盘 I/O 的开销远远高于内存访问，因此基于磁盘的历史数据库必须假设数据位于磁盘上。它们针对减少性能瓶颈点（即磁盘 I/O）进行了优化。如果这就是所采用的假设，那么基于磁盘的优化器不会总为主要（或完全）位于主内存中的数据生成最佳计划。

而 IMDB 技术知道数据位于主内存中，并基于更简单的假设优化它的查询。它不需要根据磁盘驻留进行最差情形假设，因此它的开销估算可能会更简单并更准确。

（2）缓冲池管理

在常规的历史数据库体系结构中，必须为已经高速缓存到主内存中的数据维护高速缓冲池。当 SQL 查询处理器需要一页数据时，访问方法必须先在缓冲池中搜索该内存中数据。即使该数据位于缓冲池中，在许多情况下仍必须将它复制出缓冲池以便随后进行处理。此缓冲池的维护和管理以及额外的数据副本显著增加了将数据提供给应用程序的最初开销。尽管缓冲池在基于磁盘的数据管理解决方案中是必要的，但在基于内存的数据管理中却不是必要的。实时数据库没有缓冲池。由于数据已经位于主内存中，因此不需要缓冲池。这样便降低了代码路径的长度和引擎的内存需求，避免了复制，简化了算法并更迅速地将数据传输给应用程序。此外，历史数据库不会在运行时动态改变这些基于磁盘的假设。基于磁盘的假设在系统代码库中的交错过于紧密，以至于无法使用几个精心设计的 if-then-else 语句来简化它们。在以下两个体系结构的索引树结构中可以看到有关这个相互交错、基于磁盘的假设示例。

（3）索引结构

在典型的历史数据库的 B+ 树索引页中，键值和数据指针均保存在 B+ 树条目中。B+ 树节点（磁盘上的页）由许多 B+ 树条目组成，每个条目都保存索引数据值和下一个相应索引节点的页码或保存所搜索到的数据行的磁盘块页码。该结构使得树非常平和宽（非常适于减少磁盘 I/O）。在 B+ 树结构中，主要的目标是减少完成数据文件索引化查找所需的磁盘 I/O 数量。B+ 树实现此目标的方法是：(1) 将键值保存在 B+ 树节点本身中（减少磁盘 I/O），(2) 将尽可能多的索引条目保存到节点中（增加可以用单个 I/O 提供服务的 B+ 树条目数）。

该结构尽管对于磁盘上的数据和索引文件比较适合，但对于内存中的数据则不太适合。将数据保存到内存中以后，索引模式的目标是减少 CPU 周期而非 I/O。CPU 周期用于扩展和比较 B+ 树中的压缩索引值。大量的 CPU 周期还用来管理保存数据的缓冲区以及已经从磁盘读取到内存中的索引。

在实时数据库中，情况要简单一些。T 树针对主内存访问进行了优化。就大小和算法而言，它的索引条目比 B+ 树的更经济。T 树节点之间的连接通过 <= 和 >= 指针实现。这两个指针引用内存位置，而非磁盘上的页。仅仅通过这两个比较，T 树搜索算法便知道它正在搜索的值是位于当前节点上还是位于内存中其他位置了。每次使用新的索引节点指针间接运算符都会使搜索区域减半 – 体现了真正的二分搜索（如图 1）。前面的体系结构差别示例表明，消除磁盘驻留性假设（或数据位置的模糊性）将显著降低复杂性。计算机指令数至少降低十倍，缓冲池管理不复存在，不需要额外的数据副本，索引页减小，并且它们的结构得到简化。当数据的内存驻留性成为基本事实假设时，一切变得更简单、更简洁、更精巧、更快速。（4）数据复制

所谓的复制就是在数据库之间复制数据。实时数据库的复制主要是为了使任务关键的应用程序能够在尽可能不影响性能的情况下连续使用数据。除了在故障恢复中的重要作用外，复制在多个数据库之间分配用户负载从而实现最高性能和简化在线升级和维护方面也很有用。

实时数据库遵循“主服务器-用户服务器”复制

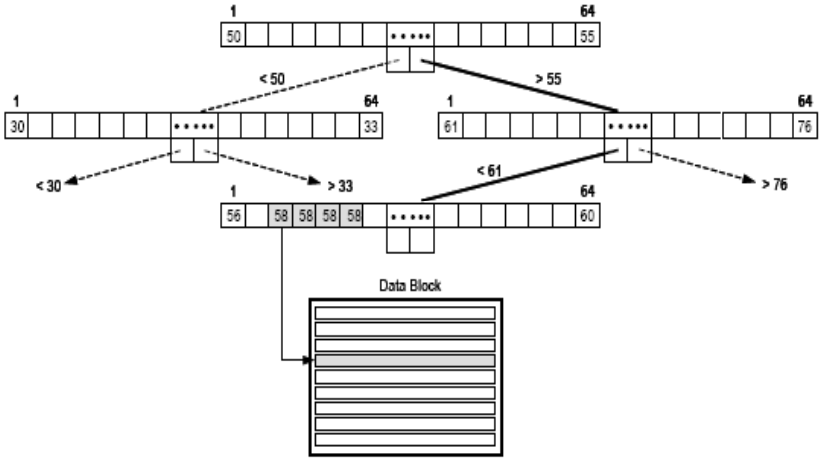

图 1 内存中数据存储区的 T 树索引

模型，即将提交的更改从它们的源复制到一个或多个用户服务器数据存储区。复制通过 SQL 语句进行配置，可以应用于指定的表或整个数据存储区。为提高效率并降低开销，实时数据库使用一个基于事务日志的复制模式。复制使用无磁盘日志记录或基于磁盘的日志记录。

每个主服务器数据存储区和用户服务器数据存储区上的复制都由通过 TCP/IP 流 socket 通信的复制代理控制。主服务器数据存储区上的复制代理从其事务日志中读取记录，并将检测

到的对复制元素的任何更改转发给用户服务器数据存储区的复制代理。然后，用户数据存储区的复制代理将把更新应用于其数据存储区。如果主服务器复制代理转发更新后，用户服务器复制代理并未运行，则主服务器复制代理将把更新一直保存在它的日志中，直到可以在用户服务器复制代理上应用这些更新。

默认情况下，实时数据库复制是一个异步机制。当使用异步复制时，一个应用程序将更新主服务器数据存储区并继续工作，而不等待用户服务器数据存储区接收更新。主服务器数据存储区和用户服务器数据存储区使用内部机制确认用户服务器数据存储区已经成功接收和提交更新。这些机制可以确保只在用户服务器数据存储区上应用一次，但它们对于应用程序是完全透明的。

通过将一个数据存储区同时指定为主服务器和用户服务器，可以配置双向复制此外，历史数据库还允许多节点“N 路”复制，从而提供了各种可能的复制拓扑，其中包双机热备份配置和负载均衡配置。后者还可以分成分割式负载（其中的每个复制表只有一个主服务器）和分布式负载（其中的每个表有多个主服务器）。在分布式负载配置中，应用程序负责在两个系统之间进行分工，以避免发生复制冲突。如果发生了冲突，则基于时间戳的冲突检测和解决机制可以防止出现不一致的复制。

（5）高速缓存

实时数据库通常与基于磁盘的历史数据库部署在一起，当需要实时捕获或处理数据时，将使用实时数据库。当数据转换到非实时状态（例如，当完成了股票交易或评估了呼叫详细记录的等级后），将把信息从实时数据库传到历史数据库。有多种方法可以实现此集成。

集成实时数据库数据存储区和历史数据库的一个简单、透明的方法是通过它内置的连接添加 Cache Connect to RDBMS。

与大多数只读的高速缓存不同，Cache Connect to RDBMS 支持对历史数据库数据的高速缓存进行读/写操作，并在实时数据库和历史数据库数据库之间双向传播更新。另一个不同之处是“高速缓存组”这一概念，它描述了一组与历史数据库中所有表或表子集相对应的 实时数据库表。一个高速缓存组可以包含这些历史数据库表中的所有行和列或行和列的一个子集。

Cache Connect to RDBMS 可以控制历史数据库数据在高速缓存中保存的时间。除了能够设置自动功能之外，可以通过 SQL 语句在需要时加载、刷新、清除和卸载高速缓存组。

Cache Connect to RDBMS 与历史数据库进行交互来执行所有同步高速缓存组操作，如创建高速缓存组、从历史数据库中加载高速缓存组以及在高

速缓存组与历史数据库之间传播更新。此外，还有一个称为历史数据库代理 的系统进程，该进程执行所有异步高速缓存操作，如将 历史数据库中的更新自动传播给高速缓存组。

Cache Connect to RDBMS 应用程序可以通过单一连接向高速缓存组或历史数据库发送 SQL 语句。此单一连接功能由 pass-through 特性启用，该特性检查高速缓存表能否在本地处理 SQL 语句或者是否必须将其重定向到历史数据库。可以在实时数据库高速缓存组或历史数据库 数据库中更新高速缓存的数据。Cache Connect to RDBMS 能够将更新从高速缓存组自动传播给历史数据库，并可以从历史数据库传播给高速缓存组。

4　结束语

随着电力调度系统传输的信息量不断增多，能否实时处理并智能地捕获、分析和响应实时数据和重要事件逐渐成为调度自动化系统的基本要求。电力用户希望调度自动化系统能够提供高度定制的交互和最快的响应。

所需要的是一代轻型基础架构软件，它应提供易于使用、功能强大的界面和广泛使用的查询语言— 可以轻松地与现有后端数据库、消息处理系统以及应用服务器进行通信，可以充分利用当前内存丰富的联网计算平台的全部潜在性能。这正是实时数据库的用武之地。

参考文献

[1] Hector Garcia-Molina,Jeffrey D.Ullman,Jennifer Widom ,"Database System Implementation", Prentice Hall,2000

[2] D.Comer,"the ubiquitous B-tree",Computing Surveys,1979

[3] R.Bayer and E.M.McCreight,"Organization and maintenance of large ordered indexes", Acta Infornatica,1972

[4] W.W Peterson, "Addressing for random access storage," IBM J.Research And Development

[5] E.F.Codd,"A retional model for shared data banks",Comm.ACM

[6] G.Graefe," Query evaluation techniques for large databases", Computing Surveys, 1993

[7] S.Chaudhuri," An overview of query optimization in retional databases", IBM Systems,1977

[8] M.W. Blasgen and K.P. Eswaran," Storge access in relational databases", IBM Systems,1977

[9] H.T. Chou and D.J.Dewitt, "An evaluation of buffer management startigies for relational database systems", Proc.Intl.Conf. on Very Large Databases,1985.

收稿日期：2006-09-08。

作者简介：

周宁，男，重庆电力调度通信中心高级工程师，毕业于重庆大学电力系统及其自动化专业，并获重庆大学计算机技术工程硕士学位，从事 DMIS 系统、调度数据网及安全防护等工作；

丁琦，男，重庆电力调度通信中心总工程师，毕业于成都科技大学电力系统及其自动化专业，从事电力调度生产管理，科技管理等工作。