**内存数据库**

## 一、内存数据库简介

**1 什么是内存[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)**  
传统的数据库管理系统把所有数据都放在磁盘上进行管理，所以称做磁盘数据库（DRDB:Disk-Resident Database）。磁盘数据库需要频繁地访问磁盘来进行数据的操作，由于对磁盘读写数据的操作一方面要进行磁头的机械移动，另一方面受到系统调用（通常通过CPU中断完成，受到CPU时钟周期的制约）时间的影响，当数据量很大，操作频繁且复杂时，就会暴露出很多问题。 近年来，内存容量不断提高，价格不断下跌，[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)已经可以支持更大的地址空间（计算机进入了64位时代），同时对数据库系统实时响应能力要求日益提高，充分利用内存技术提升数据库性能成为一个热点。

在数据库技术中，目前主要有两种方法来使用大量的内存。一种是在传统的数据库中，增大缓冲池，将一个事务所涉及的数据都放在缓冲池中，组织成相应的[数据结构](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)来进行查询和更新处理，也就是常说的共享内存技术，这种方法优化的主要目标是最小化磁盘访问。另一种就是内存数据库(MMDB:Main Memory Database，也叫主存数据库)技术，就是干脆重新设计一种数据库管理系统，对查询处理、并发控制与恢复的[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)和数据结构进行重新设计，以更有效地使用CPU周期和内存，这种技术近乎把整个数据库放进内存中，因而会产生一些根本性的变化。两种技术的区别如下表：



内存数据库系统带来的优越性能不仅仅在于对内存读写比对磁盘读写快上，更重要的是，从根本上抛弃了磁盘数据管理的许多传统方式，基于全部数据都在内存中管理进行了新的体系结构的设计，并且在数据缓存、快速算法、并行操作方面也进行了相应的改进，从而使数据处理速度一般比传统数据库的数据处理速度快很多，一般都在10倍以上，理想情况甚至可以达到1000倍。 而使用共享内存技术的实时系统和使用内存数据库相比有很多不足，由于优化的目标仍然集中在最小化磁盘访问上，很难满足完整的数据库管理的要求，设计的非标准化和软件的专用性造成可伸缩性、可用性和系统的效率都非常低，对于快速部署和简化维护都是不利的。

**2.  内存数据库历史和发展**

**（1）雏形期**  
从上个世纪60年代末到80年代初。在这个时期中，出现了主存数据库的雏形。1969年IBM公司研制了世界上最早的数据库管理系统------基于层次模型的数据库管理系统IMS，并作为商品化软件投入市场。在设计IMS时，IBM考虑到基于内存的数据管理方法，相应推出了IMS/VS Fast Path。Fast Path是一个支持内存驻留数据的商业化数据库，但它同时也可以很好地支持磁盘驻留数据。在这个产品中体现了主存数据库的主要设计思想，也就是将需要频繁访问，要求高响应速度的数据直接存放在物理内存中访问和管理。在这个阶段中，包括网状数据库、关系数据库等其他各种数据库技术也都逐渐成型。

**（2）技术理论成熟期**  
1984年，D J DeWitt等人发表了《主存数据库系统的实现技术》一文。第一次提出了Main Memory Database（主存数据库）的概念。预言当时异常昂贵的计算机主存价格一定会下降，用户有可能将大容量的数据库全部保存在主存中，提出了AVL树、哈希算法、主存数据库恢复机制等主存数据库技术的关键理论，为主存数据库的发展指出了明确的方向 。  
1984年，D J DeWitt等人提出使用非易逝内存或预提交和成组提交技术作为主存数据库的提交处理方案，使用指针实现主存数据库的存取访问。  
1985年，IBM推出了IBM 370上运行的OBE主存数据库   
1986年，RB Hagman提出了使用检查点技术实现主存数据库的恢复机制。威斯康星大学提出了按区双向锁定模式解决主存数据库中的并发控制问题。并设计出MM-DBMS主存数据库。贝尔实验室推出了DALI主存数据库模型。   
1987年，ACM SIGMOD会议中提出了以堆文件（HEAP FILE）作为主存数据库的数据存储结构。Southern Methodist大学设计出MARS主存数据库模型。  
1988年普林斯顿大学设计出TPK主存数据库。   
1990年普林斯顿大学又设计出System M主存数据库。

**（3）产品发展期和市场成长期**  
随着互联网的发展，越来越多的网络应用系统需要能够支持大用户量并发访问、高响应速度的的数据库系统，主存数据库市场成熟   
半导体技术快速发展，半导体内存大规模生产，动态随机存取存储器（DRAM）的容量越来越大，而价格越来越低，这无疑为计算机内存的不断扩大提供了硬件基础，使得主存数据库的技术可行性逐步成熟   
1994年美国OSE公司推出了第一个商业化的，开始实际应用的主存数据库产品Polyhedra   
1998年德国SoftwareAG推出了Tamino Database。   
1999年日本UBIT会社开发出XDB主存数据库产品。韩国Altibase推出Altibase   
2000年奥地利的QuiLogic公司推出了SQL-IMDB   
2001年美国McObject推出eXtremeDB。加拿大Empress公司推出EmpressDB

二、几种主存技术应用的比较  
**第一代：**用户定制的主存数据库。通过应用程序来管理内存和数据；不支持SQL语句, 不提供本地存储, 没有数据库恢复技术；性能好但很难维护和在别的应用中不能使用；应用在实时领域比如工厂自动化生产。  
**第二代：**简单功能的内存数据库。能够快速处理简单的查询；支持部分的 SQL语句和简单的恢复技术；主要目的是能够快速处理大量事务；针对简单事务处理领域,尤其是交换机, 移动通信等。  
**第三代：**通用的主存数据库。针对传统的商业关系型数据库领域，能够提供更高的性能、通用性以及稳定性；提供不同的接口来处理复杂的SQL语句和满足不同的应用领域；可以应用在计费、电子商务、在线安全领域，几乎包括磁盘数据库的所有应用领域。

三、目前几种常见的通用内存数据库  
**eXtremeDB：**eXtremeDB实时数据库是McObject公司的一款特别为实时与[嵌入式](http://lib.csdn.net/base/embeddeddevelopment" \o "嵌入式开发知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)系统数据管理而设计的数据库，只有50K到130K的开销，速度达到微秒级。eXtremeDB完全驻留在主内存中，不使用文件系统（包括内存盘）。eXtremeDB采用了新的磁盘融合技术，将内存拓展到磁盘，将磁盘当做虚拟内存来用，实时性能保持微秒级的同时，数据管理量在32BIT下能达到20G。

**[Oracle](http://lib.csdn.net/base/oracle" \o "Oracle知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank) TimesTen：**Oracle TimesTen是Oracle从TimesTen公司收购的一个内存优化的关系数据库，它为应用程序提供了实时企业和行业（例如电信、资本市场和国防）所需的即时响应性和非常高的吞吐量。Oracle TimesTen可作为高速缓存或嵌入式数据库被部署在应用程序层中，它利用标准的 SQL 接口对完全位于物理内存中的数据存储区进行操作。

**SolidDB：**Solid Information Technology 成立于 1992 年，全球总部位于加州Cupertino，  
Solid数据管理平台将基于内存和磁盘的全事务处理数据库引擎、载体级高可用性及强大的数据复制功能紧密地融为一体。ALTIBASE公司从1999年就一直致力于内存数据库软件和其应用的开发，提供高性能和高可用性的软件解决方案。特别适合通信、网上银行、证券交易、实时应用和嵌入式系统领域。目前占据80%以上内存数据库市场，可以说是当今数据库软件技术的领导者。目前Altibase在国内成功案例也比较多，尤其是在电信行业，已经得到了广泛认可.

**四、 常用内存数据库**

**1 SQLite-一个小型的C程序库，实现了独立的，可嵌入的，零配置的SQL数据库引擎**

**（1）SQLite特性**

事务操作是原子，一致，孤立，并且持久的(ACID)，即使在系统崩溃和电源故障之后。

零配置——不需要安装和管理。

实现了绝大多数SQL92标准。

整个数据库存储在一个单一的文件中。

数据库文件可以在不同字节序的机器之间自由地共享。

支持最大可达2T的数据库。 (241 字节)

字符串和BLOB类型的大小最大可达 2G 字节(231字节)。

小的代码： 完整配置的少于250KB，忽略一些可选特性的少于150KB。

在大多数常见操作上比流行的客户/服务器数据库引擎[更快](http://sqlite.org/speed.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)。

简单，易于使用的[API](http://sqlite.org/capi3.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)。

内建[TCL绑定](http://sqlite.org/tclsqlite.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)。 [另外提供](http://www.sqlite.org/cvstrac/wiki?p=SqliteWrappers" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)可用于许多其他语言的绑定。

具有良好注释的源代码，95%经过测试。

独立：没有外部依赖。

源代码位于[公共域](http://sqlite.org/copyright.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)。 可用于任何用途。

SQLite发行版包含一个独立的命令行访问程序([sqlite](http://sqlitecn.org/sqlite.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank))，可用于管理SQLite数据库，并适合作为一个如何使用SQLite库的例子。

**（2）License:**     SQLite使用Public domain授权（注），对于个人使用和商业使用都是免费的。

**（3）技术上的优点和特性**  
◇轻量级

先说它的第一个特色：轻量级。想必SQLite的作者很看重这个特性，连它的Logo都是用的“羽毛”，来显摆它的轻飘飘。SQLite和C/S模式的数据库软件不同，它是进程内的数据库引擎，因此不存在数据库的客户端和服务器。使用SQLite一般只需要带上它的一个动态库，就可以享受它的全部功能。而且那个动态库的尺寸也挺小，以版本3.6.11为例，Windows下487KB、[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)下347KB。

◇ 绿色软件

SQLite的另外一个特点是绿色：它的核心引擎本身不依赖第三方的软件，使用它也不需要“安装”。所以在部署的时候能够省去不少麻烦。  
◇单一文件

所谓的“单一文件”，就是数据库中所有的信息（比如表、视图、触发器、等）都包含在一个文件内。这个文件可以copy到其它目录或其它机器上，也照用不误。

**（4）技术上的缺点和不足**  
◇并发访问的锁机制  
SQLite在并发（包括多进程和多线程）读写方面的性能一直不太理想。数据库可能会被写操作独占，从而导致其它读写操作阻塞或出错。  
SQL标准支持不全  
在它的[官方网站](http://www.sqlite.org/omitted.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)上，具体列举了不支持哪些SQL92标准。我个人感觉比较不爽的是不支持外键约束。  
◇网络文件系统（以下简称NFS）  
有时候需要访问其它机器上的SQLite数据库文件，就会把数据库文件放置到网络共享目录上。这时候你就要小心了。当SQLite文件放置于NFS时，在并发读写的情况下可能会出问题（比如数据损坏）。原因据说是由于某些NFS的文件锁实现上有Bug。  
  
**（5）编程语言接口**  
SQLite支持很多种语言的编程接口。这对于我这种喜欢混用多种编程语言的人来说，是很爽的。下面我大概介绍一下。  
◇C/C++  
由于SQLite本身是C写的，它[自带的API](http://www.sqlite.org/cintro.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)也是C接口的。所以C/C++用起来最直接了。假如你不喜欢面向过程的C API风格，可以另外找个C++的包装库。想重新发明轮子的同学，也可以自己包装一个。  
◇[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)  
如果要用Java访问SQLite，可以通过SQLite的JDBC驱动，或者通过专门的SQLite包装库。我个人建议走JDBC方式，万一将来要换数据库，代码就不用大改。  
◇[Python](http://lib.csdn.net/base/python" \o "Python知识库" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)  
[pysqlite](http://www.pysqlite.org/" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)是Python操作SQLite的首选。从Python 2.5开始，它已经被整合到Python的标准库中。看来Python社区还是蛮喜欢SQLite嘛。  
◇.Net  
对于喜欢.Net的同学，可以通过SQLite的[ADO.NET](http://sqlite.phxsoftware.com/" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)驱动来访问。  
◇Ruby  
Ruby可以通过[SQLite-Ruby](http://rubyforge.org/projects/sqlite-ruby/" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)操作SQLite数据库，不过我没用过。  
◇Perl  
在CPAN上有[DBD::SQLite](http://search.cpan.org/search?module=DBD::SQLite" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)，不过我也没用过。  
  
**（6）一些非技术的参考因素**

需要根据“[如何选择开源项目](http://program-think.blogspot.com/2009/02/how-to-choose-opensource-project.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)”里面提到的几个参考因素，再评估一下。  
◇授权协议（License）  
SQLite使用的是[Public Domain](http://en.wikipedia.org/wiki/Public_domain" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)协议，这是最爽一种，可以放心大胆地用。  
◇用户的普及程度  
最近这几年，使用SQLite的人越来越多。包括一些大公司也开始把它整合到产品中（比如Google的Gears、Apple的Safari、Adobe的AIR）。  
◇开发的活跃程度  
如果到SQLite的[Change Log](http://www.sqlite.org/changes.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)上大致了解一下，可以看出最近5年基本上每1-2个月都会有更新。说明开发的活跃度还是非常高的。

**2 SQLite不同于其他大部分的SQL数据库引擎,因为它的首要设计目标就是简单化:**

易于管理

易于使用

易于嵌入其他大型程序

易于维护和配置

许多人喜欢SQLite因为它的小巧和快速. 但是这些特性只是它的部分优点, 使用者还会发现SQLite是非常稳定的. 出色的稳定性源于它的简单, 越简单就越不容易出错. 除了上述的简单、小巧和稳定性外, 最重要的在于SQLite力争做到简单化.

简单化在一个数据库引擎中可以说是一个优点, 但也可能是个缺点, 主要决定于你想要做什么. 为了达到简单化, SQLite省略了一些人们认为比较有用的特性, 例如高并发性、 严格的存取控制、丰富的内置功能、 存储过程、复杂的SQL语言特性、 XML以及Java的扩展, 超大的万亿级别的数据测量等等. 如果你需要使用上述的这些特性并且不介意它们的复杂性, 那么SQLite也许就不适合你了. SQLite没有打算作为一个企业级的数据库引擎, 也并不打算和Oracle或者PostgreSQL竞争.

仅凭经验来说SQLite适用于以下场合: 当你更看中简单的管理、使用和维护数据库, 而不是那些企业级数据库提供的不计其数的复杂功能的时候，使用SQLite是一个比较明智的选择. 事实也证明, 人们在许多情况下已经清楚的认识到简单就是最好的选择.

**3 SQLite最佳试用场合**

**（1）网站**

作为数据库引擎SQLite适用于中小规模流量的网站(也就是说, 99.9%的网站). SQLite可以处理多少网站流量在于网站的数据库有多大的压力. 通常来说, 如果一个网站的点击率少于100000次/天的话, SQLite是可以正常运行的. 100000次/天是一个保守的估计, 不是一个准确的上限. 事实证明, 即使是10倍的上述流量的情况下SQLite依然可以正常运行.

**（2）嵌入式设备和应用软件**

因为SQLite数据库几乎不需要管理, 因此对于那些无人值守运行或无人工技术支持的设备或服务, SQLite是一个很好的选择. SQLite能很好的适用于手机, PDA, 机顶盒, 以及其他仪器. 作为一个嵌入式数据库它也能够很好的应用于客户端程序.

**（3）应用程序文件格式**

SQLite作为桌面应用程序的本地磁盘文件格式取得了巨大成功.例如金融分析工具、CAD 包、档案管理程序等等. 一般的数据库打开操作需要调用sqlite3\_open()函数，并且标记一个显式本地事务的起始点（BEGIN TRANSACTION）来保证以独占的方式得到文件的内容. 文件保存将执行一个提交（COMMIT）同时标记另一个显式本地事务起始点. 这种事务处理的作用就是保证对于应用程序数据文件的更新是原子的、持久的、独立的和一致的.数据库里可以加入一些临时的触发器,用来把所有的改变记录在一张临时的取消/重做日志表中. 当用户按下取消/重做按钮的时候这些改变将可以被回滚. 应用这项技术实现一个无限级的取消/重做功能只需要编写很少的代码.

**（4）替代某些特别的文件格式**

许多程序使用fopen(), fread(), 或 fwrite()函数创建和管理一些自定义的文件用来保存数据. 使用SQLite替代这些自定义的文件格式将是一种很好的选择.

**（5）内部的或临时的数据库**

对于那些有大量的数据需要用不同的方式筛选分类的程序, 相对于编写同样功能的代码, 如果你把数据读入一个内存中的SQLite数据库, 然后使用连接查询和ORDER BY子句按一定的顺序和排列提取需要的数据, 通常会更简单和快速. 按照上述的方法使用内嵌的SQLite数据库将会使程序更富有灵活性, 因为添加新的列或索引不用重写任何查询语句.

**（6）命令行数据集分析工具**

有经验的SQL用户可以使用SQLit命令行程序去分析各种混杂的数据集. 原是数据可以从CSV（逗号分隔值文件）文件中导入, 然后被切分产生无数的综合数据报告. 可能得用法包括网站日志分析, 运动统计分析, 编辑规划标准, 分析试验结果.当然你也可以用企业级的客户端/服务器数据库来做同样的事情. 在这种情况下使用SQLite的好处是: SQLite的部署更为简单并且结果数据库是一个单独的文件, 你可以把它存储在软盘或者优盘或者直接通过email发给同事.

**4 哪些场合适合使用其他的关系型数据库管理系统（RDBMS）**

**（1）客户端/服务器程序**

如果你有许多的客户端程序要通过网络访问一个共享的数据库, 你应当考虑用一个客户端/服务器数据库来替代SQLite. SQLite可以通过网络文件系统工作, 但是因为和大多数网络文件系统都存在延时, 因此执行效率不会很高. 此外大多数网络文件系统在实现文件逻辑锁的方面都存在着bug(包括Unix 和windows). 如果文件锁没有正常的工作, 就可能出现在同一时间两个或更多的客户端程序更改同一个数据库的同一部分, 从而导致数据库出错. 因为这些问题是文件系统执行的时候本质上存在的bug, 因此SQLite没有办法避免它们.

好的经验告诉我们, 应该避免在许多计算机需要通过一个网络文件系统同时访问同一个数据库的情况下使用SQLite.

**（2） 高流量网站**

SQLite通常情况下用作一个网站的后台数据库可以很好的工作. 但是如果你的网站的访问量大到你开始考虑采取分布式的数据库部署, 那么你应当毫不犹豫的考虑用一个企业级的客户端/服务器数据库来替代SQLite.

**（3）超大的数据集**

当你在SQLite中开始一个事务处理的时候(事务处理会在任何写操作发生之前产生, 而不是必须要显示的调用BEGIN...COMMIT), 数据库引擎将不得不分配一小块脏页(文件缓冲页面)来帮助它自己管理回滚操作. 每1MB的数据库文件SQLite需要256字节. 对于小型的数据库这些空间不算什么, 但是当数据库增长到数十亿字节的时候, 缓冲页面的尺寸就会相当的大了. 如果你需要存储或修改几十GB的数据, 你应该考虑用其他的数据库引擎.

**（4）高并发访问**

SQLite对于整个数据库文件进行读取/写入锁定. 这意味着如果任何进程读取了数据库中的某一部分, 其他所有进程都不能再对该数据库的任何部分进行写入操作. 同样的, 如果任何一个进程在对数据库进行写入操作, 其他所有进程都不能再读取该数据库的任何部分. 对于大多数情况这不算是什么问题. 在这些情况下每个程序使用数据库的时间都很短暂, 并且不会独占, 这样锁定至多会存在十几毫秒. 但是如果有些程序需要高并发, 那么这些程序就需要寻找其他的解决方案了.

|  |  |  |  |
| --- | --- | --- | --- |
| **方面** | **具体要求** | **必要条件** | **详细描述** |
| License | 是否收费 |  | 免费使用 |
| 是否开源 |  | 开源 |
| 是否有技术支持 |  | 主要是社区支持，如果需要专业支持需要购买 |
| 商业目的的分发版本是否仍要收费 | 是 | 免费 |
| 其他 |  |  |
| 性能 | 数据容量支持100000条以上记录 | 是 | 支持 |
| 并发查询处理能力 |  | SQLite在并发（包括多进程和多线程）读写方面的性能一直不太理想。数据库可能会被写操作独占，从而导致其它读写操作阻塞或出错。 |
| 查询速度 | 是 |  |
| 修改速度 | 是 |  |
| 平台支持 | 32/64位 |  | 全部支持 |
| Linux/window/UNIX/mobile |  | 支持Linux/Mac OS/Windows |
| 运行方式支持 | 支持嵌入式 |  | 支持 |
| 支持独立运行 |  | 不支持 |
| 连接方式支持 | 支持ODBC |  | 默认不支持，必须通过第三方的ODBC驱动 |
| 支持JDBC |  | 默认不支持，必须通过第三方的JDBC驱动 |
| 支持内存访问 |  | 通过c接口（专用API） |
| 支持网络访问 |  | 不支持 |
| SQL支持 | 支持SQL | 是 | 支持 |
| 支持Index，Trigger，  Constrains，Views |  | 支持，有资料说其不支持外键约束。 |
| 管理界面 | 支持管理界面 |  | 支持CLI |
| 管理界面友好程度 |  | 较差 |

**5 Altibase**

Altibase™内存数据库管理系统（DBMS），内存数据管理系统的最新技术，是一个在事务优先的环境中提供高性能和高可用性的软件解决方案。Altibase提供极限性能、容错能力和事务管理的方便性，特别是在通信、网上银行、证券交易、实时应用和嵌入式系统领域。Altibase能够最大限度的发挥数据库服务系统的潜力，使用Altibase能大大增强您公司的数据服务器的处理能力。

Altibase™内存DBMS为需要容错服务的系统提供实时数据库复制的功能。采用Altibase数据库复制的系统可以实现高性能、高可用性、数据库一致性、负载平衡和系统可伸缩性。如果您希望您的业务能够实现最大的成功，请在您的事务优先的系统中使用我们的Altibase数据库复制解决方案。

资料比较少，且需要商业License，没有详细去研究

**6 Oracle 内存数据库系列 Berkeley DB 和 TimesTen**

Oracle是最重要的商业数据库产品提供商，它也有内存数据库的产品系列：主要就是Oracle Berkeley DB 和 Times Ten.前者是只支持嵌入式内存数据，后者是独立的内存优化数据库。

（1） Oracle Berkeley DB

 Oracle Berkeley DB是Oracle 收购了开源数据库厂商后推出的产品，其前身是Berkeley DB。它有开源版本，但且对于开源软件免费。商业版本是要付费。

Oracle Berkeley DB 系列的可嵌入开源数据库为开发人员提供了无需管理的快速、可靠的本地持久性。Oracle Berkeley DB 系列通常部署为“前沿”数据库，为不需要 SQL 的应用程序用例提供很高的性能、可靠性、可伸缩性以及可用性。

Oracle Berkeley DB 产品系列

—     [Berkeley DB](http://www.oracle.com/technology/global/cn/products/berkeley-db/db/index.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank) — 事务处理式存储引擎，用于基本键/值数据结构中的非类型化数据 — 新增！版本 4.7 现已推出

—     针对 Java 环境优化的纯 Java 版 Berkeley DB — 新增！版本 3.3

—     [Berkeley DB XML](http://www.oracle.com/technology/global/cn/products/berkeley-db/xml/index.html" \t "https://blog.csdn.net/dh2442897094/article/details/_blank) — 原生 XML 数据库，可基于 XQuery 访问容器中存储的文档，并根据其内容进行索引 — 新增！版本 2.4 现已推出

**7 Oracle TimesTen**

Oracle 内存数据库 TimesTen 是一个针对内存进行了优化的关系数据库，它为应用程序提供了当今实时企业和行业（如电信、资本市场和国防）所需的即时响应性和非常高的吞吐量。Oracle 内存数据库 TimesTen 作为独立或嵌入式数据库部署在应用层中，利用标准的 SQL 接口对完全位于物理内存中的数据库进行操作。它也可以用作 Oracle 数据库的内存中数据库缓存，以改进用户应用程序的响应时间和吞吐量。

**8 eXtremeDB**

（1）eXtremeDB内存式实时数据库是为实时系统及嵌入式系统而特别设计的数据库。与同类产品不同，eXtremeDB不是通过 对企业数据库面向实时嵌入式应用进行剪裁而来；而是总结了30年来McObject公司在编译器、实时编程、数据管理、内核级驱 动软件等领域的经验，面向实时嵌入式应用从头开发的最新实时数据管理技术。

（2）eXtremeDB满足了您对实时数据库的一切期待：高级数据定义语言、并行访问、基于交易及灵活的索引… …等等。不仅如此，出乎您的意外，eXtremeDB在紧凑的引擎中还提供诸如事件触发、目标历史等等功能。

（3）eXtremeDB嵌入式数据库满足更多的实时开发的要求。

·                     最快的内存数据库。

·                     [极小尺寸和极小的内存消耗](http://www.mcobject.com/tiny_footprint_database" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)

·                     [多种索引支持](http://www.mcobject.com/database_index" \t "https://blog.csdn.net/dh2442897094/article/details/_blank)

·                     高可用性-组合选项

·                     [非常灵活的数据存储：](http://www.mcobject.com/hybrid_database.shtml" \t "https://blog.csdn.net/dh2442897094/article/details/_blank) 内存式，磁盘式或混合式

·                     [多种应用接口：](http://www.mcobject.com/database_api" \t "https://blog.csdn.net/dh2442897094/article/details/_blank) 两种 SQL， 两种更快的原始接口

·                     几乎牢不可破 -

又一个商业内存数据库产品，这个特点是实时数据库，号称最快。

**9 H2 Database**

（1）h2是Thomas Mueller提供的一个开源的、纯java实现的关系数据库，官方网站：

（2）它的主要特性是：

非常速的数据库引擎

开源、免费数据库

支持 JDBC和ODBC API，支持SQL

支持嵌入式，服务器和集群模式。支持内存数据库。

提供基于浏览器的管理控制台

整个应用本身只有1MB左右。

其他特性还包括

基于磁盘或内存的数据库、表，支持只读数据库、临时表。

两段式事务支持

支持多个连接。表级别的锁。

基于成本的优化，为复杂查询使用遗传算法，零管理。

滚动的、可修改的result set支持。支持大结果集、外部结果排序。

加密数据库（AES或XTEA）,SHA-256密码加密。

性能比较（摘自h2database网站）

（3）嵌入模式下H2的性能比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Unit** | **H2** | **HSQLDB** | **Derby** |
| Simple: Init | ms | 610 | 657 | 3187 |
| Simple: Query (random) | ms | 297 | 312 | 1828 |
| Simple: Query (sequential) | ms | 203 | 266 | 1766 |
| Simple: Update (random) | ms | 1078 | 1484 | 22031 |
| Simple: Delete (sequential) | ms | 234 | 281 | 7407 |
| Simple: Memory Usage | MB | 6 | 7 | 11 |
| BenchA: Init | ms | 859 | 438 | 4047 |
| BenchA: Transactions | ms | 5266 | 2875 | 17500 |
| BenchA: Memory Usage | MB | 9 | 14 | 10 |
| BenchB: Init | ms | 4016 | 2687 | 16875 |
| BenchB: Transactions | ms | 2609 | 3282 | 4250 |
| BenchB: Memory Usage | MB | 9 | 10 | 8 |
| BenchC: Init | ms | 891 | 594 | 5766 |
| BenchC: Transactions | ms | 4359 | 75438 | 11718 |
| BenchC: Memory Usage | MB | 9 | 18 | 9 |
| Executed statements | # | 594255 | 594255 | 594255 |
| Total time | ms | 20422 | 88314 | 96375 |
| Statements per second | # | 29098 | 6728 | 6166 |

.Net使用H2嵌入式应用。有一个项目在为.Net使用H2，使用CLI重新编译H2。还没有深入关注。

ODBC。但性能一般。

**10 其他内存数据库**

包括Derby, HSQLDB等.