《Architecture of a Database System》

（中文版）

Joseph M. Hellerstein, Michael Stonebraker and James Hamilton

翻译：林子雨

厦门大学数据库实验室

http://dblab.xmu.edu.cn

中文版网址：http://dblab.xmu.edu.cn/node/459

厦门大学计算机科学系教师 林子雨 翻译作品

http://www.cs.xmu.edu.cn/linziyu

2013 年 9 月

前言

本文翻译自经典英文论文《Architecture of a Database System》，原文作者是 Joseph M. Hellerstein, Michael Stonebraker 和 James Hamilton。该论文可以作为中国各大高校数据库实验室研究生的入门读物，帮助学生快速了解数据库的内部运行机制。

本文一共包括 6 章，分别是：第 1 章概述，第 2 章进程模型，第 3 章并行体系结构：进程和内存协调，第 4 章关系查询处理器，第 5 章存储管理，第 6 章事务：并发控制和恢复，第 7 章共享组件，第 8 章结束语。

本文翻译由厦门大学数据库实验室林子雨老师团队合力完成，其中，林子雨老师负责统稿校对，刘颖杰同学负责翻译第 1 章和第 2 章，罗道文同学负责翻译第 3 章和第 4 章，谢荣东同学负责翻译第 5 章、第 6 章、第 7 章和第 8 章。

如果对本文翻译内容有任何疑问，欢迎联系林子雨老师。

林子雨的E-mail是：ziyulin@xmu.edu.cn。

林子雨的个人主页是：http://www.cs.xmu.edu.cn/linziyu。

厦门大学数据库实验室网站是：http://dblab.xmu.edu.cn。

本文中文版的网址是：http://dblab.xmu.edu.cn/node/459。

林子雨于厦门大学海韵园

2013 年 9 月

摘 要

数据库管理系统（DBMS）广泛存在于现代计算机系统中，并且是其重要的组成部分。它是学术界以及工业界数十年研究和发展的成果。在计算机发展史上，数据库属于最早开发的多用户服务系统之一，因此，它的研究也催生了许多为保证系统可拓展性以及稳定性的系统开发技术，这些技术如今被应用于许多其他的领域。虽然许多数据库的相关算法和概念广泛见于教科书中，但关于如何让一个数据库工作的系统设计问题却鲜有资料介绍。本文从体系架构角度探讨数据库设计的一些准则，包括处理模型、并行架构、存储系统设计、事务处理系统、查询处理及优化结构以及具有代表性的共享组件和应用。当业界有多种设计方式可供选择时，我们以当前成功的商业开源软件作为参考标准。

第 3 章 并行架构：进程和内存协调

厦门大学计算机科学系教师 林子雨 编著

个人主页：http://www.cs.xmu.edu.cn/linziyu

中文版网址：http://dblab.xmu.edu.cn/node/459

2013 年 9 月

第 3 章 并行架构：进程和内存协调

并行硬件在现代的服务器中已经是一个不争的事实，它们存在于各种各样的配置中。在这一章，我们将总结标准的数据库管理系统术语，接着分别讨论进程模型和内存协调（memory coordination）问题。

3.1 共享内存

在一个共享内存的并行系统中（如图 3-1 所示），所有的处理机可以使用相同的内存和硬盘，并且拥有大致相同的性能。这个架构现在已经是一个标准。大多数服务器硬件都包含 2 个到 8 个的处理器。高端的机器可以包含数十个处理器，可以提供更高的处理器资源，但是，也往往价格更加昂贵。高度并行共享内存的机器是硬件行业最后剩下的摇钱树之一，并大量使用在高端在线事务处理应用程序中。服务器硬件的代价与管理系统的代价相比，不免就相形见绌了，所以，购买少量大型的、昂贵的系统有时候被看作是可以接受的折中方案。

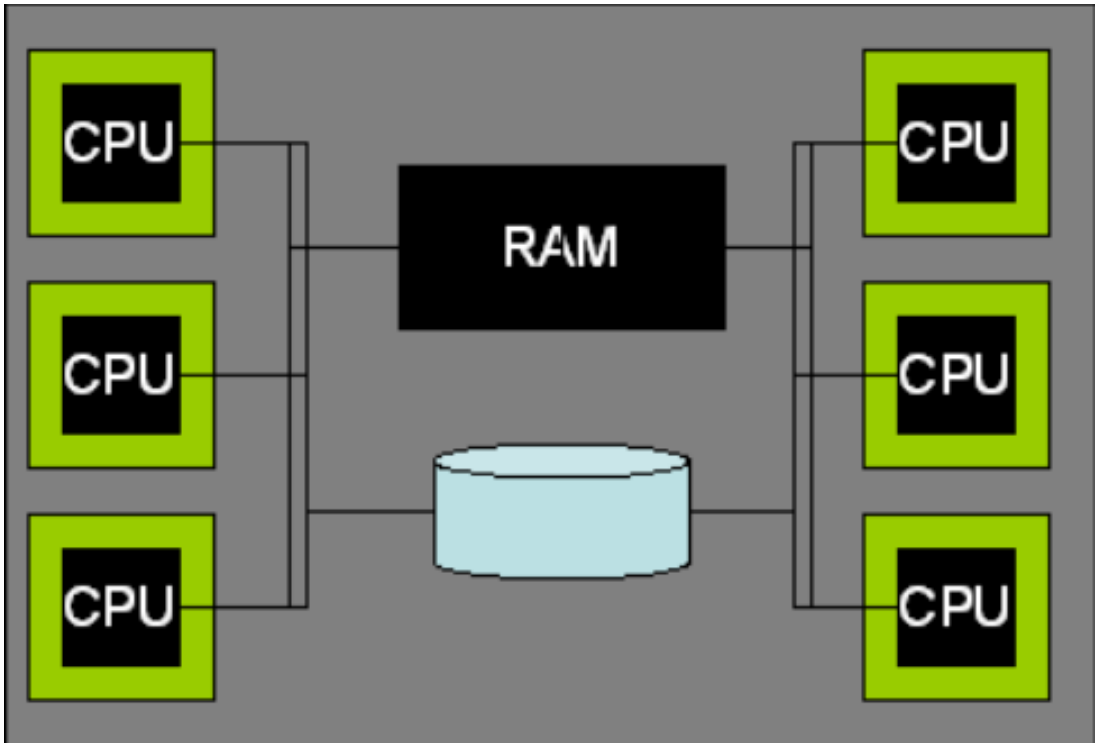

图 3-1 共享内存体系架构

多核处理器在单一芯片上支持多个处理内核和共享一些基础结构，如高速缓存（cache）和内存总线。这使得它们在编程方面非常类似于共享内存的架构。如今，几乎所有的数据库部署都涉及到多个处理器，并且每个处理器都不只一个 CPU。DBMS 架构需要充分利用这种潜在的并行。幸运的是，在第二章描述的所有三个 DBMS 的架构，可以在现代共享内存硬件架构上运行得很好。

共享内存机器的进程模型很自然地遵循单处理机的方式。事实上，大多数数据库系统都是从他们最原始的单处理机器实现，然后演化为共享内存实现。在共享内存的机器中，操作系统通常支持作业（进程或线程）被透明地分配到每个处理器上，并且共享的数据结构继续可以被所有的作业所访问。所有三模型可以在这些系统上运行良好，而且支持多个独立的 SQL 并行请求的执行。面临的主要挑战是修改查询执行层，从而利用将一条单一的查询语句并行到多个处理器上能力。我们将推迟到第 5 章介绍。

3.2 无共享

一个无共享的并行系统是由多个独立计算机的集群组成的，这些计算机可以高速地通过网络进行互连通信，或者逐渐频繁地在商业网络组件上通信。对于给定的一个系统，无法直接访问另一个系统的内存和硬盘。

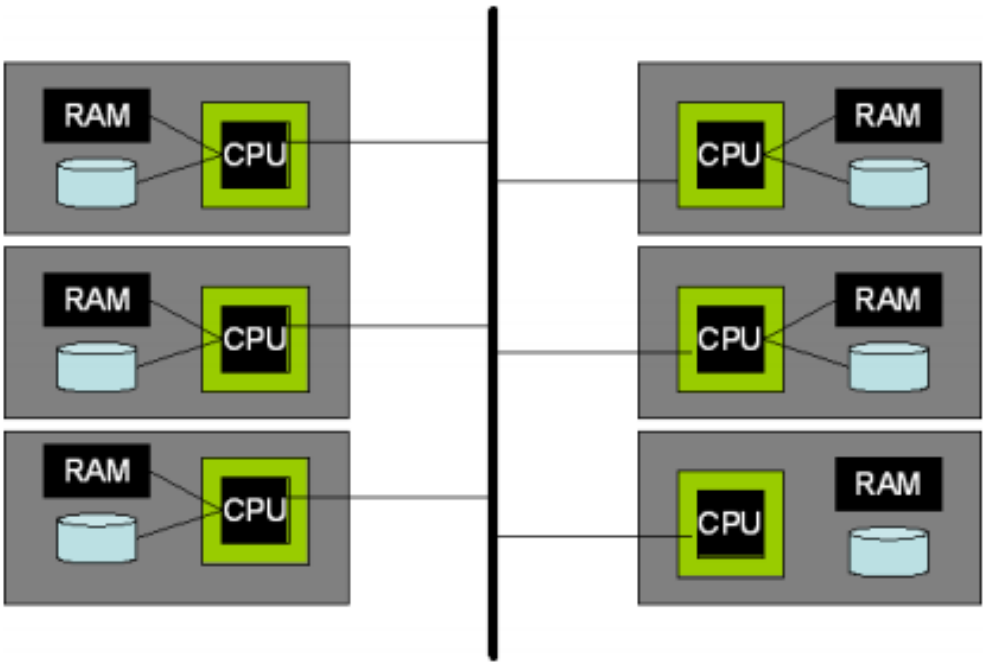

图 3-2 无共享体系架构

无共享系统并没有提供抽象的硬件共享，协调不同机器的任务完全留给了 DBMS。DBMS 支持这些集群最常用的技术就是在集群中的每台机器或每个节点上运行它们标准的进程模型。每一个节点都能够接受客户端 SQL 请求、访问需要的元数据、编译 SQL 请求、进行数据访问，正如上述描述的单一共享内存一样。主要的区别就是，集群中的每个系统仅仅保存一部分的数据。对于每一个 SQL 请求，无共享系统中的每一个节点不只是单独执行查询本地的数据，这个请求会被发送到集群中的其他成员，然后所有的计算机并行地执行查询本地所保存的数据。每个表格会通过*水平数据分区*传播到集群的多个系统中，因此，每个处理器可以独立于其他处理器执行。

数据库中的每一个元组被分配到到单独的机器，因此，每一张表被水平地切分，然后分布到整个系统。典型的数据分区方案包括：元组属性基于哈希的分区，元组属性基于范围的分区，round-robin，以及混合型分区方案（前两种方案的混合）。每一个单独的计算机都负责访问、锁定、记录本地磁盘上的数据。在查询执行期间，查询优化器会选择如何对表进行重新水平分区，然后把中间结果分布到各个计算机上以满足查询的需要，并且给每一台机器分配一个作业的逻辑分区。不同的计算机上的查询执行部件，将数据请求和元组传输到其他计算机中，但是，没必要传输任何线程状态和其他低级的信息。数据库元组采用基于值的分区导致的一个很自然的结果就是，在这些系统中，只需要最少的协调工作。然而，为了得到很好的性能，就需要很好地数据分区。这就给数据库管理员一个重大的负担——合理明智地

确定表的分布。同时给查询优化器一个重大的负担——需要在负载分区方面做得很好。

这简单分区方案并不能处理 DBMS 上所有的问题。例如，必须采取明确的跨处理器协调来处理事务完成，提供负载平衡以及支持某种维护任务。例如，对于一些像分布式死锁检测和两阶段提交[30]等问题，处理器之间必须交换明确的控制信息。这就需要额外的逻辑，而且如果做得不好，就可能是个性能瓶颈。

同时，在无共享系统中，局部故障是必须被妥善管理的。在一个无共享系统中，一个处理器发生故障通常会导致整个系统的停止运行，因此，整个 DBMS 也会停止运行。在一个无共享系统中，集群中一个单一的节点发生故障，并不一定会影响到其他的节点。但是，这肯定会影响 DBMS 的整体运行，因为，失败的节点里寄存着数据库中部分的数据。在这种情形下，至少有三种可行的方法。第一种办法就是，如果有任何一个节点发生故障，就停止运行所有的节点；这本质上是模拟在一个共享内存系统中将会发生的事情。第二种方法，在 Informix 上称为“数据跳跃”，允许在正常的节点上继续执行查询，而跳过故障节点的数据。这在数据的可用性比结果的完整性更重要的情况下是很有用的。但是，竭尽全力的结果，不具备良好定义的语义，对于许多负载而言，这不是一个有用的选择——尤其是因为在多层系统中 DBMS 通常被用作“记录存储库”，需要在更高的层面（通常是一个应用服务器）来权衡可靠性和一致性。第三种方法是采用冗余方案，范围从完整的数据库失败恢复（需要两倍的计算基数量和软件数量）到类似于链式分簇的细粒度冗余[43]。在这后面的技术中，元组副本会分布在集群中的多个节点。链式分簇比其他更简单机制的优势是：(a)需要部署更少的机器，就可以保证比其他简单机制获得更好的可用性；(b)当一个节点发生故障时，系统负载将会相当均匀地分配到剩下的节点：剩下的 n-1 个节点，每个完成原来工作的 n/(n-1)。这种形式的线性性能下降，会随着节点的失败而持续。实际上，许多通用的商业化系统是介于中间的，既不是粗粒度的全数据库冗余，也不是细粒度的链式分簇。

无共享架构如今已是相当普遍，并且拥有无与伦比的可拓展性与成本特性。它主要用在极高端应用中，通常是决策支持应用和数据仓库。在一个硬件架构的有趣组合中，一个无共享的集群通常是由许多节点构成的，而每个节点都是共享内存的多处理器。

3.3 共享磁盘

如图 3-3 所示，在一个共享磁盘的并行系统中，所有的处理器可以访问具备大致相同性能的磁盘，但是，不能访问彼此的 RAM。这种架构是相当普遍的，两个突出的例子是 Oracle

RAC 和 DB2 for zSeries SYSPLEX。随着存储区域网络（SAN：Storage Area Networks）的普及，共享磁盘在最近几年已经变得越来越普遍。一个 SAN 允许一个或多个逻辑磁盘被安装到一个或多个宿主系统上，这样可以使得创建共享磁盘配置变得较为容易。

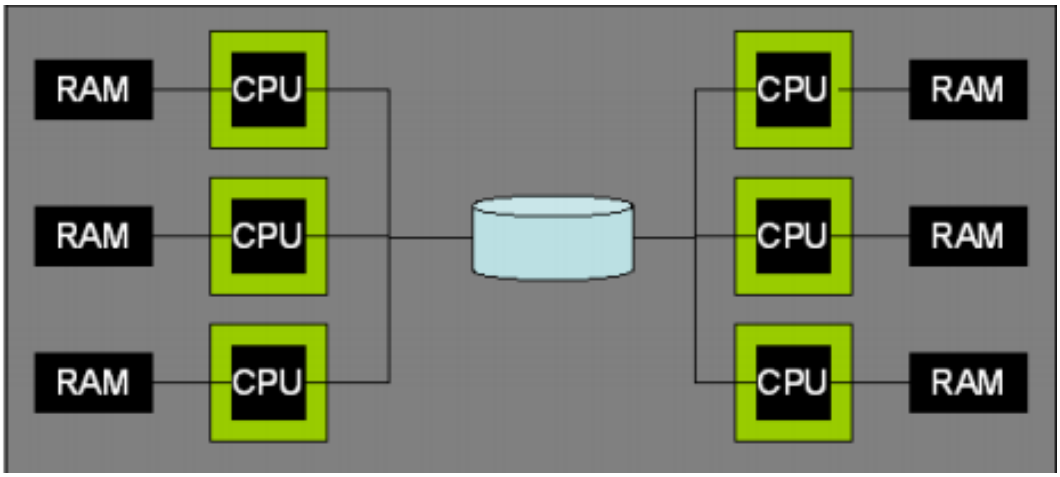

图 3-3 共享磁盘体系架构

共享磁盘系统相对于无共享系统而言的一个潜在的优势就是，它们管理成本较低。共享磁盘系统的 DBAs 没必要为了实现并行而考虑对表进行分区并分布到不同计算机上。但是，大型的数据库仍然需要分区，所以，对于大规模数据库，二者的区别不是很显著。共享磁盘架构的另一个引人注目的特征是，单个 DBMS 处理节点发生故障不会影响其他节点访问整个数据库。这一点既不同于共享内存系统（作为一个整体发生故障），也不同于无共享系统（由于某个节点发生故障会导致至少无法访问一些数据，除非使用一些其他数据冗余机制）。然而，即使有这些优势，共享磁盘系统仍然容易受到单点故障。当数据未到达存储子系统时，如果数据由于硬件或软件故障而损坏或发生其他方式的毁坏，则系统所有的节点只能访问这些损坏的页面。如果存储子系统使用 RAID 或者其他数据冗余技术，则损坏的页面将会被冗余储存，但所有的副本仍然是损坏的。

因为在一个共享磁盘系统中不需要对数据进行分区，所以数据可以被复制到 RAM，并在多个计算机上进行修改。与共享内存系统不同的是，共享磁盘系统不存在很自然的内存位置来协调数据的共享，每个计算机都有为锁和缓冲池页面准备的本地内存。因此，需要在多台计算机之间进行显示地数据共享协调。共享磁盘系统依赖于一个分布式锁管理设备和一个管理分布式缓冲池的高速缓存一致性协议。这些都是复杂的软件组件，并且对于具备竞争性的负载而言会成为瓶颈。有些系统是由硬件子系统来实现锁管理的，比如 IBM zSeries SYSPLEX。

3.4 非均衡内存访问

非均衡内存访问（NUMA：Non-Uniform Memory Access）系统，在一个拥有独立内存的集群系统中，提供了共享内存编程模型。集群中的每个系统都能快速访问本地内存，然而高速集群中每个计算机之间的远程访问，会存在一定程度上的互连延迟。这个架构的名字来自于这种内存访问时间的不一致。

NUMA 的硬件架构是处于无共享系统和共享内存系统之间的一个有趣的中间地带，它们比无共享集群更容易编程，同时，比共享内存系统拥有更大规模的处理器，这样就可以避免共享点的争用，例如共享内存系统总线。

NUMA 集群在商业上并没有获得广泛的成功，但是，NUMA 设计概念正在被共享内存多处理器领域所采用。伴随着共享内存多处理器系统已经扩展到大量处理器的规模，它们在内存架构上已经展示了逐渐增长的非均衡性。通常大规模的共享内存多处理器系统的内存被分为多个部分，每个部分与系统中处理器一个小子集相关联。每一个内存和 CPU 相结合的子集通常被称为一个“pod”。每一个处理器访问本地的 pod 内存的速度，要稍快于访问远程 pod 的内存。NUMA 设计模式的使用已经允许共享内存系统扩展到更多数量处理器的规模，从而使得 NUMA 共享内存多处理器如今已经非常普遍了，而 NUMA 集群并没有成功取得任何显著的市场份额。

DBMS 可以在 NUMA 共享内存系统上运行的一种方式是，通过忽略内存访问的非均衡性。如果非均衡性较小的时候，这种方式还勉强可以接受。但是，当近程内存访问时间与远程内存访问时间的比率从 1.5:1 升到 2:1 的范围时，DBMS 需要使用优化，以避免内存访问瓶颈。这些优化可以有不同的形式，但是，所有都要遵循相同的基本方法：（1）当为一个处理器分配内存使用的时候，尽量使用本地的内存（避免使用远程内存）；（2）如果可能的话，保证一个 DBMS 使用者被安排到与之前相同的硬件处理器。这种组合允许 DBMS 负载在更大的规模上运行良好，同时，共享内存系统拥有一些内存访问时间的非均衡性。

尽管 NUMA 集群已经全部消失了，但是，编程模式和优化技术仍然对于当代的 DBMS 系统很重要，因为许多大规模的共享内存系统在内存访问性能上有着显著的非均衡性。

3.5 线程和多处理器

当我们除去第 2.1 节中关于单处理机硬件的最后两个简化假设的时候，使用 DBMS 线

程来实现每个 DBMS 工作者一个线程时，一个潜在的问题就会立即变得非常明显。第 2.2.1 节描述的轻量级 DBMS 线程包的自然实现方式就是，所有线程运行在一个单一的操作系统进程中。不幸的是，一个单一的进程一次只能在一个处理器上执行。因此，在一个多处理器系统中，DBMS 一次只能用一个处理器，而系统中其他的处理器将会被闲置。早期的 Sybase SQL Server 架构就受到这样的限制。在 90 年代，随着共享内存多处理器越来越受欢迎，Sybase 很快就更改了系统架构，开发出了多系统进程的架构。

当在多进程上运行多个 DBMS 线程的时候，就会出现这样的时刻，一个进程拥有大部分的工作，其他进程（处理器也一样）就被闲置了。在这种情形下，为了让这种模式工作得更好，DBMS 必须实现在进程之间迁移线程。

当 DBMS 线程映射到多个操作系统进程的时候，需要作出以下决定：（1）需要使用多少个操作系统进程；（2）怎么将 DBMS 线程分配到操作系统线程；（3）怎么分配到多个操作系统进程。 一个好的经验法则就是每个物理处理器产生一个进程。这样就可以最大化硬件固有的并行度，同时最小化每个进程的内存开销。

3.6 标准的操作规程

对于并行度的支持，趋势就和上节所述类似，即大多数主流的 DBMS 支持多种模式的并行。 由于共享内存系统（SMPs，多核系统和二者的结合）在商业上的普及，所有主要的 DBMS 供应商都支持共享内存的并行。但是，我们开始看到，在多节点的集群并行方面（可以采用共享磁盘或者无共享的设计），不同厂商提供了不同的支持。

- *共享磁盘*：所有主要的商业 DBMS 提供对共享内存并行的支持，包括：IBM DB2、Oracle 和 Microsoft SQL Server；
- *无共享*：这种模型被 IBM DB2、Informix、Tandem 和 NCR Teradata 所支持；Greenplum 提供了一种 PostgreSQL 的定制版本，可以支持无共享并行；
- *共享磁盘：*这种模型被被 Oracle RAC、RDB(是甲骨文从数字设备公司收购而来的)和 IBM DB2 的 zSeries 所支持。

IBM 销售不同的 DBMS 产品，选择在一些产品上实现共享磁盘支持，在其他产品上支持无共享模型。到目前为止，没有一款领先的商业系统在一个单一的代码库中同时支持无共享模式和共享磁盘模式。Microsoft SQL Server 也没有实现。

3.7 讨论与附加材料

上述的设计代表了在不同服务器系统中所使用的一些硬件/软件架构模式。虽然这些设计率先出现在 DBMS 中，但是，在其他数据密集领域，包括像 Map-Reduce（有越来越多的用户使用它来处理各种各样的自定义数据分析任务）这样低级别的可编程的数据处理后端，这些设计理念正在获得越来越多的认可。然而，尽管这些设计理念正在更加广泛地影响计算，数据库系统并行性的设计仍然有新的问题产生。

并行软件架构在未来十年将迎来一个重要挑战，这个挑战来自于处理器供应商开发出新一代“众核”架构的构想。这些设备将会引进一种新的硬件设计理念，即在一个单一的芯片上拥有数十、数百甚至数千的处理单元，通过高速芯片网络互联，但是，仍然保留了许多现有的瓶颈，比如访问芯片外的内存和磁盘。这将会导致在磁盘和处理器的内存路径上产生新的不平衡和瓶颈，这就必定需要重新审视 DBMS 架构，以满足硬件性能潜力。

在面向服务的计算领域，一些相关的架构转向更大的规模已经是可以预见的。这里的核心理念是，由数以万计的电脑组成的大数据中心来为用户安排处理（硬件和软件）。在这样的规模下，只有实现高度自动化，才能负担得起应用程序和服务器管理。没有什么管理任务可以随着服务器的数量一直扩展。而且，集群中通常使用更不可靠的商业服务器，故障是不可避免的，从故障中恢复就需要完全的自动化操作。在这样规模下的服务，将会每天出现磁盘故障，每星期出现服务器故障。在这样的环境中，管理数据库的备份通常被整个数据库的冗余在线副本所取代，这些副本维护在不同磁盘上不同的服务器中。根据数据的价值，冗余副本可能保存在不同的数据中心。仍然可以采用自动离线备份，从而可以从应用程序、管理工作或者用户错误中恢复。然而，从最常见的错误和故障中恢复是将故障快速转移到冗余的在线副本。冗余可以用多种方式实现：(a)在数据存储级的复制（存储区域网络）；(b)在数据库存储引擎级别的复制（将在第 7.4 节讨论）；(c)由查询处理器执行冗余查询（第 6 章）；或者(d)在客户端软件级别自动生成冗余数据库请求（例如，WEB 服务器或应用程序服务器）。

在一个更高的解耦水平，在实际应用中为多个具备 DBMS 功能的服务器进行分层部署是很普遍的，以减少“DBMS 记录”的请求率。这些机制包括各种形式的、针对 SQL 查询的中间层数据库缓存（包括专业的内存数据库，比如 Oracle TimesTen），以及更多被配置成服务于这个目的的传统数据库。在部署层次的更高层面，许多支持编程模型（比如 Enterprise Java Bean）的、面向对象的“应用-服务器”架构，可以配置成为配合 DBMS 工作的、应用对象的事务缓存。然而，这些各式各样的方案的选择、设置和管理，仍然是不标准的，并且

很复杂，普遍接受的优秀模型仍然难以实现。

附录 1:译者介绍

林子雨(1978－),男,博士,厦门大学计算机科学系助理教授,主要研究领域为数据库,数据仓库,数据挖掘.
主讲课程：《大数据技术基础》
办公地点：厦门大学海韵园科研 2 号楼
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu