- □ **存储管理器**(storage manager)子系统在数据库中存储的低层数据与应用程序和向系统提交的查询之间提供接口。
- □ **查询处理器**(query processor)子系统编译和执行 DDL 和 DML 语句。

- **事务管理**(transaction management)负责保证不管是否有故障发生，数据库都要处于一致的(正确的)状态。事务管理器还保证并发事务的执行互不冲突。
- 数据库系统的体系结构受支持其运行的计算机系统的影响很大。数据库系统可以是集中式的，或者客户 - 服务器方式的，即一个服务器机器为多个客户机执行工作。数据库系统还可以设计成具有能充分利用并行计算机系统结构的能力。分布式数据库跨越多个地理上分布的互相分离的计算机。
- 典型地，数据库应用可被分为运行在客户机上的前端和运行在后端的部分。在两层的体系结构中，前端直接和后端运行的数据库进行通信。在三层结构中，后端又被分为应用服务器和数据库服务器。 32
- 知识发现技术试图自动地从数据中发现统计规律和模式。**数据挖掘**(data mining)领域将人工智能和统计分析研究人员创造的知识发现技术，与使得知识发现技术能够在极大的数据库上高效实现的技术结合起来。
- 有 4 种不同类型的数据库用户，按用户期望与数据库进行交互的不同方式来区分他们。为不同类的用户设计了不同的用户界面。

术语回顾

- 数据库管理系统(DBMS)
- 数据库系统应用
- 文件处理系统
- 数据不一致性
- 一致性约束
- 数据抽象
- 实例
- 模式
  - □ 物理模式
  - □ 逻辑模式
- 物理数据独立性
- 数据模型
  - □ 实体 - 联系模型
  - □ 关系数据模型
  - □ 基于对象的数据模型
  - □ 半结构化数据模型
- 数据库语言
  - □ 数据定义语言
  - □ 数据操纵语言
  - □ 查询语言
- 元数据
- 应用程序
- 规范化
- 数据字典
- 存储管理器
- 查询处理器
- 事务
  - □ 原子性
  - □ 故障恢复
  - □ 并发控制
- 两层和三层数据库体系结构
- 数据挖掘
- 数据库管理员(DBA)

实践习题

1.1 这一章讲述了数据库系统的几个主要的优点。它有哪两个不足之处?

1.2 列出 Java 或 C++ 之类的语言中的类型说明系统与数据库系统中使用的数据定义语言的 5 个不同之处。

1.3 列出为一个企业建立数据库的六个主要步骤。 33

1.4 除 1.6.2 节中已经列出的之外，请列出大学要维护的至少 3 种不同类型的信息。

1.5 假设你想要建立一个类似于 YouTube 的视频节点。考虑 1.2 节中列出的将数据保存在文件系统中的各个缺点，讨论每一个缺点与存储实际的视频数据和关于视频的元数据(诸如标题、上传它的用户、标签、观看它的用户)的关联。

1.6 在 Web 查找中使用的关键字查询与数据库查询很不一样。请列出这两者之间在查询表达方式和查询结果是什么方面的主要差异。

习题

1.7 列出四个你使用过的很可能使用了数据库来存储持久数据的应用。

1.8 列出文件处理系统和 DBMS 的四个主要区别。

1.9 解释物理数据独立性的概念，以及它在数据库系统中的重要性。

1.10 列出数据库管理系统的五个职责。对每个职责，说明当它不能被履行时会产生什么样的问题。

1.11 请给出至少两种理由说明为什么数据库系统使用声明性查询语言，如 SQL，而不是只提供 C 或者 C++ 的函数库来执行数据操作。

1.12 解释用图 1-4 中的表来设计会导致哪些问题。

1.13 数据库管理员的五种主要作用是什么？

1.14 解释两层和三层体系结构之间的区别。对 Web 应用来说哪一种更合适？为什么？

1.15 描述可能被用于存储一个社会网络系统如 Facebook 中的信息的至少 3 个表。

工具

如今已有大量的商业数据库系统投入使用，主要的有：IBM DB2（www. ibm. com/software/data/db2）、Oracle（www. oracle. com）、Microsoft SQL Server（www. microsoft. com/sql）、Sybase（www. sybase. com）和 IBM
34 Informix（www. ibm. com/software/data/informix）。其中一些对个人或者非商业使用或开发是免费的，但是对实际的部署是不免费的。

也有不少免费/公开的数据库系统，使用很广泛的有 MySQL（www. mysql. com）和 PostgreSQL（www. postgresql. org）。

在本书的主页 www. db-book. com 上可以获得更多的厂商网址的链接和其他信息。

文献注解

我们在下面列出了关于数据库的通用书籍、研究论文集和 Web 节点。后续各章提供了本章略述的每个主题的资料参考。

Codd[1970]的具有里程碑意义的论文引入了关系模型。

关于数据库系统的教科书有 Abiteboul 等[1995]、O'Neil 和 O'Neil[2000]、Ramakrishnan 和 Gehrke[2002]、Date[2003]、Kifer 等[2005]、Elmasri 和 Navathe[2006]，以及 Garcia-Molina 等[2008]。涵盖事务处理的教科书有 Bernstein 和 Newcomer[1997]以及 Gray 和 Reuter[1993]。有一本书中包含了关于数据库管理的研究论文的汇集，这本书是 Hellerstein 和 Stonebraker[2005]。

Silberschatz 等[1990]、Silberschatz 等[1996]、Bernstein 等[1998]，以及 Abiteboul 等[2003]给出了关于数据库管理已有成果和未来研究挑战的综合评述。ACM 的数据管理兴趣组的主页（www. acm. org/sigmod）提
35 ~ 36 供了关于数据库研究的大量信息。数据库厂商的网址（参看上面的工具部分）提供了他们各自产品的细节。

据库缓冲区，系统将这些数据存放在共享内存中。除了处理查询的进程，还有执行诸如锁和日志管理以及检查点等任务的系统进程。

- □ 数据服务器系统提供给用户的是未加工的数据。这样的系统通过把数据和锁高速缓存在客户端，来努力使客户端和服务器之间的通信最小化。并行数据库系统使用类似的优化。

- 并行数据库系统由通过高速互连网络连接在一起的多台处理器和多张硬盘构成。加速比衡量通过增加并行性可以得到的对单个事务的处理速度的增长。扩展比衡量通过增加并行性可以得到的处理大量事务的能力。干扰、偏斜和启动代价是得到理想的加速比和扩展比的障碍。
- 并行数据库体系结构包括共享内存、共享硬盘、无共享以及层次的体系结构。这些体系结构在可扩展性以及通信速度方面各有千秋。
- 791 分布式数据库系统是部分独立的一组数据库系统，它们共享一个公共模式(理想情况下)，并且协调地处理访问非本地数据的事务。系统之间通过通信网络来相互通信。
- 局域网连接分布在小的地理范围内的结点，比如连接单个建筑或几个相邻建筑。广域网连接分布在大的地理范围内的结点。现在 Internet 是使用最广泛的广域网。
- 存储区域网是一种特殊形式的局域网，是为大型存储设备和多台计算机之间提供快速互连而设计的。

术语回顾

- 集中式系统
- 服务器系统
- 粗粒度并行
- 细粒度并行
- 数据库进程结构
- 互斥
- 线程
- 服务器进程
  - □ 锁管理进程
  - □ 数据库写进程
  - □ 日志写进程
  - □ 检查点进程
  - □ 进程监视进程
- 客户 - 服务器系统
- 查询服务器
- 数据服务器
  - □ 预读取
  - □ 逐步降级
  - □ 数据高速缓冲存储
  - □ 缓存一致性
- 792 □ 锁高速缓冲存储
  - □ 收回
- 并行系统
- 吞吐量
- 响应时间
- 加速比
  - □ 线性加速比
  - □ 亚线性加速比
- 扩展比
  - □ 线性扩展比
  - □ 亚线性扩展比
  - □ 批量型扩展比
  - □ 事务型扩展比
- 启动代价
- 干扰
- 偏斜
- 互连网络
  - □ 总线
  - □ 网格
  - □ 超立方体
- 并行数据库体系结构
  - □ 共享内存
  - □ 共享硬盘(集群)
  - □ 无共享
  - □ 层次的
- 容错性
- 分布式虚拟存储器
- 非一致性内存体系结构(NUMA)
- 分布式系统
- 分布式数据库
  - □ 站点(结点)
  - □ 局部事务
  - □ 全局事务
  - □ 局部自治性
- 多数据库系统
- 网络类型
  - □ 局域网(LAN)
  - □ 广域网(WAN)
  - □ 存储局域网(SAN)

实践习题

17.1 将共享结构存储在一个专用进程的本地内存中，而不是存储在共享内存中，通过与该进程间的通信存取共享数据。这种体系结构的缺陷是什么?

17.2 在典型的客户 - 服务器系统中，服务器机器比客户机的能力要强得多。也就是说，其处理器速

度更快，可能有多个处理器，有更大的内存和磁盘容量。请考虑另外一种设想，其中客户机和服务器机器能力相同。构建这样一个客户 - 服务器系统有意义吗？为什么？哪一种设想更适合于数据服务器体系结构？

17.3　考虑一个基于客户 - 服务器体系结构的数据库系统，其服务器是一个数据服务器。

a. 客户和服务器间的互连速度对于选择是进行元组传送还是进行页面传送有什么影响？

b. 如果采用页面传送，数据在客户端的高速缓存可以组织成元组缓存或者页面缓存。页面缓存以页面为单位存储数据，而元组缓存以元组为单位存储数据。假设元组比页面小，请描述元组缓存比页面缓存优越的一个地方。

17.4　假设一个事务是将 SQL 嵌入到 C 中书写的，大约 80% 的时间花在运行 SQL 代码上，剩余 20% 的时间花在运行 C 代码上。如果仅仅对 SQL 代码实施并行，那么可以期望得到多大的加速比？说明理由。

17.5　一些数据库操作，比如连接操作，在数据（例如，连接涉及的其中一个关系表中的数据）是存放在内存中或者不是存放在内存中这两种情况下，会出现速度上的明显差异。请说明这个事实如何解释了**超线性加速比**（superlinear speedup）现象，这里一个应用程序的加速比高于分配给它的资源数量的增长。

17.6　并行系统通常有这样的网络结构：多个包含 n 个处理器的集合连接到单个以太网交换机上，而这些以太网交换机本身又连接到另一台以太网交换机上。这个架构是否相当于总线、网格或超立方体架构？如果不，你会如何描述这种互连架构？

习题

17.7　如果不需要对单个查询进行并行化，为什么将数据库系统从单处理器机器移植到多处理器机器相对比较容易？ 793

17.8　对于处理短事务的客户 - 服务器关系数据库，常采用事务服务器体系结构。而对于处理较长事务的面向对象数据库系统，常采用数据服务器体系结构。请给出两条理由，解释为什么数据服务器适合于面向对象数据库而不适合于关系数据库。

17.9　什么是锁逐步降级？什么情况下需要锁逐步降级？为什么当数据传送的单位是数据项时不需要锁逐步降级？

17.10　假设你负责一个公司的数据库的运行，主要任务是处理事务。假设该公司每年都在迅速增长，大到使得公司当前的计算机系统已不再适用。当你选择一台新的并行计算机时，你最关注加速比、批量型扩展比，还是事务型扩展比？为什么？

17.11　典型的数据库系统由一组共享一个共享内存区域的进程（或线程）实现。

a. 如何控制对于共享内存区域的存取？

b. 两阶段锁协议适用于串行化访问共享内存中的数据结构吗？请解释你的答案。

17.12　允许用户进程访问数据库系统的共享内存区域是否明智？请解释你的答案。

17.13　在事务处理系统中，对于线性加速比起负面影响的有哪些因素？对于共享内存、共享硬盘、无共享这几种体系结构中的每一种，分别说明哪个因素可能是最重要？

17.14　内存系统可以分为多个模块，在一个特定时刻，每个模块可以处理一个独立的请求。这种存储架构会对共享内存系统所支持的处理器数量产生什么影响？

17.15　考虑一个银行，它有若干站点，每个站点运行一个数据库系统。假设这些数据库之间唯一的交互方式是利用持久消息进行电子转账，这样的系统称得上是分布式数据库吗？为什么？

文献注解

Hennessy 等[2006]对计算机架构领域提供了极好的介绍，Abadi[2009]提供了对云计算和在该环境中运行数据库事务所面临的挑战的非常好的介绍。

Gray 和 Reuter[1993]是一本描述事务处理的教科书，包括对客户 - 服务器体系结构和分布式系统

的描述。第5章的文献注解提供了有关ODBC、JDBC以及其他数据库访问API的更多信息的参考文献。

DeWitt和Gray[1992]对并行数据库系统进行了综述，包括其体系结构和性能度量。Duncan[1990]对并行计算机体系结构进行了综述。Dubois和Thakkar[1992]是一本关于可伸缩共享内存体系结构的文集。运行着Rdb的DEC集群是共享磁盘数据库体系结构的早期商业用户之一。Rdb现在属于Oracle，命名为Oracle Rdb。Teradata数据库机器是最早使用无共享数据库体系结构的商用系统之一。Grace和Gamma研究原型也使用了无共享体系结构。

Ozsu和Valduriez[1999]是介绍分布式数据库系统的教科书。关于并行和分布式数据库系统的进一步的参考文献在第18章和第19章的文献注解中分别列出。

794
|
796

Comer[2009]、Halsall[2006]和Thmoas[1996]描述了计算机网络和互联网。Tanenbaum[2002]、Kurose和Ross[2005]、Peterson和Davie[2007]对计算机网络进行了一般性概述。

提供了更快的存取时间。在多级高速缓存设计中，高速缓存分成 L_1、L_2 等级别，其中 L_1 是最快的高速缓存(因此每字节的价格也最昂贵，所以容量也最小)，L_2 是第二快的，依此类推。其结果是将第 10 章讨论的存储体系架构扩展到在主存下包含不同级别的高速缓存。 817

尽管数据库系统可以控制在磁盘和主存之间传输数据，但电脑硬件仍可控制在各个级别的高速缓存之间以及在高速缓存与主存之间的数据传输。尽管缺乏这种直接控制，数据库系统的性能仍然受到高速缓存使用方式的影响。如果一个核需要访问的数据项不在高速缓存中，那么它必须从主存获取。因为主存比处理器要慢得多，所以当一个核等待来自主存的数据时，处理速度就可能会蒙受巨大损失。这种等待称为**高速缓存失效**(cache miss)。

计算机设计者试图减小高速缓存失效所带来的影响的一种方式是采用多线程(multithreading)。**线程**(thread)是一个执行流，它与运行在同一个核上的其他线程共享内存。[⊖]如果在一个核上当前执行的线程遇到高速缓存失效(或其他类型的等待)，那么核就会执行另外的线程，从而在等待时不浪费计算速度。

线程引了在多核之上的另一种并行性。每个新一代的处理器都支持更多的核和更多的线程。Sun UltraSPARC T2 处理器有 8 个核，每个核能支持 8 个线程，总的来说，在一块处理器芯片上能支持 64 个线程。

原始速度增长放缓同时核数量增长这样的体系架构上的趋势对数据库系统设计产生了重大影响，这点我们很快就会看到。

18.9.3 适应现代体系架构的数据库系统设计

可能看起来数据库系统是一个有效利用大量核和线程的理想应用，因为数据库系统支持大量并发事务。然而，有很多因素使得对现代处理器的优化利用充满了挑战。

当我们允许更高的并发度去利用现代处理器的并行性时，我们会增加高速缓存中的数据量。这可能会导致更多的高速缓存失效，甚至可能多到需要多线程的核来等待来自内存中的数据。

并发事务需要某种形式的并发控制来保证第 14 章讨论过的 ACID 属性。当并发事务访问相同数据时，必须对该并发访问施加某种限制。这些限制，无论是基于封锁、时间戳还是验证，都会导致等待或由于事务中止而损失工作。为了避免长时间等待或大量的工作损失，理想情况是并发事务的冲突很少，但试图确保减少冲突会增加需要存放到高速缓存中的数据量，并导致更多的高速缓存失效。

最后，存在一些被所有事务所共享的数据库系统组件。在使用封锁的系统中，锁表被所有事务共享，对锁表的访问可能会成为瓶颈。类似的问题也同样存在于其他形式的并发控制中。同样，服务于所有事务的缓冲管理器、日志管理器和恢复管理器也是潜在的瓶颈。 818

由于具有大量的并发事务可能不能最好地利用现代处理器的优势，因此需要找到允许多核在单个事务上工作的方式。这就要求数据库查询处理器找到有效的并行查询方式，而不会对高速缓存产生过多的需求。这可以通过建立数据库查询操作的流水线以及通过寻找单个数据库操作的并行化方式来实现。

适应于多核和多线程系统的数据库系统设计和数据库查询处理仍然是一个活跃的研究领域。更多细节请参见文献注解。

18.10 总结

- 在过去 20 年中，并行数据库已经得到了广泛的商业认同。
- 在 I/O 并行中，把关系划分到多张可用的磁盘中，从而使检索速度更快。三种常用的划分技术是轮转法划分、散列划分和范围划分。
- 偏斜是一个主要的问题，特别是当并行度增高时。平衡的划分向量、使用直方图以及虚处理器划分是用于减少偏斜的技术。
- 在查询间并行中，我们并发地运行不同的查询以提高吞吐量。

⊖ 在技术上，以操作系统的术语来讲，指其地址空间。

- 查询内并行试图减少运行查询的代价。有两类查询内并行：操作内并行和操作间并行。
- 我们采用操作内并行来并行地执行关系运算，例如排序和连接。因为关系运算是面向集合的，所以操作内并行对关系运算是很自然的。
- 对于像连接这样的二元运算，有两种基本的并行化方法。
  - □ 在基于划分的并行中，两个关系分成几个部分，而且 r_i 中的元组仅与 s_i 中的元组进行连接。基于划分的并行仅适用于自然连接和等值连接。
  - □ 在分片和复制中，两个关系都被划分，并且每个划分都被复制。在非对称的分片和复制中，一个关系被复制，而另一个关系被划分。与基于划分的并行不同，分片和复制以及非对称的分片和复制对于任何连接条件都适用。

819

  这两种并行技术都可以与任何一种连接技术结合使用。
- 在独立的并行中，互不依赖的多个不同的操作按并行方式执行。
- 在流水线并行中，处理器在计算一个操作结果的同时将结果发送给另一个操作，无须等待整个操作的完成。
- 并行数据库中的查询优化比串行数据库中的查询优化要复杂得多。
- 现代的多核处理器引入了并行数据库中新的研究问题。

术语回顾

- 决策支持查询
- I/O 并行
- 水平划分
- 划分技术
  - □ 轮转法
  - □ 散列划分
  - □ 范围划分
- 划分属性
- 划分向量
- 点查询
- 范围查询
- 偏斜
  - □ 执行偏斜
  - □ 属性值偏斜
  - □ 划分偏斜
- 偏斜处理
  - □ 平衡的范围划分向量
  - □ 直方图
  - □ 虚处理器
- 查询间并行
- 高速缓存一致性
- 查询内并行
  - □ 操作内并行
  - □ 操作间并行
- 并行排序
  - □ 范围划分排序
  - □ 并行的外部排序归并
- 数据并行
- 并行连接
  - □ 基于划分的连接
  - □ 分片 - 复制连接
  - □ 非对称的分片 - 复制连接
  - □ 基于划分的并行散列连接
  - □ 并行嵌套循环连接
- 并行选择
- 并行去重
- 并行投影
- 并行聚集
- 并行计算的代价
- 操作间并行
  - □ 流水线并行
  - □ 独立并行
- 查询优化
- 调度
- 交换算子模型
- 并行系统设计
- 联机索引创建
- 多核处理器

实践习题

18.1 当在范围划分属性上进行范围选择时，可能仅需要访问一张磁盘，请描述这一特性的优点和缺点。

18.2 对于下述每种任务，哪种并行形式(查询间并行、操作间并行或操作内并行)可能是最重要的?

a. 提高具有许多小查询的系统的吞吐量。

b. 在磁盘和处理器的数目都很大的情况下，提高具有少量大查询的系统的吞吐量。

18.3 对于流水线并行，即便当有许多处理器可用时，在单个处理器上用流水线来执行多个操作通常是个好主意。

a. 请解释原因。

b. 如果机器采用共享内存体系结构，你在 a 部分提出的论据还成立吗? 请解释成立或不成立的原因。

c. 对于独立的并行，a 部分提出的论据成立吗?（也就是说，即使在不将操作组织到流水线中，而且有许多处理器可用的情况下，在同一个处理器上执行几个操作是否仍然是个好主意呢?）

18.4 考虑采用范围划分的对称的分片和复制来处理连接。如果连接条件形如 $|r.A - s.B| \leqslant k$，其中 k 是一

个小常数，$|x|$表示取 x 的绝对值，带有这种连接条件的连接称作**带状连接**(band join)，那么你如何来优化查询计算？

18.5　回顾一下，可以利用直方图来建立负载均衡的范围划分。

a. 假设你有一个取值范围为 1 ~ 100 的直方图，划分为 10 个范围：1 ~ 10，11 ~ 20，…，91 ~ 100，其出现频率分别为 15，5，20，10，10，5，5，20，5，5。请给出一个负载均衡的范围划分函数，将这些值划分成 5 个部分。

820 / 821

b. 请写出一个算法，对于给定的包括 n 个范围的频率分布直方图，计算出划分为 p 个分区的一个均衡的范围划分。

18.6　大规模并行数据库系统对每个数据项都会在附属于不同处理器的磁盘上存储一个额外的副本，以避免当一个处理器失效时所引起的数据丢失。

a. 不是把来自一个处理器的数据项的额外副本保存到一个单点备份的处理器上，更好的方法是把一个处理器的数据项副本划分到多个处理器上。请解释原因。

b. 解释如何利用虚处理器划分来有效实施前述副本的划分。

c. 如果采用 RAID 存储，而不是存储每个数据项的额外副本，有哪些优点和缺点？

18.7　假设我们要对一个已经划分的关系进行索引。这种划分的思想(包括虚处理器划分)能够应用于索引吗？解释你的答案，考虑下面两种情况(为了简单起见，假设划分和索引都是在单个属性上进行的)。

a. 索引建立在关系的划分属性上。

b. 索引基于的属性不是关系的划分属性。

18.8　假设已经为一个关系选好了非常平衡的范围划分向量，但是这个关系随后更新了，导致划分失衡。即使采用虚处理器划分，某个虚处理器最终也可能在更新后得到相当大量的元组，从而需要重新划分。

a. 假设一个虚拟处理器有相当多的超量元组(比如说，超过平均值的两倍)。解释如何通过分裂分区从而增加虚处理器的数量来实现重新划分。

b. 如果虚处理器不是轮流分配的，虚拟分区可以以任意的方式分配给处理器，并使用映射表来记录分配信息。如果某个特定的节点过载(相比其他节点而言)，请解释如何均衡负载。

c. 假设没有更新，那么在执行重新划分或者重新分配虚处理器的过程中是否需要中断查询处理？解释你的答案。

822

习题

18.9　对于轮转法、散列划分和范围划分这三种划分技术，请各给出一个查询实例，使得应用该划分技术能提供最快的响应。

18.10　当关系在它的一个属性上采用：

- 散列划分
- 范围划分

进行划分时，哪些因素会导致偏斜？对于上述两种情况，分别可以采取什么措施来减小偏斜？

18.11　请给出一个连接的实例，它不是简单的等值连接，但可以采用基于划分的并行。应该基于什么属性来进行划分？

18.12　对于下述每种运算，给出一种好的并行化方法。

a. 差运算。

b. 使用 **count** 运算的聚集。

c. 使用 **count distinct** 运算的聚集。

d. 使用 **avg** 运算的聚集。

e. 左外连接，其连接条件只涉及相等比较。

f. 左外连接，其连接条件涉及相等之外的比较。

g. 完全外连接，其连接条件涉及相等之外的比较。

18.13　请描述流水线并行的优点和缺点。

18.14　假设你希望使用无共享体系结构的并行来处理一个由大量小事务组成的工作负载。

a. 在这样的情形中需要查询内并行吗？如果不需要，为什么？哪种形式的并行适用呢？

b. 对于这样的工作负载，哪种形式的偏斜会很显著呢？

c. 假设大多数事务都会访问一条 *account* 记录，该记录包括一个 *account_type* 属性，以及一条对应的
823 *account_type_master* 记录，它提供了有关账户类型的信息。你会如何划分和(或)复制数据以加速事务的执行？你可以假定 *account_type_master* 关系极少会更新。

18.15 对关系进行划分所基于的属性可以对查询代价产生很大的影响。

a. 给定在单个关系上的一个 SQL 查询工作负载，什么属性会作为划分的候选属性？

b. 基于这样的工作负载，你会如何在不同的划分技术中进行选择？

c. 有可能在关系的不止一个属性上进行划分吗？请解释你的答案。

文献注解

在 20 世纪 70 年代末和 80 年代初，当关系模型取得了相当稳固的地位时，人们意识到关系运算是高度可并行的，而且具有很好的数据流特性。因而发起了包括 GAMMA(DeWitt 等[1990])、XPRS(Stonebraker 等[1989])和 Volcano(Graefe [1990])在内的几个研究项目来研究关系运算并行执行的实用性。

Teradata 是最早的商用并行数据库系统之一，并继续占有着一个很大的市场份额。Red Brick 数据仓库是另一个早期的并行数据库系统，后来 Red Brick 被 Informix 收购，而 Informix 自己又被 IBM 收购。更新的并行数据库系统包括 Netezza、DATAllegro(现为微软的一部分)、Greenplum 和 Aster Data。

Joshi[1991]以及 Mohan 和 Narang[1992]讨论了并行数据库中的封锁问题。Dias 等[1989]、Mohan 和 Narang[1992]、Rahm[1993]讨论了并行数据库系统的高速缓存一致性协议。Carey 等[1991]讨论了客户 - 服务器系统中的高速缓存问题。

Graefe 和 McKenna[1993b]对于查询处理(包括查询的并行处理)给了一个极好的综述。交换算子模型是 Graefe[1990]以及 Graefe 和 McKenna[1993b]提出的。

DeWitt 等[1992]讨论了并行排序。Garcia 和 Korth[2005]以及 Chen 等[2007]讨论了基于多核和多线程处理器的并行排序。Nakayama 等[1984]、Richardson 等[1987]、Kitsuregawa 和 Ogawa[1990]、Wilschut 等[1995]，以及其他文献讨论了并行连接算法。

Walton 等[1991]、Wolf[1991]、Dewitt 等[1992]描述了并行连接中的偏斜处理。

Lu 等[1991]、Ganguly 等[1992]描述了并行查询优化技术。

自 2005 年以来，每年举办一次现代硬件上的数据管理(Data Management on Modern Hardware，DaMoN)
824 国际研讨会，其论文集讨论了适用于多核和多线程体系架构的数据库系统设计和查询处理算法。

LDAP 中对复制的标准化工作正在进行中。

19.11 总结

- 分布式数据库系统由站点的集合构成，每个站点维护一个本地数据库系统。各个站点能够处理局部事务：这些事务访问的数据仅位于该单个站点上。此外，站点可以参与到全局事务的执行中：这些全局事务访问多个站点上的数据。全局事务的执行需要在站点之间进行通信。
- 分布式数据库可能是同构的，其中所有站点拥有共同的模式和数据库系统代码，或者是异构的，其中模式和系统代码可能不同。
- 关于在分布式数据库中存储关系涉及几个问题，包括复制和分片。系统应尽量减小用户需要了解关系如何存储的程度，这是非常重要的。

875

- 分布式系统可能遭受与集中式系统相同类型的故障。但是，分布式环境中还有另一些需要处理的故障，包括站点故障、链路故障、消息丢失以及网络划分。在分布式故障恢复模式的设计中需要考虑每个这样的问题。
- 为了保证原子性，执行事务 T 的所有站点必须在执行的最终结果上取得一致。T 要么在所有站点上提交，要么在所有站点上中止。为了保证这一特性，T 的事务协调器必须执行一种提交协议。使用最广泛的提交协议是两阶段提交协议。
- 两阶段提交协议可能导致阻塞，在这种情况下，事务的命运必须等到故障站点(协调器)恢复后才能确定。为了减少阻塞的可能性，可以使用三阶段提交协议。
- 持久消息为分布式事务处理提供了一种可选模式。该模式将单个事务拆分成在不同数据库执行的多个部分。持久消息(无论是否发生故障，都保证正好只传送一次)被传送到需要采取动作的远程站点。虽然持久消息避免了阻塞问题，但是应用程序开发者必须编写代码来处理各种类型的故障。
- 在集中式系统中使用的各种并发控制方案修改后可用于分布式环境。
  - □ 就封锁协议而言，须做的唯一改变是锁管理器的实现方式。这里有各种不同的方式。可以采用一个或多个中央协调器。而如果采用分布式锁管理器方式，复制的数据就必须特殊对待。
  - □ 处理已复制数据的协议包括主副本协议、多数协议、有偏协议和法定人数同意协议。它们在开销方面和在发生故障时工作的能力方面各有不同的取舍权衡。
  - □ 就时间戳和有效性验证方案而言，所需的唯一修改是开发一种产生全局唯一性时间戳的机制。
  - □ 许多数据库系统支持延迟复制，其中更新被传播到执行更新的事务的范围之外的副本。这样的工具必须小心使用，因为它们可能导致不可串行化的执行。

876

- 分布式锁管理器环境中的死锁检测需要多个站点之间的合作，因为甚至在没有局部死锁的情况下也可能有全局死锁。
- 为了提供高可用性，分布式数据库必须检测故障，重构系统以使计算能够继续进行，并在处理器或链路修复之后能够恢复。由于要在网络划分和站点故障之间进行区分是很困难的，因此这个任务就变得非常复杂。

  通过使用版本号，可以对多数协议进行扩展使其在即使存在故障的情况下仍允许进行事务处理。虽然该协议代价昂贵，但它无论在何种类型的故障下都能工作。可以使用较小代价的协议来处理站点故障，但是它们假设不会发生网络划分。
- 一些分布式算法需要使用协调器。为了提供高可用性，系统必须维护一个准备好在协调器故障时能继续其职责的备份副本。另一种方法是在协调器发生故障后选出新的协调器。确定哪个站点应该作为协调器的算法称为**选举算法**。
- 分布式数据库上的查询可能需要访问多个站点。可以使用几种优化技术来识别需要访问的最佳站点集。查询可以依据关系的分片来自动重写，然后可以在每个分片的副本之间做出选择。可以应用半连接技术减少跨不同站点的关系(或相应的分片或副本)连接中所涉及的数据传输。
- 异构分布式数据库允许站点有它们自己的模式和数据库系统代码。多数据库系统提供了一种环境，在其中新的数据库应用可以访问位于多种异构软硬件环境的各个先前存在数据库中的数

据。局部数据库系统可以采用不同的逻辑模型以及数据定义和数据操纵语言，并且可以在它们的并发控制和事务管理机制上存在差别。多数据库系统虚拟了逻辑上的数据库集成，不需要物理上的数据库集成。

- 为了响应超大规模 Web 应用对数据存储的需求，近年来在云上构建了大量数据存储系统。这些数据存储系统允许扩展到地理上分布的数千个结点上，而且具有高可用性。然而，它们并不支持通常的 ACID 特性，而且在划分时以副本一致性为代价来获得可用性。当前的数据存储系统还不支持 SQL，而且只提供简单的 **put()/get()** 接口。即使对于传统数据库云计算也是有吸引力的，但因为缺乏对数据存放和地理副本的控制，也面临一些挑战。
- 目录系统可视为一种特殊形式的数据库，其中信息按照一种分层的方式组织，类似于文件系统中文件的组织方式。目录通过标准化目录访问协议(例如 LDAP)来访问。目录可以分布到多个站点上来提供对各个站点的自治。目录可以包含对其他目录的引用，这有助于建立集成视图，借此查询被发送给单个目录，并且在所有相关目录上透明地执行。 877

术语回顾

- 同构分布式数据库
- 异构分布式数据库
- 数据复制
- 主副本
- 数据分片
  - 水平分片
  - 垂直分片
- 数据透明性
  - 分片透明性
  - 复制透明性
  - 位置透明性
- 名字服务器
- 别名
- 分布式事务
  - 局部事务
  - 全局事务
- 事务管理器
- 事务协调器
- 系统故障模式
- 网络划分
- 提交协议
- 两阶段提交协议(2PC)
  - 就绪状态
  - 疑问事务
  - 阻塞问题
- 三阶段提交协议(3PC)
- 持久消息
- 并发控制
- 单锁管理器
- 分布式锁管理器
- 副本协议
  - 主副本
  - 多数协议
  - 有偏协议
  - 法定人数同意协议
- 时间戳
- 主从复制
- 多主(任意地方更新)复制
- 事务一致性快照
- 延迟传播
- 死锁处理
  - 局部等待图
  - 全局等待图
  - 假环
- 可用性
- 健壮性
  - 基于多数的方法
  - 读一个、写所有
  - 读一个、写所有可用
  - 站点重构
- 协调器选择
- 备份协调器
- 选举算法
- 威逼算法
- 分布式查询处理
- 半连接策略
- 多数据库系统
  - 自治
  - 中间件
  - 局部事务
  - 全局事务
  - 确保全局可串行化
  - 标签
- 云计算
- 云数据存储
- 表块
- 目录系统
- LDAP：轻量级目录访问协议
  - 可区别名称(DN)
  - 相对可区别名称(RDN)
  - 目录信息树(DIT)
- 分布式目录树
  - DIT 前缀
  - 引用

实践习题

19.1　为局域网设计的分布式数据库与为广域网设计的分布式数据库会有哪些区别?

19.2　要建立一个高可用性的分布式系统，你必须知道会发生什么类型的故障。

a. 列出分布式系统中可能的故障类型。

b. 在你的 a 小题列表中的哪些故障也可能出现在集中式系统中?

19.3 考虑在事务的2PC中发生的故障。对你在实践习题19.2a中列出的每种可能的故障，解释尽管存在故障，2PC怎样保证事务的原子性。

19.4 考虑一个有两个站点 A 和 B 的分布式系统。站点 A 能区别出下列情况吗？

- B 崩溃。
- A 和 B 之间的连接断开。
- B 极度过载，并且响应时间比正常情况下长100倍。

878 ~ 879

你的答案对于分布式系统中的恢复有何启示？

19.5 本章描述的持久消息模式依赖于时间戳，并结合了丢弃接收到的过于陈旧的消息。给出一个基于序列号而不是时间戳的替代方案。

19.6 给出读一次、写所有可用方法导致错误状态的一个例子。

19.7 解释在分布式系统中的数据复制和维护远程备份站点之间的差异。

19.8 给出一个即使更新时得到了主要的(主)副本上的排他锁，延迟复制仍能够导致数据库状态不一致的例子。

19.9 考虑下面的死锁检测算法。当站点 S_1 上的事务 T_i 请求站点 S_3 上来自事务 T_j 的资源时，发送带时间戳 n 的请求消息。边(T_i, T_j, n)被插入到 S_1 的局部等待图中。仅当 T_j 已接收到请求消息但不能立即授予所请求的资源时，才把边(T_i, T_j, n)插入到 S_3 的局部等待图中。同一站点上从事务 T_i 到 T_j 的请求按照通常的方式处理；边(T_i, T_j)上不与时间戳相关联。中央协调器通过对系统中的每个站点发送初始化消息来调用检测算法。

一旦接收到这样的消息，站点就发送它自己的局部等待图给协调器。注意此图包含该站点所拥有的关于真实图状态的所有局部信息。等待图反映了站点的瞬间状态，但是它没有与任何其他站点同步。当控制者从每个站点都接收到回答后，它构造如下的图：

- 对于系统中的每个事务，该图包含一个顶点。
- 该图有边(T_i, T_j)当且仅当：
  - □ 在某个等待图中有边(T_i, T_j)。
  - □ 边(T_i, T_j, n)(对某些 n)在不止一个等待图中出现。

证明，如果在构造的图中有环，那么系统处于死锁状态，并且如果在构造的图中没有环，那么系统在该算法开始执行时没有处于死锁状态。

19.10 考虑按 *plant_number* 水平分片的关系：

880

employee (*name*, *address*, *salary*, *plant_number*)

假设每个分片有两个副本：一个存在New York站点，且另一个存在本地的工厂站点。为下述在San Jose站点提出的查询设计一种好的处理策略。

a. 找出Boca厂的所有员工。

b. 找出所有员工的平均工资。

c. 找出在以下每个站点薪酬最高的员工：Toronto、Edmonton、Vancouver、Montreal。

d. 找出公司中薪酬最低的员工。

19.11 对图19-9中的关系计算 $r \ltimes s$。

A	B	C
1	2	3
4	5	6
1	2	4
5	3	2
8	9	7

r

C	D	E
3	4	5
3	6	8
2	3	2
1	4	1
1	2	3

s

图19-9 实践习题19.11中的关系

19.12 给一个在云上的理想化的应用程序的例子，再给另一个很难在云上成功实现的例子，解释你的答案。

19.13　假设 LDAP 的功能可以在数据库系统之上实现，那么 LDAP 标准需要什么？

19.14　考虑一个多数据库系统，它保证在任何时候最多只有一个全局事务是活跃的，并且每个本地站点保证局部可串行化。

a. 给出多数据库系统可以保证在任何时候最多只有一个活跃的全局事务的方法。

b. 举例说明尽管有上述假设，还是有可能导致不可串行化的全局调度。

19.15　考虑一个多数据库系统，其中每个本地站点保证局部可串行化，并且所有全局事务都是只读的。

a. 举例说明在这样的系统中可能产生不可串行化的执行。 881

b. 说明你如何使用标签机制来保证全局可串行化。

习题

19.16　讨论集中式数据库和分布式数据库的相对优点。

19.17　解释下述说法有何区别：分片透明性、复制透明性、位置透明性。

19.18　数据复制和分片何时有用？解释你的答案。

19.19　解释透明性和自治的概念。为什么从人的因素的角度来看这些概念是需要的？

19.20　如果我们把第 15 章中多粒度协议的分布式版本应用于分布式数据库，负责 DAG 根结点的站点可能成为瓶颈。假设我们修改协议如下：

- 根结点上只允许有意向模式的锁。
- 所有事务被自动授予根结点上所有可能的意向模式锁。

说明这些修改可以无须允许任何非串行化调度来减轻这个问题。

19.21　研究并总结你正使用的数据库系统为处理由于更新的延迟传播而导致的不一致性状态而提供的工具。

19.22　讨论 19.5.2 节介绍的用来产生全局唯一性时间戳的两种方法的优缺点。

19.23　考虑关系：

$$employee(name,\ address,\ salary,\ plant\_number)$$
$$machine(machine\_number,\ type,\ plant\_number)$$

假设 *employee* 关系按 *plant_number* 水平分片，且每个片片本地存储在它所对应的工厂站点。假设整个 *machine* 关系存在 Armonk 站点。为以下每个查询设计一种好的处理策略。

a. 找出包含机器号 1130 的工厂的所有员工。

b. 找出包含机器类型为“milling machine”的工厂的所有员工。

c. 找出 Almaden 工厂的所有机器。 882

d. 找出 employee ⋈ machine。

19.24　对习题 19.23 的每种策略，说明你选择的策略如何依赖于下列因素：

a. 查询提出的站点。

b. 需要结果的站点。

19.25　表达式 $r_i \ltimes r_j$ 必然等于 $r_j \ltimes r_i$ 吗？在什么条件下 $r_i \ltimes r_j = r_j \ltimes r_i$？

19.26　如果使用云数据存储服务来存储两个关系 r 和 s，而且我们需要对 r 和 s 进行连接，为什么将此连接作为物化视图来维护可能是有用的？在你的答案中，务必区分出“有用”的各种含义：整体吞吐量、空间使用效率，以及对用户查询的响应时间。

19.27　为什么云计算服务通过使用虚拟机的方式能最好地支持传统数据库系统，而不是通过直接在服务供应商的实际机器上运行的方式？

19.28　描述如何使用 LDAP 来提供对数据的多层分层视图，而不用复制基础级数据。

文献注解

教材中关于分布式数据库的讨论由 Ozsu 和 Valduriez[1999]提供。Breitbart 等[1999b]提供了对分

布式数据库的概述。

分布式数据库中事务概念的实现由 Gray[1981]和 Traiger 等[1982]给出。2PC 协议由 Lampson 和 Sturgis[1976]开发。三阶段提交协议来自 Skeen[1981]。Mohan 和 Lindsay[1983]讨论了 2PC 的两种修改版，叫做假设提交和假设中止，它们通过考虑事务命运、定义默认假设来减少 2PC 的开销。

19.6.5 节中的威逼算法来自 Garcia-Molina[1982]。分布式时钟同步在 Lamport[1978]中讨论。分布式并发控制由 Bernstein 和 Goodman[1981]介绍。

R* 的事务管理器在 Mohan 等[1986]中描述。分布式并发控制模式的验证技术由 Schlageter[1981]和 Bassiouni[1988]描述。

Gray 等[1996]在数据仓库环境中再讨论了复制数据的并发更新问题。Anderson 等[1998]讨论了关于延迟复制和一致性的问题。Breitbart 等[1999a]描述了用于处理复制的延迟更新协议。

各种数据库系统的用户手册提供了它们如何处理复制和一致性的详细信息。Huang 和 Garcia-Molina[2001]阐述了复制消息系统中恰好一次的语义。

Knapp[1987]综述了分布式死锁检测的文献。实践习题 19.9 来自 Stuart 等[1984]。

Epstein 等[1978]、Hevner 和 Yao[1979]讨论了分布式查询处理。Daniels 等[1982]讨论了 R* 采用的分布式查询处理方法。

Ozcan 等[1997]阐述了多数据库中的动态查询优化。Adali 等[1996]和 Papakonstantinou 等[1996]描述了中间件系统中的查询优化问题。

Mehrotra 等[2001]讨论了多数据库系统中的事务处理。Georgakopoulos 等[1994]介绍了标签方案。Mehrotra 等[1991]介绍了 2LSR。

在 Ooi 和 S. Parthasarathy[2009]中有一组关于云系统上数据管理的论文。Chang 等[2008]描述了 Google 的 Bigtable 的实现，而 Cooper 等[2008]描述了雅虎的 PNUTS 系统。Brantner 等[2008]介绍了使用 Amazon 的 S3 基于云的存储来构建数据库的经验。Lomet 等[2009]介绍了在云系统中使事务正确工作的方法。Brewer[2000]推测了 CAP 定理，并由 Gilbert 和 Lynch[2002]形式化及证明。

883
~
884

Howes 等[1999]提供了涵盖 LDAP 的教材。

电子商务有关。它们包括非盈利团体发布的标准以及企业为建立事实标准而付出的努力。

RosettaNet 属于前者，它是一个工业协会利用基于 XML 的标准来帮助计算机与信息技术产业中的供应链管理。供应链管理指购买机构运行所需要的材料和服务。相反，顾客关系管理指公司交互的前端，是与客户打交道的。供应链管理要求很多事情的标准化，例如：

- **全局公司标识**(global company identifier)：RosettaNet 指明唯一识别公司的系统，使用 9 位数字的标识符，称作*数据通用编号方式系统*(DUNS)。
- **全局产品标识**(global product identifier)：RosettaNet 指明了一个 14 位数字的*全局商业项目编号*(GTIN)，用于标识产品和服务。
- **全局类别标识**(global class identifier)：这是一个称作*联合国/标准产品和服务码*(UN/SPSC)的 10 位数字层次编码，用于对产品和服务进行分类。
- **贸易伙伴之间的接口**(interface between trading partners)：RosettaNet *伙伴接口过程*(PIP)定义了伙伴之间的商业过程。PIP 是基于 XML 的系统对系统的会话：它们定义了处理过程所涉及的商业文档格式和语义以及完成事务所采取的步骤。作为例子，这些步骤可能包括获得产品和服务的信息、购货订单、订单货品计价、付款、订单状态请求、存货管理、包括服务担保在内的售后支持等。设计、配置、处理过程和质量信息的交换还有可能协调跨机构的制造活动。

电子市场的参与者可能将数据存储在多种数据库系统中。这些系统可能使用不同的数据模型、数据格式和数据类型。此外，数据间可能存在语义差异(公制对英制，不同流通货币，等等)。电子市场的标准包括使用 XML 模式*包装*每个这样的异构系统的方法。这些 XML *包装器*为分布在所有市场参与 1055
方的数据建立了统一视图的基础。

简单对象访问协议(Simple Object Access Protocol，SOAP)是一种远程过程调用标准，它使用 XML 编码数据(参数和结果)，利用 HTTP 作为传输协议。这样，函数调用就成为一个 HTTP 请求。SOAP 由万维网联盟(W3C)支持，并且获得了产业界的广泛支持。SOAP 可用于各种应用。例如，在 B2B 的电子商务中，在某个站点上运行的应用程序可以通过 SOAP 访问其他站点上的数据并执行动作。

在 23.7.3 节已经详细介绍了 SOAP 和 Web 服务。

24.5　总结

- 调整数据库系统参数和更高级别的数据库设计(如模式、索引和事务)对于实现高性能至关重要。查询可以进行调整以提高集合面向性，而批量加载功能可以大大加快数据导入到数据库中的速度。

  调整的最好办法是确定瓶颈所在，然后消除瓶颈。数据库系统通常有多种可调参数，如缓冲区大小、内存大小和磁盘数量。可以选择适当的索引和物化视图集合，以使总体代价达到最小。可以调整事务使锁竞争达到最小。快照隔离和支持早期锁释放的序号编号功能是减少读写和写写竞争的有用工具。
- 性能基准程序在对数据库系统进行比较方面扮演了重要的角色，尤其在数据库系统变得越来越与标准兼容时。TPC 基准程序集使用广泛，不同的 TPC 基准程序可以用于不同工作负载下的数据库系统性能的比较。
- 应用程序在开发时和部署前需要进行大量的测试。测试是用来捕获错误的，以及确保达到性能目标。
- 遗产系统是基于老一代技术(如非关系数据库或甚至直接基于文件系统)的系统。当运行关键任务系统时，遗产系统与新一代系统之间的连接通常是很重要的。从遗产系统到新一代系统的移植必须非常小心以避免破坏，这种移植是非常昂贵的。
- 由于数据库系统的复杂性和互操作的需要，标准对数据库系统来说很重要。SQL 有其正式标准。事实标准(如 ODBC 和 JDBC)和被行业组织所采纳的标准(如 CORBA)，在客户－服务器数据库系统的发展中发挥了重要作用。 1056

术语回顾

- 性能调整
- 面向集合
- 批处理更新(JDBC)

- 批量加载
- 批量更新
- 合并语句
- 瓶颈
- 队列系统
- 可调参数
- 硬件调整
- 五分钟规则
- 一分钟规则
- 模式调整
- 索引调整
- 物化视图
- 立即视图维护
- 延迟视图维护
- 事务调整
- 锁竞争
- 序号
- 小型批处理事务
- 性能模拟
- 性能基准程序
- 服务时间
- 完成时间
- 数据库应用类型
- TPC 基准程序
  - □ TPC-A
  - □ TPC-B
  - □ TPC-C
  - □ TPC-D
  - □ TPC-E
  - □ TPC-H
- 每秒的网络交互数
- 回归测试
- 消灭突变
- 遗产系统
- 逆向工程
- 工程再设计
- 标准化
  - □ 正式标准
  - □ 事实标准
  - □ 预见标准
  - □ 反应标准
- 数据库连接标准
  - □ ODBC
  - □ OLE-DB
  - □ X/Open XA 标准
- 对象数据库标准
  - □ ODMG
  - □ CORBA
- 基于 XML 的标准

实践习题

24.1 很多应用程序需要产生每个事务的序列号。

1057 a. 如果一个序号计数器采用两段封锁协议，可能成为一个并发瓶颈。解释产生这种情况的原因。

b. 很多数据库系统支持内置的序号计数器，采用非两阶段封锁协议；当一个事务请求一个序号时，计数器上锁，累加，然后解锁。

i. 解释这样的计数器如何提高并发度。

ii. 解释在最终提交事务的序号之间存在空缺的原因。

24.2 假设给定一个关系 $r(a, b, c)$。

a. 给出一个例子，说明什么情况下在属性 a 上的等值选择查询的性能会受关系 r 是如何聚集的影响很大。

b. 假设还有在属性 b 上的范围选择查询。r 如何聚集会使 $r.a$ 上的等值选择查询和 $r.b$ 上的范围选择查询都能高效地回答？请解释你的答案。

c. 如果上述聚集方法是不可能的，给出一个建议，如何通过选择合适的索引使两种类型查询都能高效执行，假设你的数据库支持只有索引的计划(即如果一个查询请求的所有信息在一个索引上都可获得，那么数据库能产生一个只使用索引而不需访问关系的执行计划)。

24.3 假设数据库应用程序看起来没有瓶颈，即 CPU 和磁盘使用率都较高，且所有数据库队列大致平衡。那是否意味着应用程序就不能进一步调整性能？解释你的答案。

24.4 假设一个系统运行三种类型的事务。A 类事务以 50/s 的速度运行，B 类事务以 100/s 的速度运行，C 类事务以 200/s 的速度运行，假设混合事务中包含 25% 的 A 类事务，25% 的 B 类事务，50% 的 C 类事务。

a. 假设事务之间没有干扰，系统的平均事务吞吐率是多少？

b. 什么因素会导致不同类型的事务之间的干扰，以至于计算得到的吞吐率不正确？

24.5 列出预见标准和反应标准相比的优点和缺点。

习题

1058 24.6 找出你特别喜爱的数据库提供的所有性能信息。至少找出以下这些：有哪些当前正在执行或者最近执行过的查询，它们各消耗了哪些资源(CPU 和 I/O)，哪些页面片段的请求导致了缓冲区遗漏(可能的话针对每个查询)，哪些锁被高度争夺。你还可以从操作系统上获得关于 CPU 和 I/O 使用率的信息。

24.7　a. 调节数据库系统的哪三个主要层次可以提高性能?

b. 对每个层次请举出两个例子说明调节是如何进行的。

24.8　当进行性能调整时，应该首先调整硬件(通过增加磁盘或者内存)，还是应该首先调整事务(通过增加索引或者物化视图)。解释你的答案。

24.9　假设你的应用程序有这样的事务：每个事务访问并更新存储在 B^+ 树文件组织的大关系中的单个元组。假定 B^+ 树的所有内部结点都在内存中，但只有非常少量的叶子页面能放进内存。解释如何计算支持每秒钟 1000 个事务的工作负载最少需要的磁盘数。利用 10.2 节给出的磁盘参数值，计算需要的磁盘数。

24.10　将一个长事务分割成一系列小事务的动机是什么？其结果会引发什么问题？这些问题如何避免?

24.11　假设内存价格下降一半，磁盘访问速度(每秒的访问次数)加倍，其他因素保持不变。此改变对于 5 分钟规则和 1 分钟规则将会有什么影响?

24.12　列出 TPC 基准程序的至少 4 个有助于得到实际的和可信赖的评测结果的特性。

24.13　TPC-D 基准程序为什么会被 TPC-H 和 TPC-R 基准程序所替代?

24.14　解释应用程序的哪些特性有助于你决定最好选择 TPC-C、TPC-H 和 TPC-R 中的哪个来对应用程序进行建模。

文献注解

Kleinrock[1975]是关于排队论的经典教科书。

数据库系统基准程序的一个早期建议(Wisconsin 基准程序)由 Bitton 等[1983]提出。TPC-A、TPC-B 和 TPC-C 基准程序在 Gray[1991]里介绍。所有 TPC 基准程序描述以及基准程序结果的联机版本可以在 URL 为 1059
www. tpc. org 的万维网上获得；该站点还包含有关新基准程序建议的最新信息。OODB 的 OO1 基准程序在 Cattell 和 Skeen[1992]中描述；OO7 基准程序在 Carey 等[1993]中描述。

Shasha 和 Bonnet[2002]提供了数据库调整方面的详细报告。O'Neil 和 O'Neil[2000]是一本很好地描述了性能度量和调整的教科书。Gray 和 Graefe[1997]描述了 5 分钟规则和 1 分钟规则，Graefe[2008]最近扩充到考虑了主存、闪存和磁盘的组合。

Ross 等[1996]、Chaudhuri 和 Narasayya[1997]、Agrawal 等[2000]和 Mistry 等[2001]中讨论了索引选择和物化视图选择。Zilio 等[2004]、Dageville 等[2004]和 Agrawal 等[2004]中描述了 IBM DB2、Oracle 和 Microsoft SQL Server 支持的调整功能。

有关 ODBC、OLE-DB、ADO 和 ADO. NET 的信息可以在 Web 站点 www. microsoft. com/data 中找到，并且有关该主题的许多书籍可以通过 www. amazon. com 找到。每季发行的 *ACM Sigmod Record* 中有一个常规章节用于讲述数据库中的标准。

基于 XML 的标准和工具的大量信息可以在 Web 站点 www. w3c. org 在线找到。有关 RosettaNet 的信息可以在 Web 站点 www. rosettanet. org 找到。

Cook[1996]讲述了商务处理的工程再设计。Umar[1997]讲述了工程再设计和遗产系统处理中的问题。 1060

性通信。一致性检查可以延迟到需要数据时再进行，尽管这种延迟可能增加数据库整体的不一致性。

断开连接的可能性和无线通信的开销限制了第 19 章所讨论的分布式系统的事务处理技术的实用性。通常最好是由用户在移动主机上准备事务，但需要他们将事务提交给服务器执行，而不是在本地执行这些事务。跨越多台计算机并且其中包含移动主机的事务在事务提交过程中会面临长时间的阻塞，除非断开连接的情况极少发生或者是可以预测的。

25.6 总结

- 时间在数据库系统中扮演着重要的角色。数据库是现实世界的模型。尽管大部分数据库只模拟现实世界在一个时间点(在当前时间)上的状态，但时态数据库模拟的是现实世界随时间变化的状态。
- 时态关系中的事实与当它们有效时的时间相关联，而时间可以用时段的并来表示。时态查询语言简化了时间建模以及与时间相关的查询。
- 目前空间数据库正越来越多地用于存储计算机辅助设计数据和地理数据。
- 设计数据主要以矢量数据的形式存储；地理数据包含矢量数据和光栅数据的组合。空间完整性约束对于设计数据十分重要。
- 矢量数据可以编码成为第一范式数据，或者用非第一范式结构来存储，如列表。专用索引结构对于访问空间数据和处理空间查询尤为重要。 1085
- R 树是 B 树的多维扩展；它和它的变体(如 R^+ 树和 R^* 树)在空间数据库中得到了广泛的应用。将空间以某种固定方式进行划分的索引结构(如四叉树)有助于处理空间连接查询。
- 多媒体数据库正变得越来越重要。诸如基于相似性的检索以及按可以确保的速率传输数据那样的问题是当前研究的重要课题。
- 移动计算系统的普及使得人们对在这类系统上运行的数据库系统产生了兴趣。在这类系统中的查询处理可能会涉及在服务器端数据库上的查找。查询代价模型必须包括通信代价，其中包括金钱上的成本和电池电源的成本，它们对于移动系统来说是相对较高的。
- 广播方式比点对点通信在每接收者上的成本要便宜得多，诸如股市之类数据的广播有助于移动系统低成本地接收数据。
- 连接断开操作、广播数据的使用和数据缓存是移动计算中正致力解决的三个重要问题。

术语回顾

- 时态数据
- 有效时间
- 事务时间
- 时态关系
- 双时态关系
- 全球协调时间(UTC)
- 快照关系
- 时态查询语言
- 时态选择
- 时态投影
- 时态连接
- 空间和地理数据
- 计算机辅助设计(CAD)数据
- 地理数据
- 地理信息系统
- 三角剖分
- 设计数据库
- 地理数据
- 光栅数据
- 矢量数据
- 全球定位系统(GPS)
- 空间查询
- 临近查询
- 最近邻居查询
- 区域查询
- 空间连接
- 空间数据索引
- k-d 树
- k-d-B 树
- 四叉树
  - □ PR 四叉树
  - □ 区域四叉树
- R 树
  - □ 边界框
  - □ 二次方分裂
- 多媒体数据库
- 等时数据
- 连续媒体数据
- 基于相似性的检索
- 多媒体数据格式
- 视频服务器
- 移动计算
  - □ 移动主机
  - □ 移动支持站点
  - □ 单元
  - □ 交接
- 位置相关查询
- 广播数据
- 一致性
  - □ 失效报告
  - □ 版本向量方案

实践习题

25.1 时间有哪两种类型？怎样区分它们？为什么将一个元组与这两类时间联系起来是有意义的？

25.2 假设有一个关系包含 x，y 坐标和餐馆名。并假设查询只能是如下形式：查询指定一个点，问是否恰好有一家餐馆在这个点上。R 树或 B 树哪一种索引更好？为什么？

25.3 假设有一个空间数据库支持区域查询（圆形区域），但不支持最近邻居查询。描述一个算法，通过利用多个区域查询来找到最近的邻居。

25.4 假设要在 R 树中存储线段。如果线段与坐标轴不平行，它的边界框会很大，并包含大量的空白区域。

- 请说明在查询与一个给定区域相交的线段时，大边界框对性能的影响。
- 简要描述一种能提高此类查询性能的技术，并给出体现其优点的例子。提示：可以把线段分为更小的部分。

25.5 请给出一个使用 R 树索引来有效计算两个关系的空间连接的递归过程。（提示：使用边界框检查一对内部结点之下的叶结点项是否相交。）

25.6 描述 RAID 磁盘组织（10.3 节）中的想法是如何应用于数据广播环境的，该环境中有时会有噪音阻止部分传输数据的接收。

25.7 定义一个重复广播数据的模型，其中广播介质建模为一个虚拟磁盘。描述这个虚拟磁盘的存取时间和数据传输率与一个普通硬盘的对应值的差异。 1086 | 1087

25.8 考虑一个将所有文档保存在中央数据库的文档数据库。有些文档的副本存储在移动计算机上。假设移动计算机 A 在断开连接期间更新了文档 1 的副本，同时，移动计算机 B 在断开连接期间更新了文档 2 的副本。请说明在移动计算机重新连接时，版本向量方案如何保证对中央数据库和移动计算机的正确更新。

习题

25.9 如果通过增加一个时间属性来将关系转换为时态关系，函数依赖能够保持吗？这一问题在时态数据库中是怎样处理的？

25.10 考虑二维矢量数据，其中的数据项互不重叠。是否可以把这些矢量数据转换为光栅数据？如果可以，存储经过这种变换得到的光栅数据而非原始矢量数据有什么缺点？

25.11 研究你使用的数据库系统所提供的空间数据支持，实现以下内容：

a. 一个模式，描述饭店的地理位置和其他特征，如饭店提供的烹调风格和消费级别。

b. 一个查询，寻找中等价位的提供印度食物，并且距离你的房子（为你的房子假设一个任意的位置）5 英里以内的饭店。

c. 一个查询，为每个饭店寻找其与最近的一个具有相同的烹调风格和消费级别的饭店之间的距离。

25.12 在连续媒体系统中，如果数据传送过慢或过快会导致什么问题？

25.13 列出无线网络上的移动计算与传统分布式系统相区别的三个主要特性。

25.14 列出传统查询优化器没有考虑，但在移动计算的查询优化中需要考虑的三个因素。

25.15 给出版本向量方案无法保证可串行性的例子。（提示：使用实践习题 25.8 中的例子，假设文档 1 和文档 2 在移动计算机 A 和 B 上都可用，并考虑文档读出时没有被更新的可能性。） 1088

文献注解

Stam 和 Snodgrass[1988] 和 Soo[1991] 提供了关于时态数据管理的概览。Jensen 等[1994] 给出了时态数据库概念的一个术语表，旨在统一术语。Tansel 等[1993] 是有关时态数据库不同方面文章的一个汇集。Chomicki[1995] 给出了管理时态完整性约束的技术。

Heywood 等[2002] 是讲述地理信息系统的教科书。Samet[1995b] 是关于空间索引结构上大量工作的一个综述。Samet[1990] 和 Samet[2006] 是讲述空间数据结构的教科书。Finkel 和 Bentley [1974] 中有对四叉树的早期描述。Samet[1990] 和 Samet[1995b] 描述了四叉树的各种变体。Bentley[1975] 描述了 k-d 树，Robinson [1981] 描述了 k-d-B 树。Guttman[1984] 最先提出了 R 树。R 树的各种扩展由 Sellis 等[1987] 作了阐述，其

中描述了 R^+ 树。Beckmann 等[1990]描述了 R^* 树。

Brinkhoff 等[1993]讨论了使用 R 树来实现空间连接。Lo 和 Ravishankar[1996]以及 Patel 和 DeWitt[1996]介绍了计算空间连接的基于划分的方法。Samet 和 Aref[1995]综述了空间数据模型、空间运算以及空间与非空间数据的集成。

Revesz[2002]是讲述约束数据库方面领域的教科书；时间间隔和空间区域可以看作是约束的特殊情况。

Samet[1995a]阐述了多媒体数据库中的研究问题。Faloutsos 和 Lin[1995]讨论了多媒体数据的索引。

Dashti 等[2003]是讲述流媒体服务器设计的教科书，包括对磁盘子系统上数据组织的详细介绍。Anderson 等[1992]、Rangan 等[1992]、Ozden 等[1994]、Freedman 和 DeWitt[1995]以及 Ozden 等[1996b]讨论了视频服务器。Berson 等[1995]和 Ozden 等[1996a]讨论了容错性。

1089
↓
1090

Alonso 和 Korth[1993]以及 Imielinski 和 Badrinath[1994]研究了包括移动计算机的系统中的信息管理。Imielinski 和 Korth[1996]介绍了移动计算及关于该主题的一系列研究论文。

Popek 等[1981]以及 Parker 等[1983]阐述了用来检测分布式文件系统中的不一致性的版本向量方案。

10.7 总结

- 现代数据管理应用通常需要处理的数据不一定是关系形式的，而且这些应用还需要处理大量的数据，数据量远远超过单个传统组织机构所能够生成的数据量。
- 越来越多的数据传感器的使用导致传感器和嵌入在别的对象中的其他计算设备连接到互联网，这通常被称为“物联网”。
- 现在大数据应用的查询语言的选择种类繁多，是因为这些应用需要处理更为多样的数据类型，以及需要扩展到非常大的数据量 / 高速度。
- 构建能够扩展到大容量 / 高速度数据的数据管理系统需要并行地存储和处理数据。
- 分布式文件系统允许文件跨多台机器存储，同时允许使用传统文件系统接口访问文件。
- 键值存储系统允许基于键来存储和检索记录，另外还可以提供有限的查询功能。这些系统不是成熟的数据库系统，它们有时被称为 NoSQL 系统。
- 并行和分布式数据库提供传统的数据库接口，但它们跨多台机器存储数据，并跨多台机器并行执行查询处理。 511
- MapReduce 范式对并行处理中的一种常见情况进行建模，其中一些由 map() 函数标识的处理被应用于大量输入记录中的每条记录，然后一些由 reduce() 函数标识的聚集形式被应用于 map() 函数的结果。
- Hadoop 系统提供了采用 Java 语言的、广泛使用的 MapReduce 开源实现。
- 有大量应用使用 MapReduce 范式进行各种类型的数据处理，其逻辑可以很容易地用 SQL 表示。
- 关系代数构成了关系查询处理的基础，允许查询被建模为运算树。通过支持能够处理包含复杂数据类型的记录的数据集的代数运算符，并返回包含类似复杂数据类型的记录的数据集，此思想被扩展到具有更复杂数据类型的环境中。
- 有许多应用需要在连续到达的数据上连续执行查询。术语**流数据**是指以连续方式到达的数据。许多应用领域需要实时处理输入的数据。
- 图是数据库需要处理的重要数据类型。

术语回顾

- 量
- 速度
- 转换
- 物联网
- 分布式文件系统
- 名字节点服务器
- 数据节点机
- 分片
- 划分属性
- 键值存储系统
- 键值存储
- 文档存储
- NoSQL 系统
- 分片码
- 并行数据库
- reduce 键
- 洗牌阶段
- 流数据
- 静态数据
- 流上的窗口
- 连续查询
- 标点
- lambda 架构
- 滚动窗口

- 跳跃窗口
- 滑动窗口
- 会话窗口
- 发布 - 订阅系统
- pub-sub 系统
- 离散化流
- 超步

512

实践习题

10.1 假设需要存储非常大量的小文件，每个文件的大小为 2 KB。如果需要在分布式文件系统和分布式键值存储之间做出选择，你会选择哪一种？请解释原因。

10.2 假设需要在一个分布式文档存储（如 MongoDB）中存放大量学生的数据。另外，假设每名学生的数据对应于 *student* 和 *takes* 关系中的数据。你将如何表示有关学生的上述数据，以确保能够高效地访问特定学生的所有数据？请给出一名学生的数据表示作为示例。

10.3 假设希望为大量用户存储水电费账单，其中每份账单都由客户 ID 和日期来标识。如果查询请求特定客户在特定日期范围内的账单，该如何把账单存储在支持范围查询的键值存储中？

10.4 使用单个 MapReduce 步骤，给出用于计算连接 $r \bowtie_{r.A=s.A} s$ 的伪码，假设对 r 和 s 的每个元组调用 map() 函数。还假设 map() 函数可以使用 context.relname() 来找到关系的名称。

10.5 以下用于处理非常大的数据的 Apache Spark 代码片段的概念性问题是什么？注意，collect() 函数返回一个 Java 集合，Java 集合（从 Java 8 开始）支持 map 和 reduce 函数。

```
JavaRDD<String< lines = sc.textFile("logDirectory");
int totalLength = lines.collect().map(s -> s.length())
                  .reduce(0,(a,b) -> a+b);
```

10.6 Apache Spark：

a. Apache Spark 如何并行地执行计算？

b. 解释“Apache Spark 以一种延迟的方式在 RDD 上执行转换”。

c. 对 Apache Spark 中的运算进行延迟计算有哪些好处？

10.7 给定文档集合，对于每个词 w_i，令 n_i 表示该词在集合中出现的次数。N 是所有文档中出现的词汇总数。接下来，考虑文档中所有连续词汇对（w_i, w_j），用 $n_{i,j}$ 表示所有文档中词汇对（w_i, w_j）出现的次数。

513

编写一段 Apache Spark 程序，给定一个目录中的文档集合，计算 N、所有（w_i, n_i）对和所有（(w_i, w_j), $n_{i,j}$）对。然后输出满足 $n_{i,j} / N \geq 10 * (n_i / N) * (n_j / N)$ 的所有词汇对。如果两个词汇彼此独立出现，则这些词汇对出现的频率将达到预期的 10 倍或 10 倍以上。

你将在 RDD 上找到对最后一步有用的连接运算，以将相关计数汇总在一起。为简单起见，可不考虑跨行的词汇对。同样为了简单起见，还假设词汇都是小写的并且没有标点符号。

10.8 使用滚动窗口运算符考虑以下查询：

```
select item, System.Timestamp as window_end, sum(amount)
from order timestamp by datetime
group by itemid, tumblingwindow(hour, 1)
```

使用普通的 SQL 构造而不使用滚动窗口运算符来给出等价的查询。假定可以使用 to_seconds(timestamp) 函数把时间戳转换为一个整数值，它表示自（比如说）1970 年 1 月 1 日午夜以来经过的秒数。你还可以假设通常的算术函数是可用的，以及返回 $\leq a$ 的最大整数的函数 floor(a) 也可用。

10.9 假设希望将大学模式建模为一个图。对于以下每个关系，请解释关系是否可以作为节点或作为边来建模：

(i) *student*，(ii) *instructor*，(iii) *course*，(iv) *section*，(v) *takes*，(vi) *teaches*。

该模型是否表示了各授课信息和课程之间的联系？

习题

10.10 给出用户访问的网页的相关网络日志信息可供网站使用的四种方式。

10.11 大数据的特点之一是数据的多样性。解释为什么这种特性导致需要 SQL 之外的语言来处理大数据。

10.12 假设你的公司已经构建了一个运行在集中式数据库上的数据库应用，但即使使用高端计算机和在数据上创建的适当索引，系统也无法处理事务负载，从而导致查询处理速度变慢。你有哪些选择来允许应用处理事务负载？ 514

10.13 map-reduce 框架对于在文档集合上创建倒排索引非常有用。倒排索引为每个词存储一个包含它的所有文档 ID 的列表（通常也会存储文档中的偏移量，但在这个问题中我们忽略它们）。

例如，如果输入文档的 ID 和内容如下：

1: data clean

2: data base

3: clean base

那么倒排列表就是

data: 1, 2

clean: 1, 3

base: 2, 3

给出在给定文件集（每个文件都是一份文档）上创建倒排索引的 map 和 reduce 函数的伪码。假设可以使用 context.getDocumentID() 函数来获取文档 ID，并且在文档的每行调用一次 map 函数。为每个词输出的倒排列表应该是用逗号分隔的文档 ID 的列表。文档 ID 通常是有序的，但在这个问题中你不需要费心地对它们进行排序。

10.14 填写以下空白，以完成 Apache Spark 程序，该程序计算文件中每个词汇的出现次数。为了简单起见，我们假设词汇只以小写形式出现，并且没有标点符号。

```
JavaRDD<String> textFile = sc.textFile("hdfs://...");
JavaPairRDD<String, Integer> counts =
    textFile._____(s -> Arrays.asList(s.split(" "))._____())
     .mapToPair(word -> new _____).reduceByKey((a, b) -> a + b);
```

10.15 假设流不能按照元组时间戳的顺序传递元组。流应该提供哪些额外信息，以便流式查询处理系统可以决定何时看到一个窗口中的所有元组？

10.16 请解释如何使用发布 - 订阅系统（如 Apache Kafka）在流上执行多个运算。

工具

除了一些商业工具外，还有各种各样的开源大数据工具可用。此外，云平台上还提供了许多这种工具。我们在下面列出了几种流行的工具，以及可以找到它们的 URL。Apache HDFS（hadoop.apache.org）是一种广泛使用的分布式文件系统实现。开源的分布式 / 并行键值存储包括 Apache HBase（hbase.apache.org）、Apache Cassandra（cassandra.apache.org）、MongoDB（www.mongodb.com）以及 Riak（basho.com）。 515

托管的云存储系统包括 Amazon S3 存储系统（aws.amazon.com/s3）和 Google Cloud Storage（cloud.google.com/storage）。托管的键值存储包括 Google BigTable（cloud.google.com/bigtable）和 Amazon DynamoDB（aws.amazon.com/dynamodb）。

Google Spanner（cloud.google.com/spanner）和开源的 CockroachDB（www.cockroachlabs.com）是可扩展的、支持 SQL 和事务的并行数据库，并具有强一致性的存储。

开源的 MapReduce 系统包括 Apache Hadoop（hadoop.apache.org）和 Apache Spark（spark.apache.org），而 Apache Tez（tez.apache.org）支持使用代数运算符的 DAG 进行数据处理。还有来自 Amazon

Elastic MapReduce（aws.amazon.com/emr）（还支持 Apache HDF 和 Apache HBase）和 Microsoft Azure（azure.microsoft.com）的基于云的产品是可用的。

Apache Hive（hive.apache.org）是一种流行的开源 SQL 实现，它运行在 Apache MapReduce、Apache Tez 和 Apache Spark 之上。这些系统被设计用于支持在多台机器上并行运行的大型查询。Apache Impala（impala.apache.org）是一个运行在 Hadoop 上的 SQL 实现，被设计用于处理大量查询，并以最小的延迟返回查询结果。支持并行处理的云上托管的 SQL 产品包括 Amazon EMR（aws.amazon.com/emr）、Google Cloud SQL（cloud.google.com/sql）和 Microsoft Azure SQL（azure.microsoft.com）。

Apache Kafka（kafka.apache.org）和 Apache Flink（flink.apache.org）是开源的流处理系统，Apache Spark 也提供对流式处理的支持。托管的流处理平台包括 Amazon Kinesis（aws.amazon.com/kinesis）、Google Cloud Dataflow（cloud.google.com/dataflow）和 Microsoft Stream Analytics（azure.microsoft.com）。开源的图处理平台包括 Neo4J（neo4j.com）和 Apache Giraph（giraph.apache.org）。

延伸阅读

[Davoudian et al. (2018)] 提供了一个很好的 NoSQL 数据存储调查，包括数据模型查询和内部结构。关于 Apache Hadoop 的更多信息（包括有关 HDFS 和 Hadoop MapReduce 的文档）可以在 Apache Hadoop 的主页 hadoop.apache.org 上找到。关于 Apache Spark 的信息可以在 Spark 的主页 spark.apache.org 上找到。关于 Apache Kafka 流数据平台的信息可以在 kafka.apache.org 上找到，关于
516 Apache Flink 的流处理的详细信息可以在 flink.apache.org 上找到。批量同步处理在 [Valiant (1990)] 中介绍。Pregel 系统的描述（包括其对批量同步处理的支持）可在 [Malewicz et al. (2010)] 中找到，关于其等价的开源软件 Apache Giraph 的信息可以在 giraph.apache.org 上找到。

参考文献

[Davoudian et al. (2018)] A. Davoudian, L. Chen, and M. Liu, "A Survey of NoSQL Stores", *ACM Computing Surveys*, Volume 51, Number 2 (2018), pages 2-42.

[Malewicz et al. (2010)] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing", In *Proc. of the ACM SIGMOD Conf. on Management of Data* (2010), pages 135-146.

517 ~ 518

[Valiant (1990)] L. G. Valiant, "A Bridging Model for Parallel Computation", *Communications of the ACM*, Volume 33, Number 8 (1990), pages 103-111.