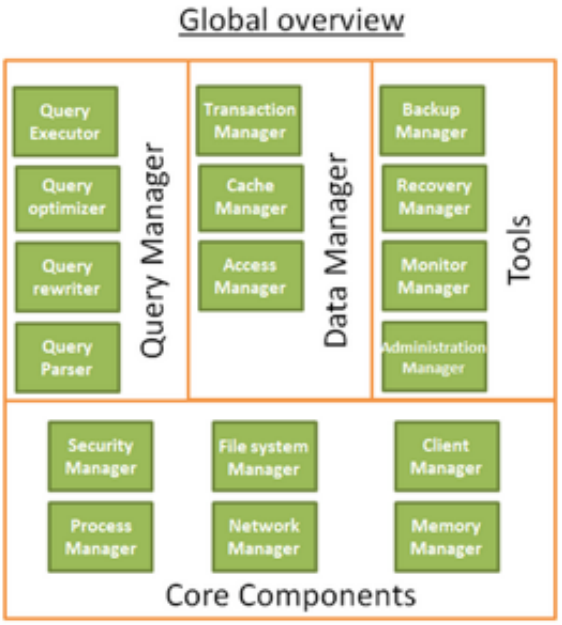
极客头条 (/) (/search)

关系型数据库是如何运作的（下）

2

锁 (http://www.csdn.net/tag/锁/news) 数据库 (http://www.csdn.net/tag/数据库/news) 缓存 (http://www.csdn.net/tag/缓存/news) 关系型数据库 (http://www.csdn.net/tag/关系型数据库/news)

此前，CSDN发布了《关系型数据库是如何运作的（上）(http://geek.csdn.net/news/detail/52539)》深受开发者朋友们的欢迎，本文是完结篇，内容涉及数据库总体架构，客户端管理器，查询管理器，数据管理器介绍等。



总体架构

前述文章从比较细的角度来讨论了数据库，现在我们尝试从宏观角度来分析。

数据库的核心组件：

- 过程管理器(The process manager)：数据库都会有一个过程池/线程池需要进行管理。此外，为了使运行时间更短，现代数据库会使用自己的线程来替代操作系统线程。
- 网络管理器(The network manager)：网络的输入输出是个大问题，特别是对于分布式数据库来说。所以部分数据库针对网络管理打造了自己的管理器。

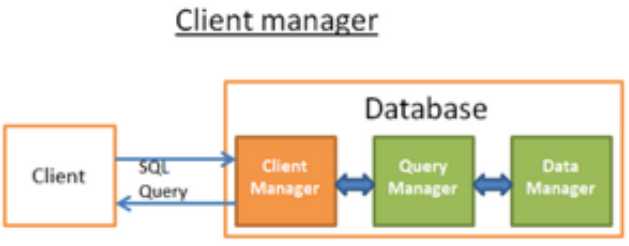
- 文件系统管理器(File system manager)：磁碟I/O是数据库的第一瓶颈。使用管理器进行磁碟文件进行管理是很重要的。
- 内存管理器(Memory manager)：当你需要处理大量内存数据或大量查询，一个高效的内存管理器是必须的。
- 安全管理器(Security manager)：进行认证和用户认证管理。
- 客户端管理器(Client manager)：进行客户端连接管理

数据库的工具：

备份管理器：进行数据库的备份与恢复
复原管理器：在数据库崩溃后进行数据库重启
监视管理器：进行数据库活动日志记录，同时进行数据库监视
管理员管理器：进行metadata存储，管理数据库，表空间，数据泵等
查询管理器：对查询进行有效性检验，优化，编译和执行
数据管理器：包括事务管理器，缓存管理器，数据访问管理器

下面将详细介绍客户端管理器，查询管理器以及数据管理器。

客户端管理器



客户端管理器用于处理和管理客户端的通信。客户端可以是一台服务器或是终端应用。客户端管理器透过不同的API来提供访问权，例如：JDBC，ODBC，OLE-DB等。

当你连接到一个数据库时：

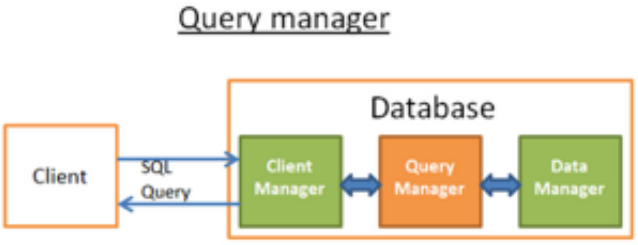
- 管理器会对你的身份和授权进行确认。
- 如果验证通过，会对你的查询请求进行处理。
- 管理器同时会检查数据库是否处于满负荷状态。
- 管理器会等待请求资源的返回。如果发生超时，它会关闭连接并返回可读的错误信息。
- 然后会把你的查询发送给查询管理器，而你的查询是被处理状态。

- 管理器会存储部分结果到缓冲区然后开始进行结果返回。
- 如果出现异常，管理器会中断连接，返回相关原因解释并释放资源。

查询管理器

查询管理器是数据库的重要组成部分。其工作过程是：

- 查询会被解释以确认有效性
- 然后会被重写以消除不必要的操作并进行预优化处理
- 然后会被优化处理以提高性能并发送到执行和数据访问计划
- 然后改计划会被编译处理
- 最后进行执行查询



查询重写器的运作

重写器的目的是：

1. 进行查询预优化处理
2. 避免不必要的操作
3. 帮助优化器找出最佳方案

常见的重写规则：

视图合并：如果你在查询中使用了视图，那么该视图会被转换层SQL视图代码

子查询扁平化：子查询使查询优化变得困难，因此重写器会修改含有子查询的查询以消除子查询。

例如：

```
SELECT PERSON.*
FROM PERSON
WHERE PERSON.person_key IN
(SELECT MAILS.person_key
FROM MAILS
WHERE MAILS.mail LIKE 'christophe%');
```

会被重写为：

```
SELECT PERSON.*
FROM PERSON, MAILS
WHERE PERSON.person_key = MAILS.person_key
and MAILS.mail LIKE 'christophe%';
```

消除不必要的操作符：例如当你使用了UNIQUE唯一约束而同时使用了DISTINCT操作符，那么DISTINCT将会被消除。

多于JOIN连接清除：当你 有两次相同条件的JOIN连接但是其中一个条件被隐藏了或者是一个多于的JOIN，那么它会被清除。

分区处理：如果你使用了一个分区表，那么重写器会找出那个分区会被使用。

自定义规则：如果你有自定义的查询规则，重写器会执行这些规则。

数据管理器

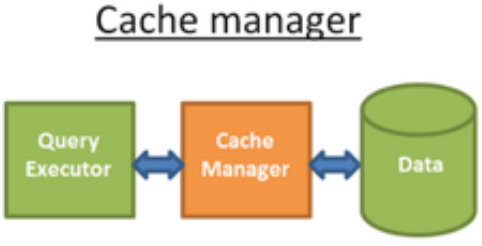
查询管理器的作用是执行查询并对资源发出请求，数据管理器会处理这些请求并返回结果。但这里有两个问题：

- 关系数据库使用的是事务模型。所以你有可能得不到数据，因为其他人可能会正同时使用/修改这些数据。
- 数据获取是数据库中最慢的操作，因此数据管理必须要能高效地获取并数据存放在内存缓冲区。

那么关系数据库是如何解决这两个问题的呢？

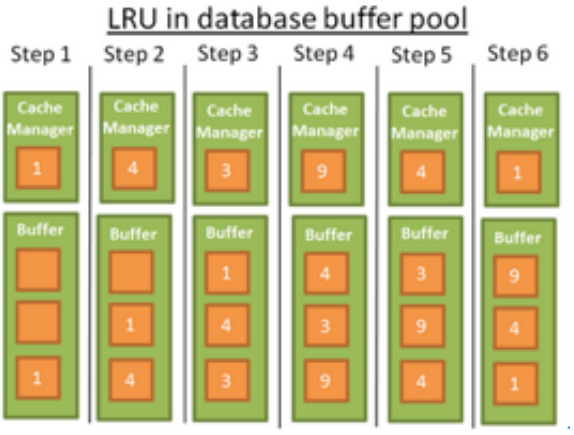
缓存管理器

如前所述，数据库的主要瓶颈是磁碟I/O。所以现代数据库使用了缓存管理器来提高效率。查询执行器的数据请求对象是缓存管理器而不是直接的文件系统。缓存管理器有一个内存里缓存叫做缓冲池。从内存获取数据会大大提高数据库速度。



缓冲–替换策略

很多主流数据库(如：SQL Server，MySQL，Oracle等)使用的是LRU算法。
LRU是Least Recently Used的简写，意思最近使用。其理念是缓存最近使用的数据以便再次使用时快速读取。



虽然它有很多优点但也存在不足，比方说表/索引的大小超过了缓冲区大小。因此出现了进阶版本的LRU，这就是LRU-K，例如在SQL Server 使用的是LRU-K，K=2。在LRU-K中：

- 首先考虑数据的K次最近使用记录；根据数据的使用次数分配权值；如果有新的数组载入缓存，旧的但经常使用的数据不会被移除，但是当旧数据不再使用，将会被移除，所以权值的设立有助于减少多余数据。

事务管理器

事务管理器是为了确保每个查询会执行自己的事务。在讲述事务管理期前，我们需要理解ACID事务的概念。

ACID是一个工作单元，它的意思是：

Atomicity(原子性)：事务是”全或全不”的，即使是10个小时的事务。如果事务崩溃了，会发生状态回滚。

Isolation(隔离性)：如果事务A和B同时运行，那么事务A和B的结果必须是一致的，不论A对于B是完成前/完成后/过程中的状态。

Durability(耐久性)：一旦事务完成，数据会存放在数据库中而不论发生什么情况(异常或错误)。

Consistency(一致性)：只有有效数据被写入数据库。一致性与原子行和隔离性关联。

并发控制

确保隔离性，附着性和原子性的关键是能对同一数据进行正确写操作(添加，更新和删除)：

如果仅仅是数据读取事务，那么它们可以不与其它修改事务发生冲突；
如果一个修改事务处理的数据被其它事务读取，数据库需要找到方法来隐藏这些修改操作。同时，它需要保证这些修改操作不会被清除。

以上问题就是并发控制。最简单的处理方法是逐个执行事务。但是这不利于进行规模扩张，也无法发挥服务器/CPU的多核性能。理想的处理方式是每当事务新建或取消时：

监视所有事务的全部操作，检查同时读取/修改相同数据的两个(或多个)事务是否发生冲突，在发生冲突的事务中进行操作记录以减少冲突部分的大小，把冲突部分以其它次序进行处理，判别某事务是否可以取消

更正规的做法是进行冲突日程表管理。但是在企业级数据库中，是很难为每个新事务事件分配足够多的处理时间。所以会使用其它方法来进行处理。

锁管理器

为了处理以上问题，多数数据库会采用锁或数据版本来进行处理。但这是个内容丰富的话题，以下会把讨论重点放在锁部分。

什么是锁呢？

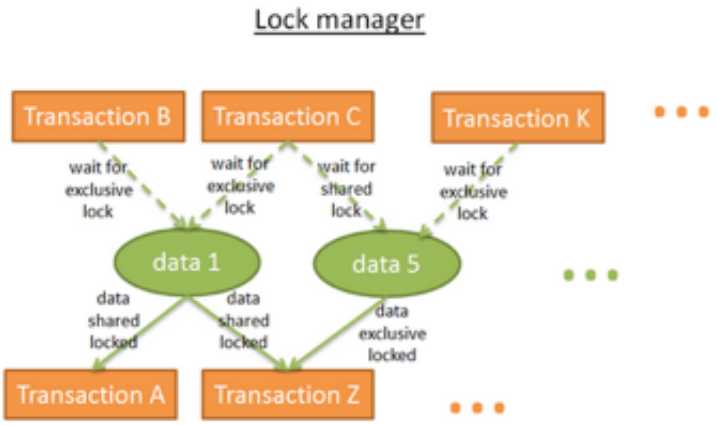
- 事务是否需要数据
- 是否锁定了数据
- 另一事务是否需要相同数据
- 是否不得不等待直至第一个事务释放这些数据

这叫做排斥锁。但是排斥锁针对的对象相同数据的读取和等待，这是不利于资源调配的。还有一种锁，叫共享锁。

在共享锁中：

- 一个事务是否只需读取数据A
- 共享锁对数据锁定并读取数据
- 如果第二个事务也只需要读取数据A
- 共享锁对数据锁定并读取数据
- 如果第三个事务只需要修改数据A
- 那么会对数据进行排斥锁锁定，但它必须等待直至事务一，二释放共享锁才对数据A进

行排斥锁锁定

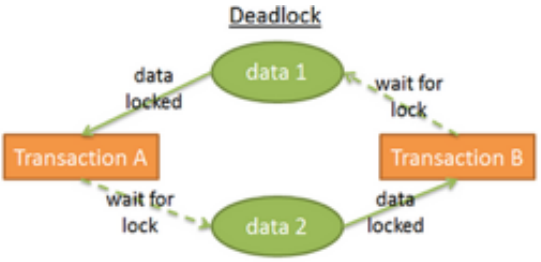


锁管理器的作用是提供和释放锁。从内部角度看，它把锁存储在一个有关联的hash数据表中。

- 哪些事务锁定了数据
- 哪些事务在等待数据

死锁

锁的存在会导致一个问题：两个事务在无限期地等待数据：



在上图中：

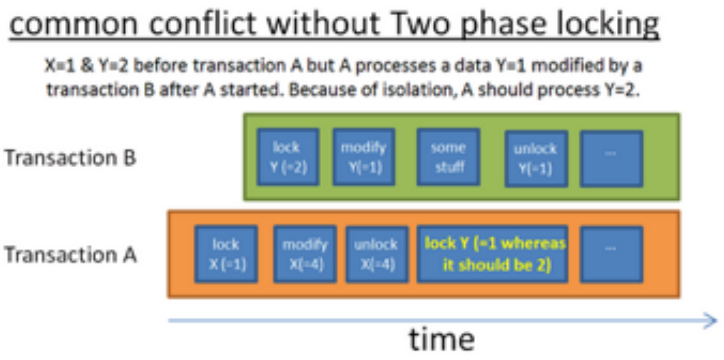
- 事务A对数据1使用了排斥锁，同时在等待获取数据2
- 事务B对数据2使用了排斥是，同时在等待获取数据1

这就是死锁。

遇到死锁后，锁管理器会选择对哪个事务进行撤销(回滚)以消除死锁。但要进行选择，并不是件容易的事。DB2和SQL Sever使用了两段锁协议(Two-Phase Locking Protocol)来进行处理。

- 在增长段，事务会得到锁，但是不能释放锁。

- 在下降段，事务可释放锁，但是不能得到锁。



其核心理念是：

- 释放不再使用的锁以减少其他事务对这些锁的等待时间
- 避免事务开始后对数据进行修改，所以这是非连贯事务

写在最后

我一直所坚持的习惯是：明白你所使用的技术，如果你想不断提升自己的开发水平，尝试深入掌握你所使用的工具的原理是个大有裨益的方法。虽然NoSQL在现今很流行，但是它们还是属于发展初期，一些特定的问题或重要思想还是得借助关系数据库才能彻底弄懂。

英文链接：http://coding-geek.com/how-databases-work/ (http://coding-geek.com/how-databases-work/)

（译者/伍昆 责编/夏梦竹）

(http://geek.csdn.net/user/publishlist/rnifeasy)

rnifeasy (http://geek.csdn.net/user/publishlist/rnifeasy)

发布于 GEEKNEWS (http://geek.csdn.net/forum/1) 2016-02-09 20:02
