第 06 讲:缓冲池

15-445/645 数据库系统(秋季 2022)

https://15445.courses.cs.cmu.edu/fall2022/

卡内基梅隆大学安迪·帕夫洛

1 介绍

DBMS 负责管理其内存和从磁盘来回移动数据。由于在大多数情况下，数据不能直接在磁盘上操作，因此任何数据库都必须能够有效地将磁盘上以文件形式表示的数据移动到内存中，以便使用。这种交互作用的示意图如图 1 所示。DBMS 面临的一个障碍是最小化数据移动速度的问题。理想情况下，它应该“看起来”好像数据已经全部在内存中了。执行引擎不应该担心如何将数据提取到内存中。

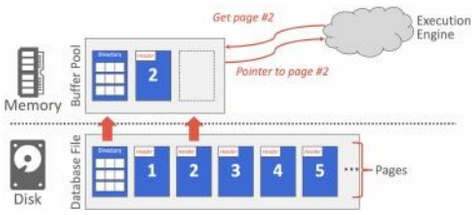

图 1:面向磁盘的 DBMS。

考虑这个问题的另一种方式是从空间和时间控制的角度。

*空间控制*指的是页面在磁盘上的物理写入位置。空间控制的目标是使经常一起使用的页面在磁盘上的物理位置尽可能靠近。

*时间控制*是指何时将页面读入内存，何时将页面写入磁盘。临时控制的目的是尽量减少从磁盘读取数据的停顿次数。

2 锁与锁存

在讨论 DBMS 如何保护其内部元素时，我们需要区分锁和闩锁。

锁:锁是一种高级的逻辑原语，用于保护数据库(如元组、表、数据库)的内容不受其他事务的影响。事务将在其整个持续时间内持有锁。数据库系统可以向用户公开在运行查询时持有哪些锁。锁需要能够回滚更改。

锁存:锁存是 DBMS 在其内部数据结构(例如，哈希表，内存区域)中使用的低级保护原语。锁存只在操作期间保持。锁存器不需要能够回滚更改。

3 缓冲池

*缓冲池*是从磁盘读取的页面的内存缓存。它本质上是在数据库内部分配的一个大内存区域，用于存储从磁盘读取的页面。

缓冲池的内存区域被组织成固定大小的页面数组。每个数组条目称为一个*帧*。当 DBMS 请求一个页面时，一个精确的副本被放入缓冲池的一个帧中。然后，数据库

当请求页面时，系统可以首先搜索缓冲池。如果没有找到该页，则系统从磁盘中获取该页的副本。脏页被缓冲，不会立即写回。缓冲池的内存组织示意图见图 2。

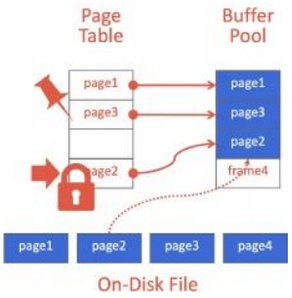

图 2:缓冲池组织和元数据

缓冲池元数据

缓冲池必须维护一定的元数据，以便有效和正确地使用。

首先，*页表*是一个内存中的哈希表，它跟踪当前在内存中的页面。它将页 id 映射到缓冲池中的帧位置。由于缓冲池中页面的顺序不一定反映磁盘上的顺序，所以这个额外的间接层允许识别池中的页面位置。

注意:不要将页表与页目录混淆，*页目录*是数据库文件中从页 id 到页位置的映射。对页目录的所有更改都必须记录在磁盘上，以便 DBMS 在重新启动时能够找到。

页表还为每个页维护额外的元数据，一个脏标志和一个引脚/引用计数器。

dirty-*flag* 是由线程在修改页面时设置的。这表明存储管理器必须将该页写回磁盘。

*引脚/引用计数器*跟踪当前正在访问该页(读取或修改该页)的线程的数量。线程必须在访问该页之前增加计数器。如果页面的计数大于零，则存储管理器不允许从内存中驱逐该页。

内存分配策略

数据库中的内存根据两个策略分配给缓冲池。

*全局策略*处理 DBMS 应该做出的决策，以使正在执行的整个工作负载受益。它考虑所有活动事务，以找到分配内存的最佳决策。

另一种选择是*本地策略*，它做出的决策将使单个查询或事务运行得更快，即使它对整个工作负载来说并不好。本地策略将帧分配给特定的事务，而不考虑并发事务的行为。

大多数系统同时使用全局视图和局部视图。

4 缓冲池优化

有许多方法可以优化缓冲池以使其适应应用程序的工作负载。

多个缓冲池

DBMS 可以为不同的目的维护多个缓冲池(即每个数据库缓冲池，每个页面类型缓冲池)。然后，每个缓冲池可以采用适合其内部存储的数据的本地策略。这种方法可以帮助

减少闩锁争用，提高局部性。

将所需页面映射到缓冲池的两种方法是对象 id 和散列。

对象 id 涉及扩展记录 id 以具有对象标识符。然后通过对象标识符，可以维护对象到特定缓冲池的映射。

另一种方法是*散列*，其中 DBMS 对页 id 进行散列以选择要访问的缓冲池。

预抓取

DBMS 还可以通过基于查询计划预取页面来进行优化。然后，在处理第一组页面的同时，可以将第二组页面预取到缓冲池中。当 DBMS 按顺序访问多个页面时，通常使用这种方法。

扫描共享(同步扫描)

查询游标可以重用从存储或运算符计算中检索到的数据。这允许多个查询附加到扫描表的单个游标上。如果一个查询开始扫描，并且已经有一个查询开始扫描，那么 DBMS 将把第二个查询的游标附加到现有的游标上。DBMS 跟踪第二个查询与第一个查询的连接位置，以便在到达数据结构的末尾时完成扫描。

缓冲池旁路

顺序扫描操作符不会将获取的页面存储在缓冲池中以避免开销。相反，内存对于正在运行的查询来说是本地的。如果操作符需要读取磁盘上连续的大量页面序列，这将很好地工作。缓冲池旁路也可以用于临时数据(排序、连接)。

5OS 页面缓存

大多数磁盘操作都通过 OS API。除非另有明确说明，否则 OS 维护自己的文件系统缓存。

大多数 DBMS 使用直接 I/O 来绕过 OS 的缓存，以避免页面的冗余副本和必须管理不同的退出策略。

Postgres 是一个使用 OS 页面缓存的数据库系统的例子。

6 缓冲区替换策略

当 DBMS 需要释放一个帧来为新页腾出空间时，它必须决定从缓冲池中删除哪个页。

替换策略是 DBMS 实现的一种算法，它在需要空间时决定从缓冲池中删除哪些页面。

替换策略的实现目标是提高正确性、准确性、速度和元数据开销。

最近最少使用(LRU)

最近最少使用替换策略维护每个页面最后访问时间的时间戳。DBMS 选择使用最旧的时间戳驱逐页面。这个时间戳可以存储在一个单独的数据结构中，比如一个队列，允许排序，并通过减少排序时间来提高效率。

CLOCK

CLOCK 策略是 LRU 的近似值，不需要每个页面都有单独的时间戳。在 CLOCK 策略中，每个页面都有一个引用位。当有页面被访问时，设置为 1。

为了可视化这一点，用“时钟指针”将页面组织在一个圆形缓冲区中。在扫描时检查页面的位是否设置为 1。如果是，设置为0，如果不是，则驱逐它。这样，时钟指针就会记住每次驱逐之间的位置。

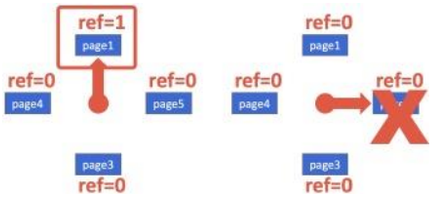

图 3:CLOCK 替换策略的可视化。引用第 1 页，设置为 1。时钟指针扫过时，将第 1 页的参考位设置为 0，并退出第 5 页。

选择

LRU 和 CLOCK 替换策略存在许多问题。

也就是说，LRU 和 CLOCK 容易受到*顺序泛洪*的影响，其中缓冲池的内容由于顺序扫描而损坏。由于顺序扫描读取每个页面，因此读取的页面的时间戳可能无法反映我们实际想要的页面。换句话说，最近使用的页面实际上是最不需要的页面。

有三种解决方案可以解决 LRU 和 CLOCK 策略的缺点。

一种解决方案是 *LRU-K*，它将最后 K 个引用的历史记录作为时间戳进行跟踪，并计算后续访问之间的间隔。这个历史记录被用来预测页面下一次被访问的时间。

另一个优化是每个查询的*本地化*。DBMS 在每个事务/查询的基础上选择要退出的页面。这最大限度地减少了每次查询对缓冲池的污染。

最后，*优先级提示*允许事务在查询执行期间根据每个页面的上下文告诉缓冲池页面是否重要。

脏页

有两种方法可以处理带有脏位的页面。最快的方法是删除缓冲池中任何非脏的页面。一种较慢的方法是将脏页写回磁盘，以确保其更改被持久化。

这两种方法说明了快速驱逐与将来不会再次读取的脏写页面之间的权衡。

避免不必要地写出页面问题的一种方法是*后台写*。通过后台写入，DBMS 可以定期遍历页表并将脏页写入磁盘。当一个脏页被安全写入时，DBMS 可以驱逐该页，或者只是取消脏标志。

7 其他内存池

除了元组和索引之外，DBMS 还需要内存。这些其他内存池可能并不总是由磁盘支持，这取决于实现。

- 排序+连接缓冲区
- 查询缓存
- 维护缓存
- 日志缓冲区
- 字典缓存