 DOI: https://doi.org/10.12345/bdai.v6i3.28262

The research of multidemension and bigdata Table association database deadlock mechanism

Yusheng Feng

School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, Jilin, 130022, China

Abstract

Every relation Database program may met datalock problem Before correlation deadlock must know the concept of lock. In multicondition, when a resourse is locked, it always release after a stage . But deadlock happen when many process visit same database, one of every process have a lock that is another process need.So produce every process can not continue. Simple speak, process A waits process B release it's resourse. Process B also wait process A release it's resourse .so two process wait each other, produce deadlock.In database attribute is regarded resourse. Deadlock usually happen in Nested construct. To request resourse graph deadlock appears in circule of loop. To Table association of high demention, Database deadlock loop circuit is complex. Commonly, mathematical logic and discrete theory may be used。

Keywords

mathematical logic; Table association; Deadlock; Ring; nesting

多维度大数据下表关联死锁机制的研究

冯毓生

长春理工大学计算机科学技术学院，中国·吉林 长春 130022

摘　要

每个使用关系型数据库的程序都可能遇到数据死锁的情况。理解什么是死锁之前首先要了解锁定的概念。多数情况下,可以认为如果一个资源被锁定,它总会在以后某个时间被释放。而死锁发生在当多个进程访问同一个数据库时,其中每个进程拥有的锁都是其他进程所需的。由此造成每个进程都无法继续下去。简单地说,进程A等待进程B释放它的资源,B又等待A进程释放它的资源，这样就相互等待形成死锁。在DB中一般将属性设置为资源。死锁一般出现在程序的嵌套结构中。请求资源图的死锁出现在循环环路的判定中。对于表关联的高维度DB下，死锁的环路判定较为复杂,一般采用数理逻辑理论和离散数学的图论理论。

关键词

数理逻辑；表关联；死锁；环；嵌套

1 引言

产生死锁的 4 个必要条件：

①互斥条件：一个资源一次只能被一个进程使用。

②请求与保持条件：一个进程因请求资源而阻塞时 , 对已获得的资源保持不放。

③不剥夺条件：进程已获得的资源 , 在未使用完之前 , 不能强行剥夺。

④循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系。

只要想办法破坏其中的任意一个或多个条件就可以避免死锁发生。

【作者简介】冯毓生，男，中国吉林人，硕士，讲师，从事计算机科学研究。

2 基本原理

2.1 竞争资源 .

当在多个事务中锁定相同的资源时 , 锁定的顺序可能会对死锁的发生产生影响。如果两个事务尝试以不同顺序锁定同一组资源，则可能会导致一个等待另一个锁 , 从而形成一个死锁状态。为了避免这种情况，需要在所有事务中使用相同的锁定顺序。

代码示例：

BEGIN TRANSACTION

Select* FRom table- namel where id=123 FOR UPDATE;

SELECT* FRom table -nam2 where id=456 FOR UPDATE;

第二条 SQL 语句 :

BEGIN TRANSACTION

select*from table nam2 where id =456 FOR UPDATE.

select*from table-namel where id=123 FOR UPDATE

在上述代码中 , 两个事务分别对两个表中的行进行了锁定操作 , 并且在不同的顺序中尝试进行。如果这两个事务同时进行 , 则它们可能会相互等待对方释放资源。

2.2 长事务

当一个事务有锁很长时间时，它可能会阻塞其他事务 , 并导致死锁和锁表的情况。这通常是由于长时间的数据检查或处理方式导致的，无法释放锁从而导致其他事务无法访问该资源。

代码示例 :

第一条 SQL 语句 .

(1) BEGIN TRANSACTION

SELECT * FROM Table- name id=123 FOR UPDATE

模拟长时间处理

WAIT FOR DELAY‘00=00:10’;

UPDAPE table name SET Column-name=‘new-value’wlene Commit

第二条 SQL 语句：

BEGIN TRANSACTIOV

UPDATE table-name SET Column neme=new-value Commit

在上述代码中 , 第一条 SQL 语句将对单 ID 为 123 的行进行锁定，并模拟长时间处理，等待 10 秒钟后才进行对行进行更新。如果同时执行第二条 SQL 语句，则可能会导致它在等待 10 秒钟后才继续执行 , 从而阻塞其他事务的执行。

2.3 并行操作

如果在多个事务中同时执行相同的操作，如 INSERT,UPDATE 或 DELETE, 则可能会出现死锁和锁表的情况。这是因为每个事务都需要对相同的资源进行操作 , 从而导致锁定和等待锁定的情况。

第一条 SQL 语句 :

BEGIN TRANSACTION

DELETE FROM table-name whene id= 123;

Commit

第二条 SQL 语句

BEGIN TRANSACTION:

DELETE From table name where id=123; Commit

在上述代码中，两个事务都试图删除 ID=123 的行。由于 DELETE 操作是一个很耗时的操作，并且每个事务都需要对相同的资源进行删除，它们可能会相互等待对方完成，从而形成死锁。

2.4 锁定级别

数据库系统支持不同的锁定级别。如行锁定，页锁定和表锁定。如果一个事务在高级别锁定下执行 , 并且阻塞其他事务以访问相同的资源，则会导致锁定和死锁情况。

(代码示例) :

(1)SET TRNSACTI0N ISOCATIOA LEVEL SERIALIZABLE;

BEGIN TRNSACTI0N;

SELECT* From table mame whene id=123 FOR UPDATE;

Commit

(2) 第 2 条 SQL 语句

(3) BEGIN TRANSACTION;

SECT *From table-name where id=123 FoR UPDATE....Commit

在上述代码中 , 第一条 SQL 语句在事务中使用了可串行锁定级别 , 以获得最高的数据一致性。如果同时处理第二条 SQL 语句 , 则将被阻塞。直到第一个事务释放由 SELECT FOR UPPATE 保留的锁。

3 解决死锁的办法

3.1 具体方法

①优化事务的执行顺序。避免多个事务同时对同一组数据进行操作。

②减少事务的持锁时间 , 尽量缩短事务的执行时间。

③使用数据库的锁机制 , 如行级锁 , 表级锁 , 来避免死锁的发生。

④使用数据库的事务隔离级别 , 如 READ,commit, serializable 来避免死锁的发生。

3.2 SQL 查询发生死锁需要采取的措施

3.2.1 终止会话 .

可以通过终止会话来解决锁问题。首先确定哪些会话造成死锁。然后使用以下语句终止会话 :

ALTER SYSTEM KILL SESSion “sid,serial#’;

其中 sid 和 seril#. 是会话的 ID 和序列号，可以从查询结果中获取。这将强制终止指定的会话，解除死锁。

3.2.2 释放锁 .

如果确定了哪些锁造成了死锁，可以尝试手动释放这些锁。但是需要谨慎操作，确保不会影响正在执行时事务。

ALTER SYSTEM KILL SESSION ‘sid,seriel# ‘以上示例代码中，Sid 和 SERIAL’需要替换为实际的会话 ID 和序列号。

3.2.3 重启数据库实例 :

作为最后的手段，如果无法终止绘画或释放锁来解决死锁问题，可以考虑重启数据库来清除死锁。

SHUTDOWN IMMEDIATE STARUP

需要注意的是，重启数据库会导致数据库不可用。因此应该在合适的时间进行，并且有备份和恢复计划。

在处理死锁问题时，需要谨慎操作，确保不会对数据库造成不可逆的影响。最好在处理之前备份数据库 , 并且在环境中谨慎操作。

4 SQL 表死锁 .

死锁是数据中两个或多个并发进程相互占用对方的资源，导致都在等待对方释放资源，从而陷入无法向前执行的状态。在 SQL 中 , 死锁通常发生在多个事务同时尝试对多个表进行修改时，并且这些事务按照不同顺序锁定了资源。

4.1 解决死锁常见的方法 :

①减少事务的大小和复杂度。

②保持事务简短并在一个批处理中。

③使用更低的隔离级别。如读提交（READ commt），以减少锁的争用。

④对于可能产生死锁的操作 , 可以尝试重新排序资源，访问的顺序。

⑤使用行级锁定，如果数据库支持 , 来减少死锁的可能性。

⑥设置超时 . 当一个事务等待超过预设的时间后，会自动释放所持有的锁。

⑦检测死锁并回滚其中的一个事务。

4.2 表锁分类

4.2.1 表级锁 : 可以分为 3 类：表锁，元数据锁和意向锁。

表锁有两种模式 : 表共享读锁和表排他锁。简称为读锁和写锁。如对某个表加锁时，可以使用如下命令：

// 读锁

Lock tables 表名 read.

// 写锁

lock tasle 表名 write.

某个请求对表加了读锁之后 , 其他请求可以继续获取读锁，但不能获得写锁。

某个请求对表加了写锁之后，其他请求不能获取写锁，也不能读锁。

释放当前会话的所有的表锁 , 可以使用如下命令 . unlock tables。

客户端断开连接时，锁也会自动释放。

4.2.2 意向锁

意锁分如下两种：

①意向共享锁（IS 锁）：给索引行加共享锁之前，必须先取得该表的 IS 锁，与表锁读锁兼容，与表写锁互斥。

②意向排他锁（IX 锁）：给索引行加排他锁之前，必须先取得该表的（IX 锁），与表锁写锁和表读锁都互斥。

意向锁的存在支持快速获取表锁。例如事务 A 对某个索引行加了排他锁，事务 B 申请表锁与写锁是要成功的话，事务 B 就可以修改表中任何一行，与事务 A 持有的行锁是冲突的。如果没有意向锁的，则需要遍历整个表索引行判断是否有行锁的存在，以避免发生冲突。

4.2.3 行锁

①行锁种类 :Record lock 记录锁，GAPLock 间隙锁，Next key,lock 临键锁

②加锁机制 加锁的对象是索引，加锁的基本单位是临键锁。在使用记录锁或者间隙锁就避免幻读的场景下，临键锁就会退化成记录锁或间隙锁。

③记录锁　　锁定单个索引行的锁，防止其他事务对此进行。update 和 delete。

④间隙锁 : 对范围内的数据加间隙锁。范围内不存在的记录叫作“间隙”。也会被锁住。间隙锁是一个左开右区间。这个区间可以是两个索引之间 (a,b)。第一个索引之前（一∞，a），最后一索引个 (b,+ ∞)。间隙锁只存在于可重复读隔离级别。目的是解决可重复隔离级别下幻读现象。间缝锁可以防止间隔内所有新数据被插入，以及防止已存在的数据被更新成间隙内的数据。

⑤临键锁：记录锁和间隙锁的组合。临键锁是一个左开右闭区间。这区间可以是两个索引之间 (a.b] 第一个索引之前 (一∞ ,a]，最后一个索引之后 (b,+ ∞)。

4.2.4 页级锁

页级锁是 MySQL 中，锁定粒度介于行级锁和表级锁中间的一种锁。表级锁速度快 , 但冲突多。行级冲突少 , 但速度。所以获取了表的页级，一次锁定相邻的一组记录，BDB 支持页级锁。

特点 : 开销和加锁时间介于表锁和行锁之间，会出现死锁。锁定粒度介于表锁与行锁之间 , 并发度一般。

对数据页，通常是连续的几个行加锁。控制并发事务对该页的访问。适用于数据较大且并发量较高的场景。

5 死锁的检测 .

5.1 死锁定理 .

①从有向图中找到既不阻塞又非孤立的结点进程 pi。在顺利情况下 , Pi 可以获得它所需要的资源不断向前推进 , 直至运行完毕。然后释放它所占资源而处于潜伏状态。这相当于在图上消去 Pi 所有的请求边和分配边，使之成为孤立结点。

②进程 Pi 所释放的资源可以唤醒因等待该资源而阻塞时的过程 Pj。 Pj; 又可获得它所需资源继续推进 , 直至运行完毕。然后释放所占有的全部资源，而处于潜伏状态。

③在实施了上述一系列化简后 , 若消去图中所有的边 , 则称该图是可完全化简的。

④若有向图不能通过任何进程予以化简 , 这称该图不可化简。

⑤状态为死锁状态的充分条件是 : 当且仅当状态的资源进程图不可完全化简。该充分条件被称为死锁定理。

5.2 检测死锁算法 .

假定系统中有 N 个进程 P1,P2,..Pn. 和 m 种资源 R,R2..Rm。一个类型的资源数目分别为 W1,W2...Wm。可表示为资源数目向量 W。在任一时刻 T, 资源分配矩阵表示为元素

Aij=/(Rj,pi)/ 表示为分配给进程 pi 的资源 Rj 的数目。行向量 Ai 表示进程 Pi 所获得之资源总数。请求矩阵可表示为 bij=/(pi,Rj)/，表示进程 pi 请求资源 Rj 的数目。行向量 Bi 表示进程 pi 所请求的资源总数。已假定 V=(V1,V2.。,Vm) 为可用资源向量，其中每个元素为 :Vj=Wj-∑aij。

采用矩阵表示法时 , 检测死锁的算法可描述如下：

①把某时刻 T 的可用资源向量 v(t) 赋予资源数目向量 w。

②把不用资源的行向量 Ai=0 记入表 L 中。

③对所有请求各种资源的数量，均小于相应资源现有数量的 j 行 , 做如下处理：

可将其进程 - 资源有向图化简 , 释放出资源 , 使可用资源数目增加。

此时 j 行已不再占有资源，将其记入 L 表中。

④最后 , 若不能把所有的行都记入 L 表中 , 则初始状态将发生死锁。

6 研究的展望

①数据库的结构分析。一般而言 , 数据库有 6 种结构。比较容易产生死锁的结构为网状结构。层次结构不容易产生死锁。在计算机网络结构中，层次结构是比较常用的结构。很多计算机网络方面的软件工具都能对网络性能进行分析。

②计算机死锁的发生 . 在数据库方面比较常见的是嵌套结构。高维度的嵌套结构是计算机死锁的一个研究方向，内层嵌套和嵌套层数对数据库的死锁影响很大。

参考文献

[1] 离散数学及其应用. 机械工业出版社.徐元通等译陈琼改编.

[2] 数据库系统概论. 高等教育出版社.王珊等编著.

[3] 计算机操作系统. 西北电讯工程学院出版社.汤子赢等编著

[4] 操作系统设计Xinu方法.电子工业出版社.唐李洋等译

[5] 数据库原理及应用教程.中国中信出版集团.人民邮电出版社.陈志伯主编

[6] 数据库原理及应用. 中国中信出版集团.人民邮电出版社.赵军民主编

[7] 数据库原理及应用.清华大学出版社.雷景生等编著

[8] 数据库原理及应用.机械工业出版社.何玉洁编著

[9] 数据库技术及应用实验教程.清华大学出版社.尹为民等编著

[10] Department of Defence.Trusted Computer System Evaluation Criteria .DoD5200.282 STD,1985

[11] The information Assurance Framework(IATF) Release 3.1.NASA. Sep.2002

[12] Time Seris Modeling for IDS Alert Management.In :Proceedings of ACM Symposium on information Computer and Communication Security.2006:102-113

[13] Techniques and Tool for Analyzing Instrusion Alerts.ACM. Transaction and System Security.2004.7