SQL Server 中事务和锁的管理

Transaction and Lock Management in SQL Server

徐晓阳（国家税务总局扬州税务进修学院　225002）

摘　要：SQL Server是能满足不同规模、不同行业需要的大型关系型数据库管理系统，在全国税务系统得到了广泛的使用。本文就其中的重点和难点-事务和锁的管理，通过具体例子，阐述了事务和锁的管理内容和方法。

关键词：事务　锁　SQL Server　死锁

1 引言

为了保证数据库数据的完整性，大型的数据库管理系统（如：SQL Server、Oracle、Sybase等）都提供了多种保证机制，这些机制有：约束、触发器、事务和锁的管理等。事务管理主要是为了保证一批相关的对数据库数据的操作能全部被完成，从而保证数据的完整性；锁机制主要是防止多个活动事务及多用户环境下数据更新时的冲突。本文通过实例来分析SQL Server中事务和锁的管理和应用。

2 事务管理

2.1 关于事务及事务日志

事务是SQL Server中的单个逻辑工作单元，在一个事务内的所有语句要么全部执行，要么全部不执行，在这些语句的执行中遇到任何错误，SQL Server实例的事务管理机制将使用日志文件中的信息回滚整个已经执行的语句，回滚不影响同时在数据库内工作的任何其他用户的工作，从而保证了数据库数据的完整性；另外在事务执行时由于某种原因中断了客户端和SQL Server 实例之间的通信，SQL Server 实例将在收到网络或操作系统发出的中断通知后也会自动回滚事务。

数据库管理系统中的事务必须具备四种属性：

（1）原子性（Atomicity）：事务内的所有操作是一个原子单位，是作为整体来完成的。

（2）一致性（Consistency）：事务内的所有操作修改的数据必须遵循数据库中的各种其他完整性的规则，如：约束、触发器等。

（3）隔离性（Isolation）：多个事务之间对数据的处理应互相不影响，一个事务所操作的一定是其他事务处理前或处理后的数据，否则会发生一些并发问题。隔离性是通过锁的机制保证的。

（4）永久性（Durability）：事务对数据库数据的修改将永久保存。

但事务中不能包括以下语句：

CREATE DATEBASE ALTER DATABASE DROP DATEBASE

LOAD DATABASE RESTROE DATEBASE RESTROE LOG

BACKUP LOG RECONFIGURE UPDATE STATISTICS

事务日志是记录数据库中已发生的所有修改和执行修改的事务（事务的启动和结束）的一系列记录。它记录的内容可能是数据修改过程中所执行的逻辑操作（如一些大的操作CREATE INDEX），也可能记录数据修改前、后的影像。每个日志记录都被分配一个日志序列号（LSN）。

SQL Server采用前写日志的方式保存事务日志，在这种方式下，数据修改操作的日志记录在“脏页”刷新之前先写入磁盘，从而保证了回滚操作的正常。

2.2 事务恢复和检查点机制

虽然现在计算机的软、硬件系统和网络环境的可靠性已经有了很大的提高，但做不到万无一失，各种人为的操作失误、病毒的破坏等都对数据库系统造成了很大的威胁，SQL Server 实例在事务日志中存储足够的信息来恢复（前滚）和撤消（回滚）构成事务的数据修改。

为了减少频繁对磁盘读写操作，提高系统运行的速度，SQL Server在内存中建立了一个缓冲区，来存放被操作的数据，等出现检查点的时候才将缓冲区中的数据写回磁盘。在检索数据时，先将数据读到缓冲区；修改数据时也是先将数据读到缓冲区，再对缓冲区内的数据进行修改。缓冲区中未写回磁盘的数据页面，叫做“脏页”。

SQL Server始终生成自动检查点。自动检查点的时间间隔基于系统活动情况和recovery interval 服务器配置选项的参数计算出来的。自动检查点的时间间隔可能有很大的变化。如果数据库只做了很少的修改，自动检查点的时间间隔就长。如果修改了大量数据，自动检查点将经常发生。

recovery interval选项指定 SQL Server系统可接受的数据库恢复时间（分钟）。

如果系统设置了数据库的trunc.log on chkpt选项，检查点机制还执行截断事务日志的操作。一般在进行数据库备份前，应该执行CHECKPOINT语句来强制执行一个检查点检查，以便将所有的“脏页”写入数据库，这样就可以减少以后数据库恢复时的时间（节省了前滚的时间）。

2.3 显式事务和隐式事务

（1）显式事务：是用户显式地执行SQL Server中的事务语句来定义的。

BEGIN TRANSACTION：标识一个显式事务的开始

COMMIT TRANSACTION（COMMIT WORK）：标识一个事务的结束，说明事务被成功提交。

ROLLBACK TRANSACTION（ROLLBACK WORK）：可以在事务的任何地方回滚已经完成的操作。

```
——创建事例表
use pubs
create table tb_transaction
    {
    id int primary key,
    name char(10)
    }
——进入显式事务模式
begin transation tran1
insert tb_ transaction values(01,'name1')
——下面语句与primary key约束冲突，引起事务的回滚操作
insert tb_ transaction values(01,'name2')
——提交事务
commit transaction
——显示结果说明没有行被插入
select * from tb_transaction
```

（2）隐式事务：隐式事务不需要BEGIN TRANSACTION语句来标识，一旦当前事务提交或回滚即自动进入下一个事务。提交和回滚仍然使用语句COMMIT和ROLLBACK。

执行SET IMPLICIT_TRANSACTIONS ON语句可以使SQL Server进入隐式事务模式。

```
——进入隐式事务模式
SET IMPLICIT_TRANSACTIONS ON
——第一个隐式事务开始
insert tb_transaction values(02,'name2')
——显示第一个隐式事务插入的数据（结果显示id为02的数据行已经插入）
select * from tb_transaction
——提交第一个隐式事务
commit transaction
——第二个隐式事务开始
insert tb_transaction values(03,'name3')
——显示第一、二个隐式事务插入的数据（结果显示id为02、03的数据行已经插入）
select * from tb_transaction
——回滚第二个隐式事务所做的插入
rollback transaction
——显示回滚第二个隐式事务后的数据情况（结果显示id为02的数据行存在，而id为03的数据行已经被回滚）
select * from tb_transaction
```

（3）自动事务模式：在SQL Server中自动事务模式是默认的事务管理模式，在这种模式下当一个语句执行完成后即被提交并进入下一个语句事务。

执行SET IMPLICIT_TRANSACTIONS OFF语句可以使SQL Server进入自动事务模式

3 锁的管理

3.1 并发问题

在分布式数据库应用系统中，对数据的并发操作是经常发生的，给数据库资源加锁可以保证事务的完整性和数据库的一致性，避免出现并发问题。并发问题有：

（1）更新的丢失：当两个用户同时更新同一个数据时，系统只能保留最后一个数据的修改。

（2）脏读：脏读发生在一个事务读取了另一个事务已经更新但未提交的数据，更新数据的事务可能还要继续更新数据或可能进行回滚操作，这样一个不确定或根本不存在的数据可能被第一个事务使用。

（3）非重复读：当事务不止一次的读取相同的行，但在两次读取中间有另一个事务刚好修改了数据，那么两次读取的数据将出现差异。

（4）幻影读：当事务不止一次的读取相同范围内的行，但在两次读取中间有另一个事务刚好插入或删除了数据，那么两次读取的相同范围内数据的行数发生了变化。

3.2 可锁定资源和锁类型

SQL Server中有各种粒度的锁来锁定数据库资源，可锁定的资源有：

RI —— 行标识符，锁定单行数据

Key —— 键值，具有索引的行

Page —— 一个8K的数据或索引页面

Extent —— 区域，连续的8K页面

Table —— 整个表，包括数据和索引

Database —— 整个数据库，在数据库恢复期间使用

在锁定各种资源时可以使用不同的锁类型：

共享锁（S）：用于读数据操作，它允许多个事务读取相同的数据，但禁止其他事务修改此数据。

更新锁（U）：修改数据的事务在修改初期先获得更新锁，在进行修改操作前再将更新锁升级为排它锁（X）。

排它锁（X）：用于修改数据，它锁定的资源不能被其他事务读取和修改。

另外还有意向共享锁（IS）、意向排它锁（IX）、意向共享排它锁（SIX）、大容量更新锁（BU）、Sch-M、Sch-S。

在事务处理过程中SQL Server会自动的为事务选择合适的锁类型和锁定粒度，并能动态的调整锁定的粒度。

3.3 锁的相容性

锁的相容性说明在同一个资源上已经有锁的情况下，还有哪些其他的锁可以被申请到。

	现有的授权模式					
请求模式	IS	S	U	IX	SIX	X
意向共享 (IS)	是	是	是	是	是	否
共享 (S)	是	是	是	否	否	否
更新 (U)	是	是	否	否	否	否
意向排它 (IX)	是	否	否	是	否	否
与意向排它共享 (SIX)	是	否	否	否	否	否
排它 (X)	否	否	否	否	否	否

3.4 管理锁

调用系统存储过程sp_lock可以查看资源的锁定情况。语法为：

sp_lock [[@spid1 =] 'spid1'] [, [@spid2 =] 'spid2']

其中：spid1和spid2为进程标识号

下面我们通过实例来验证并分析事务和锁的情况。

我们先在SQL Server实例中建两个用户（user1和user2）

（1）以user1用户连接SQL Server的查询分析器，执行下列语句：

```
use pubs
go
-启动事务，修改titles表中titles的'BU1032'行的数据，注意不要提交事务
begin transaction
update titles
    set price=$10
    where title_ID='BU1032'
-查看锁定情况
exec sp_lock
```

（2）显示锁定情况如下：

spid	dbid	ObjId	IndId	Type Resource	Mode	Status
51	5	0	0	DB	S	GRANT
51	5	2121058592	0	TAB	IX	GRANT
51	5	2121058592	1	PAG 1:205	IX	GRANT
51	5	2121058592	1	KEY (a70064fb1eac)	X	GRANT
51	1	85575343	0	TAB	IS	GRANT

说明：spid-进程ID、dbid-数据库ID、ObjId-对象ID、IndId-索引ID、Type-锁定资源类型、Resource-锁定资源类型的信息、Mode-锁的类型、Status-锁状态

（3）以user2用户连接SQL Server的查询分析器，执行下列语句：

-等待解除排它锁，可以看到以下select语句一直在等待user1用户释放对titles的'BU1032'行的排它锁

```
use pubs
go
select * from titles
go
```

（4）在user1的查询分析器中，执行下列语句提交刚才未提交的事务；提交后user2的查询语句将被继续执行：

```
——解除排它锁
commit
go
```

3.5 死锁及减少死锁现象的策略

在多用户环境下，当两个事务都锁定了不同的资源又都在申请对方锁定的资源时就发生了死锁，死锁是难免的， SQL Server能定期的搜索并结束死锁。SQL Server处理死锁的步骤有：

（1）回滚投入时间最少的事务；

（2）向被回滚的事务发出1205号错误消息，应用程序可以捕捉错误消息并进行处理；

（3）让未回滚事务继续处理。

为了减少死锁的现象，应采取以下策略：

① 在所有的事务中以相同的次序使用资源；

② 通过极小化步数缩短事务；

③ 避免在事务内和用户进行交互，减少对资源的锁定时间。

参考文献

1 Microsoft SQL Server2000 数据库编程，（美）Microsoft公司著，北京希望电子出版社 2001。

2 中文版 SQL Server2000 数据库系统管理，袁鹏飞著，人民邮电出版社 2001。