SQL Server基础介绍

SQL 默认自带数据库

master

master 数据库保存SQL Server实例范围内的元数据信息、服务器配置、实例中所有数据库的信息，以及初始化信息

model

model数据库是新数据库的模板。每个新创建的数据库最初都是model的一个副本。所以，如果想在所有新创建的数据库中都包含特定的对象(比如数据类型)，或者是所有新创建的数据库中都以特定的方式来配置某些数据库属性，就可以先把这些对象或配置属性放在midel数据库中。注意：对model数据库做出的修改不会影响现有的数据库，只会影响此后创建的数据库

tempdb

tempdb数据库是SQL Server保存临时数据的地方，这些临时数据包括工作表、排序表、行版本控制信息等等。SQL Server允许用户为自己的使用而创建临时表，这些临时表的物理保存位置就是tempdb.

注意：每次重新启动SQL Server实例时，会删除这个数据库的内容，并将其重新创建为model的一个副本。因此，当需要为测试目的而创建一些对象，而且在测试完成后不想将这些对象继续保存在数据库中时，通常可以在tempdb中创建它们。即使忘记清除这些对象，在重新启动后也会自动清除它们

msdb

msdb是称为SQL Server Agent的一种服务保存其数据的地方。SQL Server Agent负责自动化处理，包括记录有关作业、计划和报警等实体的信息。SQL Server Agent也是负责复制的服务，msdb还用于保存一些有关其他SQL Server功能的信息，例如Database Mail和Service Broker

文件扩展名 .mdf,.ldf,.ndf

数据库文件扩展名.mdf和.ldf的含义很明确。扩展名.mdf代表Master Data File（主要数据库文件，不要与master数据库混淆）

ldf代表Log Data File（日志数据文件）。当讨论辅助数据文件的扩展名时，一种有趣的说法是，.ndf代表 Not Master Data File（非主要数据文件），某位开发人员提出了这种开玩笑的说法，业界也就这么接收了

架构和对象

一个数据库包含多个架构，而每个架构则又包含多个对象。可以将架构看作是各种对象的容器，这些对象可以是表、视图、存储过程等

可以在架构级别上控制对象的访问权限。例如，可以为一个用户授予某个架构上的select权限，让这个用户能够查询该架构中的所有对象的数据

表属于架构，而架构又属于数据库

select语句完整语法：

select 目标表的列名或列表达式序列

from 基本表名和（或）视图序列

[where 行条件表达式]

[group by 列名序列]

[having 组条件表达式]

[order by 列名[asc | desc]]

整个语句的执行过程如下：

1) 读取from子句中基本表、视图的数据，执行笛卡尔积操作；

2) 选取满足where子句中给出的条件表达式的元组；

3) 按group子句中指定列的值分组，同时提取满足having子句中组条件表达式的那些组；

4) 按select子句中给出的列名或列表达式求值输出；

5) order子句对输出的目标表进行排序，按附加说明asc升序排列，或按desc降序排列。

(8)SELECT (9)DISTINCT (11)<Top Num> <select list>

(1)FROM [left\_table]

(3)<join\_type> JOIN <right\_table>

(2)ON <join\_condition>

(4)WHERE <where\_condition>

(5)GROUP BY <group\_by\_list>

(6)WITH <CUBE | RollUP>

(7)HAVING <having\_condition>

(10)ORDER BY <order\_by\_list>

运算符优先级

1、()

2、\*,/,%

3、+,-

4、=,<,>,>=,<=,<>,!=,!>,!<

5、NOT

6、AND

7、BETWEEN,IN,LINKE,OR

8、=



数据库阻塞与死锁

对于一个数据库系统，尤其是有大量用户通过不同的应用程序同时访问同一个数据库的系统，如果发生一个或多个以下现象，管理员就应该检查数据库是否遇到了阻塞或死锁

1、并发用户少的时候，一切都还正常。但是随着并发用户的增加，性能越来越慢

2、客户端经常受到以下错误

Error 1222（错误1222）：lock request time out period exceeded（已超过了锁请求时段）

Error 1205 （错误1205）：your transaction（process ID # XX）was deadlocked on resources with another process and has been chosen as the deadlock victim.return your transaction（事务（进程ID XX））与另一个进程被死锁在XX资源上，并且已被选作死锁牺牲品。请重新运行该事务）、

超时错误：timeout expired. the timeout period elapsed prior to completion of the operation or the server is not responding

3、应用程序运行很慢，但是SQL Server这里CPU和硬盘利用率很低。管理员运行sp\_who这样的短小命令能很快返回

4、有些查询能够进行，但是有些特定的查询或修改总是不能返回

5、重启SQL Server 就能解决，但是有可能跑一段时间以后又会出现问题

造成阻塞和死锁的3大因素：连接持有锁时间过长，数目过多，粒度过大，以及它们产生的原因，从而能够知道如何去缓解阻塞和死锁

锁是SQL Server实现事务隔离的一部分，阻塞正是事务隔离的体现。要实现事务的隔离，阻塞不是SQL Server自找的，而是事务对 SQL Server提出的要求，也是用户使用事务所有付出的代价。一个数据库开发这和管理者的工作，不是 去消除阻塞，而是把阻塞的时间和范围控制在一个合理的范围之内，使最终用户既能享受事务的ACID，又能享受预期的性能。我全消除阻塞，是不可能的事情

数据库引擎可以锁定的资源

RID 用于锁定堆（Heap）

KEY 用于锁定索引上的某一行，或者某个索引键

PAGE 锁定数据库中的一个8KB页，例如数据页或索引页

EXTRNT 一组连续的8页

HoBT 用于锁定表下面的某一个分区(partition)

TABLE 锁定包所有数据和索引的整个表

FILE 数据库文件

APPLICATION 应用程序专用的资源

METADATA 元数据锁

ALLOCATION\_UNIT 分配单元

DATABASE 整个数据库

数据库引擎使用的资源锁模式

共享(S) 用于不更改或不更新数据的读取操作，如SELECT语句

更新(U) 用于可更新的资源中。防止多个会话在读取、锁定以及随后可能进行的资源更新时发生常见形式的死锁

排他(X) 用于数据修改操作，例如INSERT、UPDATE或DELETE。确保不会同时对同一资源进行多重更新

意向 放置在资源层次结构的一个级别上的锁，以保护较低级别资源上的共享或排他锁。如在对表内某个页实施排他锁之前，先在该表上放置意向锁。如果第二个事务试图在该表级别上应用共享锁，则受到阻塞。第二个事务不必检查表内各个页锁，只需要检查表上的意向锁即可，以提升性能。意向锁包含三种类型：意向共享(IS)、意向排他(IX)和意向排他共享(SIX)

架构 在执行依赖于表架构的操作时使用。架构锁包括两种类型：架构修改(Sch-M)和架构稳定性(Sch-S)

大容量更新(BU)在向表进行大容量数据复制且指定了TABLOCK 提示时使用

键范围 当使用可序列化事务隔离级别时保护查询读取的行的范围

1、共享锁

共享锁(S锁)允许并发事务在封闭式并发控制下读取(SELECT)资源。资源上存在共享锁(S锁)时，任何其他事务都不能修改数据

2、更新锁

在可重复读或序列化事务中，一个修改需要先读取数据[获取资源（页或行）的共享锁(S锁)，然后修改数据[此操作要求锁转换为排他锁(X锁)]]。如果两个事务获得了同一个资源上的共享模式锁，然后试图同时更新数据，则事务会把共享锁转换为排他锁(X锁)。由于两个事务都要转换为排他锁(X锁)，并且每个事务都必须等待另外一个事务释放共享锁之后才能得到排他锁，以至于两个事务都无法完成转换，因此发生死锁

为了避免这种潜在的死锁问题，SQL Server使用更新锁(U锁)。一次只有一个事务可以获得资源的更新锁(U锁)。事务真正修改数据时，将更新锁(U锁)转换为排他锁(X锁)

3、排他锁

排他锁(X锁)可以防止并发事务对资源进行访问。使用排他锁时，任何其他事务都无法读取或修改数据，仅在使用NOLOCK提示或未提交读隔离级别时才会进读取操作

数据修改语句(如INSERT、UPDATE和DELETE)合并了读取和修改操作，语句在执行所需要的修改操作之前，需要先执行读取操作以获取数据，因此，数据修改语句通常会先请求共享锁和排他锁。例如UPDATE语句可能根据与一个表的连接修改另外一个表中的行。在此情况下，除了请求更新行上的排他锁之外，UPDATE语句还将请求在连接表中读取的行上的共享锁

4、意向锁

数据库引擎使用意向锁来保护锁层次结构的底层资源，以防止其他事务对自己锁住的资源造成伤害，提高锁冲突检测性能

例如，当读取表里的页面时，在请求页共享锁之前，事务在表级请求共享意向锁。这样可以防止其他事务随后在表上获取排他锁，修改整个表格。意向锁可以提高性能，因为数据库引擎仅在表级检查意向锁，确定事务是否能安全地获取该表上的锁，确定事务是否能安全地获取该表上的锁，而不需要检查表中的每行或每页上的锁以确定事务是否可以锁定整个表



脏读：当一个事务开始更新数据，但是一个事务并没有完成提交，这时候，第二个事务开始读取数据，把第一个事务所更改的数据读了出来。第二个事务读取的数据是临时的，而且很危险的，因为有可能第一个事务最终做roll back操作

不可重复读：在一个事务中，我们读取某一行，得到数据，这个时候，第二个事务对改数据进行了修改，然后第一个事务再次读取这个行时，发现数据被改变了。也就是在一个事务中，多次读取某一行数据，可能得到不同的结果，这称为不可重复读

幻读：在一个事务中，我们读取数据，发现没有特定的行。第一个事务还没结束。这个时候，第二个事务插入了该行数据，然后在第一个事务再次读取时，我们会发现该行突然出现了，这称为幻读

隔离级别

未提交读(隔离事务的最低级别)

已提交读(数据库引擎的默认级别，可以防止脏读)

可重复读（可以防止脏读和不可重复读，但是有时候可能会出现幻读数据）

快照：使用行版本控制来防止幻读，同时减少阻塞

可序列号(隔离事务的最高级别，它要求事务序列化执行，事务只能一个接着一个地执行，不能并发执行)



