**MCP**

**主讲老师：张胜利**

**（一）用户自定义函数（UDF）**

* **分类：**

1. **标量值函数**

**返回值为一个数据点的函数被称为标量值函数**

**语法：**

**create function fun\_name(参数1 参数1类型,参数2 参数2类型……)**

**returns 返回值类型[with encryption] [as] begin**

**--函数内容 --return 返回值 End**

**调用标量值函数：**

**declare @s int set @s=dbo.FUNC\_Sum1(100,50) print @s**

1. **表值函数**

**返回值为一个结果集**

**a:内联表值函数**

**语法：**

**create function fun\_name(参数1 参数1类型,参数2 参数2类型……)**

**returns table [with encryption] --是否加密定义文本 [as]**

**return 一个查询的结果集**

**b:多语句表值函数**

**语法：**

**create function 函数名（参数）**

**returns 表格变量名table (表格变量定义) [with {Encryption | Schemabinding }] as begin SQL语句 end**

**调用表值函数 ：**

**select \* from dbo.FUNC\_UserTab\_1(15)**

**（二）锁（LOCK）**

* **HOLDLOCK 将共享锁保留到事务完成，而不是在相应的表、行或数据页不再需要时就立即释放锁。HOLDLOCK等同于SERIALIZABLE。此选项被选中时，SQL Server 会将此共享锁保持至整个事务结束，而不会在途中释放。**
* **NOLOCK 此选项被选中时，SQL Server 在读取或修改数据时不加任何锁。 在这种情况下， 用户有可能读取到未完成事务（Uncommited Transaction）或回滚(Roll Back)中的数据, 即所谓的“脏数据”。**
* **ROWLOCK 使用行级锁，而不使用粒度更粗的页级锁和表级锁。**
* **SERIALIZABLE 用与运行在可串行读隔离级别的事务相同的锁语义执行扫描。等同于HOLDLOCK。**
* **TABLOCK 使用表锁代替粒度更细的行级锁或页级锁。在语句结束前，SQLServer一直持有该锁。但是，如果同时指定HOLDLOCK，那么在事务结束之前，锁将被一直持有。这个选项保证其他进程只能读取而不能修改数据。**
* **TABLOCKX 此选项被选中时，SQL Server 将在整个表上置排它锁直至该命令或事务结束。这将防止其他进程读取或修改表中的数据。**
* **UPDLOCK 此选项被选中时，SQL Server 在读取数据时使用修改锁来代替共享锁， 并将此锁保持至整个事务或命令结束。使用此选项能够保证多个进程能同时读取数据但只有该进程能修改数据。**
* **XLOCK 使用排它锁并一直保持到由语句处理的所有数据上的事务结束时。可以使用PAGLOCK 或 TABLOCK指定该锁，这种情况下排它锁适用于适当级别的粒度**

1. **事务隔离级别**

**事务隔离级别通过影响读操作来间接地影响写操作；可以在回话级别上设置事务隔离级别也可以在查询（表级别）级别上设置事务隔离级别。**

**事务隔离级别总共有6个隔离级别：**

1. **READ UNCOMMITTED(未提交读,读脏),相当于(NOLOCK)**

**默认的读操作:需要请求共享锁,允许其他事物读锁定的数据但不允许修改.读操作不申请锁,运行读取未提交的修改,也就是允许读脏数据,读操作不会影响写操作请求排他锁.**

1. **READ COMMITTED(已提交读,默认级别)**

**是SQL SERVER默认的隔离级别，可以避免读取未提交的数据，隔离级别比READ UNCOMMITTED未提交读的级别更高;该隔离级别读操作之前首先申请并获得共享锁，允许其他读操作读取该锁定的数据，但是写操作必须等待锁释放，一般读操作读取完就会立刻释放共享锁。**

1. **REPEATABLE READ(可以重复读),相当于(HOLDLOCK)**

**保证在一个事务中的两个读操作之间，其他的事务不能修改当前事务读取的数据，该级别事务获取数据前必须先获得共享锁同时获得的共享锁不立即释放一直保持共享锁至事务完成，所以此隔离级别查询完并提交事务很重要。**

1. **SERIALIZABLE(可序列化)**

**对于前面的REPEATABLE READ能保证事务可重复读，但是事务只锁定查询第一次运行时获取的数据资源（数据行），而不能锁定查询结果之外的行，就是原本不存在于数据表中的数据。因此在一个事务中当第一个查询和第二个查询过程之间，有其他事务执行插入操作且插入数据满足第一次查询读取过滤的条件时，那么在第二次查询的结果中就会存在这些新插入的数据，使两次查询结果不一致，这种读操作称之为幻读。为了避免幻读需要将隔离级别设置为SERIALIZABLE。**

1. **SNAPSHOT(快照)**

**SNAPSHOT和READ COMMITTED SNAPSHOT两种隔离（可以把事务已经提交的行的上一版本保存在TEMPDB数据库中）SNAPSHOT隔离级别在逻辑上与SERIALIZABLE类似。  
READ COMMITTED SNAPSHOT隔离级别在逻辑上与 READ COMMITTED类似不过在快照隔离级别下读操作不需要申请获得共享锁，所以即便是数据已经存在排他锁也不影响读操作。而且仍然可以得到和SERIALIZABLE与READ COMMITTED隔离级别类似的一致性；如果目前版本与预期的版本不一致，读操作可以从TEMPDB中获取预期的版本。如果启用任何一种基于快照的隔离级别，DELETE和UPDATE语句在做出修改前都会把行的当前版本复制到TEMPDB中，而INSERT语句不需要在TEMPDB中进行版本控制，因为此时还没有行的旧数据。无论启用哪种基于快照的隔离级别都会对更新和删除操作产生性能的负面影响，但是有利于提高读操作的性能因为读操作不需要获取共享锁。SNAPSHOT 在SNAPSHOT隔离级别下,当读取数据时可以保证操作读取的行是事务开始时可用的最后提交版本。同时SNAPSHOT隔离级别也满足前面的已提交读,可重复读,不幻读;该隔离级别实用的不是共享锁,而是行版本控制。使用SNAPSHOT隔离级别首先需要在数据库级别上设置相关选项**

1. **READ COMMITTED SNAPSHOT(已经提交读隔离)**

**也是基于行版本控制,但是READ COMMITTED SNAPSHOT的隔离级别是读操作之前的最后已提交版本,而不是事务前的已提交版本，有点类似前面的READ COMMITTED能保证已提交读,但是不能保证可重复读,不能避免幻读,但是又比 READ COMMITTED隔离级别多出了不需要获取共享锁就可以读取数据**

**对于前四个隔离级别：READ UNCOMMITTED<READ COMMITTED<REPEATABLE READ<SERIALIZABLE**

**隔离级别越高,读操作的请求锁定就越严格,锁的持有时间久越长;所以隔离级别越高,一致性就越高,并发性就越低,同时性能也相对影响越大。**