**本文导读**：删除表中的数据的方法有delete,truncate, 其中TRUNCATE TABLE用于删除表中的所有行，而不记录单个行删除操作。TRUNCATE TABLE 与没有 WHERE 子句的 DELETE 语句类似；但是，TRUNCATE TABLE 速度更快，使用的系统资源和事务日志资源更少。下面介绍SQL中Truncate的用法

Truncate是一个能够快速清空资料表内所有资料的SQL语法。并且能针对具有自动递增值的字段，做计数重置归零重新计算的作用。

**一、Truncate语法**

[ { database\_name.[ schema\_name ]. | schema\_name . } ]  
    table\_name  
[ ; ]  
 **参数**

database\_name  
数据库的名称。

schema\_name  
表所属架构的名称。

table\_name  
要截断的表的名称，或要删除其全部行的表的名称。

**二、Truncate使用注意事项**

 1、TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同：二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。

 2、DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。

 3、TRUNCATE TABLE 删除表中的所有行，但表结构及其列、约束、索引等保持不变。新行标识所用的计数值重置为该列的种子。如果想保留标识计数值，请改用 DELETE。如果要删除表定义及其数据，请使用 DROP TABLE 语句。

 4、对于由 FOREIGN KEY 约束引用的表，不能使用 TRUNCATE TABLE，而应使用不带 WHERE 子句的 DELETE 语句。由于 TRUNCATE TABLE 不记录在日志中，所以它不能激活触发器。

 5、TRUNCATE TABLE 不能用于参与了索引视图的表。

 6、对用TRUNCATE TABLE删除数据的表上增加数据时，要使用UPDATE STATISTICS来维护索引信息。

 7、如果有ROLLBACK语句，DELETE操作将被撤销，但TRUNCATE不会撤销。

**三、不能对以下表使用 TRUNCATE TABLE**

 1、由 FOREIGN KEY 约束引用的表。（您可以截断具有引用自身的外键的表。）

2、参与索引视图的表。

3、通过使用事务复制或合并复制发布的表。

4、对于具有以上一个或多个特征的表，请使用 DELETE 语句。

5、TRUNCATE TABLE 不能激活触发器，因为该操作不记录各个行删除。

**四、TRUNCATE、Drop、Delete区别**

 1.drop和delete只是删除表的数据(定义),drop语句将删除表的结构、被依赖的约束(constrain)、触发器 (trigger)、索引(index);依赖于该表的存储过程/函数将保留,但是变为invalid状态。

2.delete语句是DML语言,这个操作会放在rollback segement中,事物提交后才生效;如果有相应的触发器(trigger),执行的时候将被触发。truncate、drop是DDL语言,操作后即 生效,原数据不会放到rollback中,不能回滚,操作不会触发trigger。

3.delete语句不影响表所占用的extent、高水线(high watermark)保持原位置不动。drop语句将表所占用的空间全部释放。truncate语句缺省情况下将空间释放到minextents的 extent,除非使用reuse storage。truncate会将高水线复位(回到最初)。

4.效率方面:drop > truncate > delete

5.安全性:小心使用drop与truncate,尤其是在 没有备份的时候,想删除部分数据可使用delete需要带上where子句,回滚段要足够大,想删除表可以用drop,想保留表只是想删除表的所有数据、 如果跟事物无关可以使用truncate,如果和事物有关、又或者想触发 trigger,还是用delete,如果是整理表内部的碎片，可以用truncate跟上reuse stroage，再重新导入、插入数据。

6.delete是DML语句,不会自动提交。drop/truncate都是DDL语句,执行后会自动提交。

7、drop一般用于删除整体性数据 如表，模式，索引，视图，完整性限制等；delete用于删除局部性数据 如表中的某一元组

8、DROP把表结构都删了；DELETE只是把数据清掉

9、当你不再需要该表时， 用 drop；当你仍要保留该表，但要删除所有记录时， 用 truncate；当你要删除部分记录时（always with a WHERE clause), 用 delete.

SQL控制事务之commit命令用法详解

COMMIT命令用于把事务所做的修改保存到数据库，它把上一个COMMIT或ROLLBACK命令之后的全部事务都保存到数据库。

这个命令的语法是： commit [work];

关键字COMMIT是语法中惟一不可缺少的部分，其后是用于终止语句的字符或命令，具体内容取决于不同的实现。关键字WORK是个选项，其惟一作用是让命令对用户更加友好。

在下面这个范例里，我们首先从表PRODUCT\_TMP里的全部数据开始：

select prod\_id prod\_desc cost from products\_tmp

12345 witches costume 29.99

222 false paraffin teeth 1.1

13 plastic pumpkin 7.75

90 lighted lanterns 14.5

4 rows selected.

接下来，删除表里所有低于14.00的产品。

delete from products\_tmp where cost<14;

2 rows deleted.

使用一个COMMIT语句把修改保存到数据库，完成这个事务。

COMMIT;

commit complete.

警告：对于数据库的大规模数据加载或撤消来说，应该多使用COMMIT语句；然而，过多的COMMIT语句会让工作需要大量额外时间才能完成。

记住，全部修改都首先被送到临时回退区域，如果这个临时回退区域没有空间了，不能保存对数据库所做的修改，数据库很可能会挂起，禁止进行进一步的事务操作。

注意：在某些实现里，事务不是通过使用COMMIT命令提交的，而是由退出数据库的操作引发提交。

但是，在某些实现里，比如MySQL，在执行SETTRANSACTION命令之后，在数据库收到COMMIT或ROLLBACK之前，自动提交功能是不会恢复的。

我认为需要根据实际情况进行取舍，例如表不复杂，可以由应用实现，若表之间关联较多且复杂，那么交由数据库处理，至少保证不会错。

存在主外键关联的主表，由于存在外键关联关系，因此有些操作就会禁止，例如truncate。

实验

1. 创建测试表   
SQL> create table tbl\_a(id number, remark varchar2(1));  
Table created.

SQL> create table tbl\_b(id number, a\_id number, remark varchar2(1));  
Table created.

SQL> alter table tbl\_a add constraint pk\_tbl\_a primary key(id);  
Table altered.

SQL> alter table tbl\_b add constraint pk\_tbl\_b primary key(id);  
Table altered.

SQL> alter table tbl\_b add constraint fk\_tbl\_b\_a foreign key(a\_id) references tbl\_a(id);  
Table altered.  
   
tbl\_a是主表，tbl\_b是子表，关联tbl\_a。

2. 现在主表和子表没有任何数据，此时执行truncate主表

SQL> truncate table tbl\_a;  
Table truncated.  
可以执行。

3. 向主表插入一条记录，再次执行truncate   
SQL> insert into tbl\_a values(1, 'a');  
1 row created.

SQL> commit;  
Commit complete.

SQL> truncate table tbl\_a;  
truncate table tbl\_a  
\*  
ERROR at line 1:  
ORA-02266: unique/primary keys in table referenced by enabled foreign keys  
   
此时提示了ORA-02266：唯一/主键被启用的外键引用

看看ORA-02266的解释：

02266, 00000, "unique/primary keys in table referenced by enabled foreign keys"  
// \*Cause: An attempt was made to truncate a table with unique or   
//        primary keys referenced by foreign keys enabled in another table.  
//        Other operations not allowed are dropping/truncating a partition of a  
//        partitioned table or an ALTER TABLE EXCHANGE PARTITION.  
// \*Action: Before performing the above operations the table, disable the   
//          foreign key constraints in other tables. You can see what   
//          constraints are referencing a table by issuing the following   
//          command:  
//          SELECT \* FROM USER\_CONSTRAINTS WHERE TABLE\_NAME = "tabnam";

比较清楚地说明了问题，以及解决方法：可以在执行前，先禁用外键约束，执行truncate后再恢复外键约束。

4. 禁用外键约束，删除后执行恢复操作

看到外键约束名称：FK\_TBL\_B\_A：   
 SQL> select constraint\_name, constraint\_type, status from user\_constraints where table\_name='TBL\_B';

CONSTRAINT\_NAME                C STATUS  
------------------------------ - --------  
PK\_TBL\_B                      P ENABLED  
FK\_TBL\_B\_A                    R ENABLED

禁止外键约束：   
SQL> alter table tbl\_b disable constraint FK\_TBL\_B\_A;

Table altered.

SQL> select constraint\_name, constraint\_type, status from user\_constraints where table\_name='TBL\_B';

CONSTRAINT\_NAME                C STATUS  
------------------------------ - --------  
PK\_TBL\_B                      P ENABLED  
FK\_TBL\_B\_A                    R DISABLED

STATUS状态变为DISABLED了。

truncate表：   
SQL> truncate table tbl\_a;

Table truncated.

恢复约束：   
SQL> alter table tbl\_b enable constraint FK\_TBL\_B\_A;

Table altered.

SQL> select constraint\_name, constraint\_type, status from user\_constraints where table\_name='TBL\_B';

CONSTRAINT\_NAME                C STATUS  
------------------------------ - --------  
PK\_TBL\_B                      P ENABLED  
FK\_TBL\_B\_A                    R ENABLED

总结：   
   
1. 主外键是数据库提供的强约束，可以帮助我们控制主子表之间的关系，但同时还是一把双刃剑，当然，我们认为既然定义了主外键，就是需要这种强制关系，但有时可能就会有一些变更，因此，如何取舍，需要根据实际情况来决策。

2. 主外键关联中的主表，如果有数据，则不能直接用truncate方式删除，因为会认为有外键和其关联，不能直接截断主表，若需要做，可以先禁止外键约束，主表变成一个独立的表，这样就可以执行truncate了。

SQL Server 临时禁用和启用所有外键约束

--获得禁用所有外键约束的语句  
select  'ALTER TABLE ['  + b.name +  '] NOCHECK CONSTRAINT ' +  a.name +';' as  禁用约束     
from  sysobjects  a ,sysobjects  b       
where  a.xtype ='f' and  a.parent\_obj = b.id

--获得启用所有外键约束的语句  
select  'ALTER TABLE [' + b.name +  '] CHECK CONSTRAINT ' +  a.name +';' as  启用约束       
from  sysobjects  a ,sysobjects  b       
where  a.xtype ='f' and  a.parent\_obj = b.id

*[代码]* SQL Server禁止全部约束应用到数据表1ALTERTABLEtb12NOCHECK CONSTRAINTALL3GO*[代码]* SQL Server解禁全部约束应用到数据表1ALTERTABLEtb12CHECKCONSTRAINTALL3

GO

SQL约束控制  
 1)禁止所有表约束的SQL   
select 'alter table '+name+' nocheck constraint all ' from sysobjects where type='U'  
 2)删除所有表数据的SQL   
select 'truncate table '+name from sysobjects where type='U'  
 3)恢复所有表约束的SQL   
select 'alter table '+name+' check constraint all ' from sysobjects where type='U'  
 4)删除某字段的约束  
 declare @name varchar(100)   
 --DF为约束名称前缀  
 selectb.name from syscolumns a,sysobjects b where a.id=object\_id(''表名'') and b.id=a.cdefault ''字段名'' and b.name like ''DF%''   
 --删除约束  
 alter table 表名 drop constraint @name   
 为字段添加新默认值和约束  
 ALTER TABLE 表名 ADD CONSTRAINT @name DEFAULT (0) FOR [   
 对字段约束进行更改  
 --删除约束  
 ALTER TABLE tablename   
 Drop CONSTRAINT 约束名  
 --修改表中已经存在的列的属性（不包括约束，但可以为主键或递增或唯一）  
 ALTER column 列名 int not null   
 --添加列的约束  
 ALTER TABLE tablename   
 ADD CONSTRAINT DF\_tablename\_列名 DEFAULT(0) FOR 列名  
 --添加范围约束  
 alter table tablename (''M'',''F''))

 SQL 禁用开启表的所有约束 2000  
--禁用所有约束  
exec sp\_msforeachtable "alter table ? nocheck CONSTRAINT all"  
--再启用所有外键约束

exec sp\_msforeachtable "alter table ? check constraint all"

2005  
EXEC dbo.sp\_MSforeachtable 'ALTER TABLE ? NOCHECK CONSTRAINT ALL'

EXEC dbo.sp\_MSforeachtable 'ALTER TABLE ? CHECK CONSTRAINT ALL'