daily_draft

笔记本:	daily_draft		
创建时间:	2021/6/3 10:52	**更新时间:**	2021/6/6 15:36
作者:	134exetj717		
URL:	https://www.w3school.com.cn/sql/func_convert.asp		

2021/6/3

在搜索怎么建立聚簇索引，因为sqlserver在建立表的时候会自动建立主键，而建立主键的过程中会默认创建唯一聚簇索引，但是聚簇只能有一个，而我们的作业是需要我们建立一个聚簇索引，因此在想应该怎么修改原有的聚簇索引，经过资料调查显示，sqlserver中没有自带的修改索引函数，我们只能够通过先删除再生成的方式修改原有的聚簇索引，但是我们发现，在删除默认聚簇的过程中。不能够直接删除带有实际命名的聚簇，那么此时问题就来了，既然不能够删除默认聚簇，那么我们如何才能创建一个新的聚簇呢，这个问题深深地困扰着我。

一系列的搜索中我们发现，既然sqlserver建立主键时就会默认生成聚簇索引，那么我们是否可以先把主键删除了然后再重新生成主键呢？答案是肯定的，我们删除主键以后，伴随着主键的唯一索引就是被删除，此时我们给表创建一个自定义的聚簇索引，然后再建立主键，此时由于表中已经有了聚簇索引，因而不会再默认生成唯一聚簇，此时，我们就完成了聚簇索引的创建过程。虽然繁琐，但是不得不说，对于sqlserver坑爹的默认，该方法相对来说是非常合理的，换一种做法，我们也可以在创建表的过程中直接自定义生成唯一索引，此时主键也同样不会默认生成聚簇了，虽然此时不符合我们的作业要求，但是也给了我们一个提醒，在创建表的过程中，我们就应该考虑到约束，完整性，索引等一系列属性，否则在后期的修改之中事情会非常复杂。

但是同时面临着一个新的难题，就是假如我们已经建立了多张表，不同的表之间可能分为主表和从表，在两张表的连接过程中我们往往会参照表的完整性约束，意思是我们在删除表的内容时，尤其是主表，需要考虑到从表是否给删除，因为有时候约束是 no action，代表主表的列不能够删除。那么既然列不能删除，往往代表主键不能够删除，那么我们应该怎么创建新的聚簇索引呢?

建立新的聚簇索引的方法有两种：一种是重新建表，顺便建立聚簇索引；另一种是删除原有默认自带唯一索引的主键，建立聚簇后再创建主键。

在删除原有主键约束的过程中，我发现我把 salary 的 p_no约束删除了，但是默认的聚簇索引还在，也就是并不能达到目的。

问题：向试图插入的数据不在视图中

书本p115

前提：SQL-92标准只允许对基于一张表的视图进行更新。

对视图插入一行，也会在基表中插入一行。

原因：

对视图的插入，有可能出现在基表中，而没有出现在视图中。是因为插入数据时可能不满足视图的相关插入要求，导致插入错误，但是在SQL-92标准中是默认允许这种插入的。所以就导致了我们没有看见视图插入成功。

解决办法：

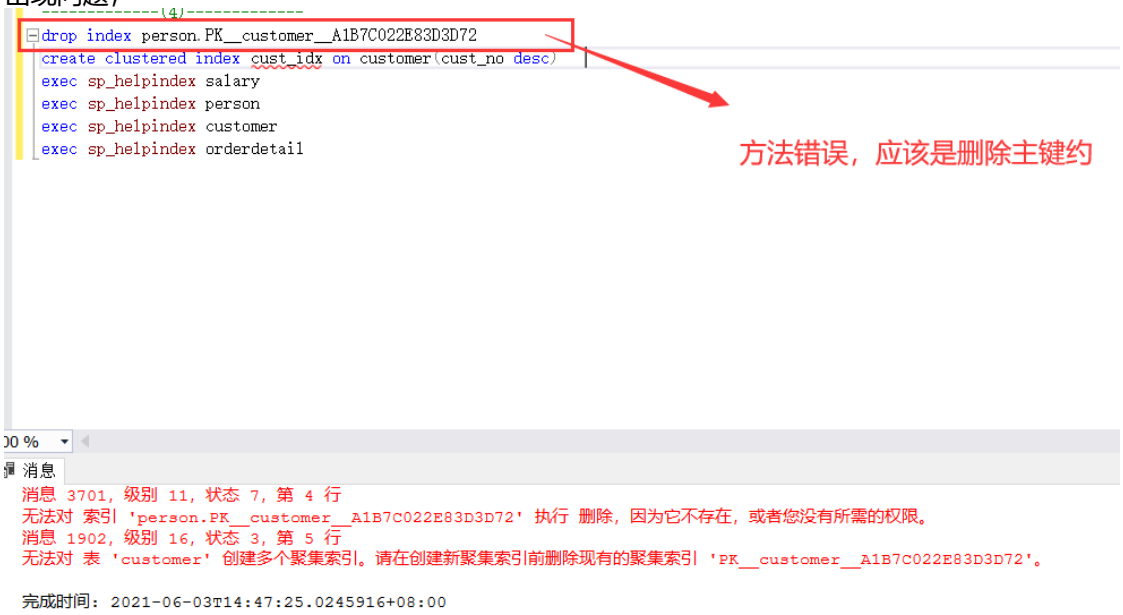
在视图定义时，加入with check option选项即可。

with check option的作用：

创建视图时我们一般都带有 where 的筛选语句，表中只有满足筛选条件的内容才会在我们的视图中正确显示出来。此时，当我们向视图中插入内容的时候，该数据也必须保证满足筛选条件，否则就插入失败，不然我们将数据插入了基表，但是因为该数据不显示在视图中，所以我们就在视图中看不到这样的数据。

批

针对与上述删除聚簇索引的问题，其实并不是删除错了带有实际命名的聚簇，而是我的删除方法出现问题，



convert() 函数是把日期转换为新数据类型的通用函数。如：convert(int,@date)是把date日期局部变量转变为int型。
Datepart()：返回代表指定日期的指定日期部分的整数

每次我们在使用查询分析器调试SQL语句的时候，通常会看到一些信息，提醒我们当前有多少个行受到了影响，这是些什么信息？在我们调用的时候这些信息有用吗？是否可以关闭呢？
答案是这些信息在我们的客户端的应用程序中是没有用的，这些信息是存储过程中的每个语句的 DONE_IN_PROC 信息。
当 SET NOCOUNT 为 ON 时，不返回计数（表示受 Transact-SQL 语句影响的行数）。当 SET NOCOUNT 为 OFF 时，返回计数。

server数据库中raiserror的作用就和asp.NET中的throw new Exception一样，用于抛出一个异常或错误。这个错误可以被程序捕捉到。

raiserror的常用格式如下：

raiserror('错误的描述',错误的严重级别代码，错误的标识，错误的描述中的参数的值(这个可以是多个)，一些其它参数)，在官方上的格式描述如下：

1. RAISERROR ({ msg_id | msg_str | @local_variable }
2. { ,severity ,state }
3. [,argument [,...n]])
4. [WITH option [,...n]]

其中，[,argument [,...n]]与 [WITH option [,...n] 两项是可以不写的。

分别解释一下各参数的用法：
一、{ msg_id | msg_str | @local_variable }
从这个参数中可以看出，这一项可能为三个值，
1,sys.messages中的自定义错误信息的错误信息号，自定义错误信息可以使用sp_addmessage存储过程添加到sys.messages中，注意, 用户定义错误消息的错误号应当大于 50000。
示例：raiserror(50001,16,1)
2,一条直接的错误描述，示例：raiserror('这里是错误描述的示例',16,1)

3,一个包含错误描述变量，示例:

```
1. declare @error_mes varchar(1000)
2. set @error_mes='这里是错误描述的示例'
3. raiserror(@error_mes,16,1)
```

二、severity

这个参数为用户定义的该错误信息的级别，我们可以指定 0 到 18 之间的严重级别。只有 sysadmin 固定服务器角色成员或具有 ALTER TRACE 权限的用户才能指定 19 到 25 之间的严重级别。若要使用 19 到 25 之间的严重级别，必须选择 WITH LOG 选项。

注意，如果错误级别在20~25之间，那么数据库会认为这个错误是致命，那么数据库会将该错误记录到错误日志和应用程序日志后终止数据库的连接。任何小于 0 的严重级别被认为等于0。大于 25 的严重级别被认为等于25。

三、state
这个参数是可以是1~127之间任意整数，可以用来标识错误的发生位置，如果一段代码的多个位置都会发生同样的错误，那么就可以将这个参数设置为不同的值，用来标识是那个位

置发生错误了。

系统给触发器提供了两张表，一张是inserted ，另一张是deleted。前者保存的是修改的新数据，后者保存的是删除的老数据。在展开删除，修改和插入触发器的过程中，只是单纯这三个触发操作提供两张表的方式不一样，最终我们判断执行三种操作还需要自行给出代码。

问题： 使用触发器时，对@@rowcount的使用出现问题，在修改表数据时总是会认为我一次性修改了多个数据而导致意外退出；

使用insert和update触发器时，注意区分这两个的判别，比如`if(select count(*) from deleted)=0 return 用于判断是否为 update`

使用批的好处是，每次用Go进行代码的提交，都能够通过一定的次序完成任务