**1、存储过程规范写法：**

**--不带事务管理**

create procedure sp\_test(@str varchar(10))

as

declare

@ls\_errmsg varchar(50),

@li\_errcode integer

begin

select @ls\_errmsg=''

select @li\_errcode=0

--涉及insert,update语句的后边必须判断错误码

delete bmzd where 1=2

**if @@error<>0**

begin

select @ls\_errmsg = '删除部门表失败!'+convert(varchar(10),@@error)

select @li\_errcode=-5 --删除部门错误

goto err

end

--正确返回时

**select @li\_errcode,@ls\_errmsg**

return 0

err:

--错误返回时

**select @li\_errcode,@ls\_errmsg**

return -1

end

**--带事务管理**

create procedure [dbo].[procname] (@zxr varchar(10),@zxrq smalldatetime,@hth varchar(20)='',@caller varchar(100)='')

as

begin

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

适用系统，总分是否一致，其他库是否与超市一致

调用者及调用时间、频次(作业)

功能说明，干了几件事

重点参数说明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--1、定义变量

declare @hs smallint, @fdbh varchar(4)

--2、定义临时表

create table #zzhth(hth varchar(20) null) --提前终止的合同

SET NOCOUNT ON

DECLARE @\_\_trancount INT

SELECT @\_\_trancount = @@TRANCOUNT -- 当前的事务数

SET XACT\_ABORT ON

BEGIN TRY

--初始化变量

--开始事务

IF @\_\_trancount = 0

BEGIN TRAN

ELSE

SAVE TRAN \_\_tranpoint

if @debug=1

insert syslog(lx,gh,xm,zymc,czsj,msg)

select 'S','lybdjs','存储过程','sp\_lybdjs',getdate(),'sp\_lybdjs计算保底开始,@caller'+@caller+'@hth'+case when @tcclbj='1' then '合同手工提前终止' else '' end+'@zxr'+@zxr+'@zxrq'+convert(varchar(10),@zxrq,120)

--处理代码

--异常信息校验

--正式处理

----关键字段或变量值说明

thend:

if @debug=1

insert syslog(lx,gh,xm,zymc,czsj,msg)

select 'S','lybdjs','存储过程','sp\_lybdjs',getdate(),'sp\_lybdjs计算保底结束,@caller'+@caller+'@hth'+case when @tcclbj='1' then '合同手工提前终止' else '' end+'@zxr'+@zxr+'@zxrq'+convert(varchar(10),@zxrq,120)

IF Xact\_state() = 1 AND @\_\_trancount = 0

COMMIT TRAN

SET XACT\_ABORT OFF

END TRY

begin catch

--判断有没有事务

-- 回滚事务

IF Xact\_state() <> 0

BEGIN

IF @\_\_trancount = 0

ROLLBACK TRAN

ELSE IF Xact\_state() = 1 AND @@TRANCOUNT > @\_\_trancount

ROLLBACK TRAN \_\_tranpoint

ELSE IF Xact\_state() = -1 AND @@TRANCOUNT >= @\_\_trancount

ROLLBACK TRAN

END

SET XACT\_ABORT OFF

exec sp\_cursor\_close 'cur\_mx'

-- 错误消息处理

DECLARE

@\_\_error\_number int,

@\_\_error\_message varchar(1000),

@\_\_error\_severity int,

@\_\_error\_state int,

@\_\_error\_line int,

@\_\_error\_procedure varchar(200),

@\_\_user\_name nvarchar(128),

@\_\_host\_name nvarchar(128)

-- a. 获取错误信息

-- 这提提取了错误相关的全部信息, 可以根据实际需要调整

SELECT

@\_\_error\_number = ERROR\_NUMBER(),

@\_\_error\_message = isnull(ERROR\_MESSAGE(),'Unknown error occurred'),

@\_\_error\_severity = ERROR\_SEVERITY(),

@\_\_error\_state = ERROR\_STATE(),

@\_\_error\_line = ERROR\_LINE(),

@\_\_error\_procedure = ISNULL(ERROR\_PROCEDURE(), 'sp\_lybdjs'),

@\_\_user\_name = SUSER\_SNAME(),

@\_\_host\_name = HOST\_NAME()

-- b. 对于重要的业务处理存储过程, 应该考虑把错误记录到表中备查(这个表需要先建立)

-- 记录错误应该在没有事务的情况下进行了, 否则可能因为外层事务的影响导致保存失败或重复记录

-- 记录内容根据需要自定义

IF Xact\_state() = 0

INSERT INTO syslog(lx,gh,xm,zybh,zymc,czsj,pcname,msg)

VALUES('U','lybdjs','联营保底',NULL,@\_\_error\_procedure,Getdate(),@\_\_host\_name,@\_\_error\_message)

--扔出错误，以便后续处理能检测到错误

RAISERROR(

N'P: %s, Ernum: %d, Erlv: %d, Line: %d, Err: %s ',

@\_\_error\_severity,

1,

@\_\_error\_procedure,

@\_\_error\_number,

@\_\_error\_severity,

@\_\_error\_line,

@\_\_error\_message)

end catch

end

**可以参照样例**

sp\_1

sp\_1\_1

sp\_2

sp\_2\_1

sp\_3

**sp\_gysxyd**

**需注意：分布式事务（如用链接服务器多库更新），不能启用事务保存点**

注：带不带事务，需要根据具体情况来定，大部分要用带事务处理的方式，不带事务如手持终端调用等可能带事务管理这个会报错。

**PB调用：**

string ls\_str,ls\_errtext

integer li\_ret

declare p procedure for sp\_test @str=:ls\_str,**@ret=:li\_ret,@errtxt=:ls\_errtext;**

execute p ;

**fetch P into :li\_ret,:ls\_errtext;**

**close P;**

if sqlca.sqlcode<>0 then

ls\_errtext=sqlca.sqlerrtext

**rollback;**

messagebox('提示','审核失败--'+ls\_errtext)

return

else

**if li\_ret=0 then**

**messagebox('提示','成功')**

**else**

**rollback;**

**messagebox('提示','失败')**

**end if**

end if

**2、关于函数**

函数不能调用过程等外部过程，不能处理外部表（临时表）

函数适用完成某些特定功能，不要做复杂的业务处理，这类应放在过程中处理更合适

**3、关于表变量**

只适用较小数据量使用，如小于5000行，较大的建议用临时表

表变量数据库不会预编译，会导致查询时系统开销很大，查询慢，可以在sql语句后加上OPTION(RECOMPILE) 让数据库在运行前进行一次精确编译以提升效率

**4、临时表清理**

建议用完后手工清理drop掉，尽管用完后数据库会自行回收销毁，但有时占用一定资源不能及时释放。

**5、关于XACT\_STATE()**

If 1, the transaction is committable.

If -1, the transaction is uncommittable and should be rolled back.

if 0, means there is no transaction and a commit or rollback operation would generate an error.

根据其值，做出处理

**6、过程sp\_syserr\_log\sp\_cursor\_close**

可以调用本过程抛出错误\关闭游标

**7、视图**

固定格式的数据可以生成视图，对外的数据要一律用视图，不要直接提供物理表

**8、关于临时表**

* 声明时以#tmp\_开头
* 字段显式指明是否允许为null，保证处理时无异常，对非通用字段加注释。
* 过程中注意null的处理

----涉及计算表达式时，要先判断null再进行运算,如:sum(isnull(fye,0)±isnull(kke,0))

----若涉及到变量，一是要赋初值，取完后再判断一下是否为null值，再进行后续处理

----这样做后保证处理时无异常

* 创建前先drop（注意使用了全局临时表不能drop），使用前先清空（如循环使用），使用后drop,以便及时清理资源占用。

**--附：高频SQL优化技巧9条**

**--附：SQL优化（系列）1-9**

**--附：SQL执行顺序**