The following **stored procedure** can be used to compare the differences between two databases. This procedure generates a report of rows inserted, updated, or deleted from each respective table. Typically, you would use this procedure to reverse engineer processes a program goes through to perform an operation by comparing the snapshots of before and after the specific process is performed.

This procedure is broken into three seperate procedures (same name, different override#):

- **sp_comparedb;1** compares two databases given by the parameters *@leftdb* and *@rightdb* .
- **sp_comparedb;2** returns the schema of a given object. This is used within the main procedure to determine which fields need to be compared for updates.
- **sp_comparedb;3** compares the respective object within two seperate databases (*@leftdb* and *@rightdb*), the object is expected to have the same name and creator in both databases.

```
 if(object_id('sp_comparedb') is not null)
drop procedure sp_comparedb
go
create procedure sp_comparedb(@leftdb varchar(32), @rightdb varchar(32)) as
/* Compares objects in two seperate databases for differences. It
   uses overloaded procedures to compare the actual objects.
*/
begin
set nocount on

declare @sql varchar(8000), @id int, @name varchar(32), @user varchar(32), @xtype char(4)

-- get objects from lhs database
select @sql='select id, name, [user]=user_name(uid), xtype from ['+@leftdb+']..sysobjects
 where xtype in (''U'')'
create table #objects(
 id int,
 name varchar(32),
 [user] varchar(32),
 xtype char(4)
 )
insert into #objects
 exec(@sql)
```

```
declare csr_objects cursor for select id, name, [user], xtype from #objects
open csr_objects
fetch next from csr_objects into @id, @name, @user, @xtype
while @@fetch_status=0 begin
 exec sp_comparedb;3 @leftdb, @rightdb, @name, @user
 fetch next from csr_objects into @id, @name, @user, @xtype
 end
close csr_objects
deallocate csr_objects
end
go

create procedure sp_comparedb;2(@db varchar(32), @object varchar(32), @user
varchar(32)=null) as
/* Returns relevant schema data for a specified object.
*/
begin
-- select field information for table from system tables
declare @sql varchar(4096)
select @sql=
 'select c.[colorder], c.[name], c.[xtype], pkik.[keyno]'+
 'from ['+@db+']..sysobjects as o '+
 'join ['+@db+']..syscolumns as c on o.id=c.id '+
 'left outer join ['+@db+']..sysobjects as pko on pko.xtype="PK" and o.id=pko.parent_obj '+
 'left outer join ['+@db+']..sysindexes as pki on pki.id=o.id and pki.[name]=pko.[name] '+
 'left outer join ['+@db+']..sysindexkeys as pkik on pkik.id=o.id and pki.indid=pkik.indid and
pkik.colid=c.colid '+
 'where o.[xtype]="U" and o.[name]="'+@object+'" and c.xtype not in (34,35)'+
 'order by c.[colorder]'
EXEC(@sql)
end
go

create procedure sp_comparedb;3(@leftdb varchar(32), @rightdb
varchar(32), @object varchar(32), @user varchar(32)=null, @validate
char(1)='n') as
/* Compares two tables in seperate databases. Return Values: 0 - ok,
tables were compared and no differences were found. >0 - ok, tables
were compared and differences were found. -100 - The table schemas
differ between databases and are thus incomparable. -101 - An error
occurred comparing the tables
*/
begin
set nocount on
-- validate parameters
if(@user is null) select @user='dbo'
```

```
declare @sql varchar(8000), @haspk int, @hasnk int
-- create a table for holding the relevant schemas of the two tables
create table #lhschema( [colorder] int, [name] varchar(32), [xtype]
int, [keyno] int -- order in primary key, null=not in pkey )
create table #rhschema( [colorder] int, [name] varchar(32), [xtype]
int, [keyno] int -- order in primary key, null=not in pkey )
-- retrieve relevant schema for lhs and rhs tables
insert into #lhschema exec sp_comparedb;2 @leftdb, @object, @user
insert into #rhschema exec sp_comparedb;2 @rightdb, @object, @user
-- ensure table schema's are compatible
if((select count(*) from #lhschema as lhs full outer join #rhschema as
rhs on lhs.[colorder]=rhs.[colorder] and lhs.[name]=rhs.[name] and
lhs.[xtype]=rhs.[xtype] and ((lhs.[keyno] is null and rhs.[keyno] is
null) or (lhs.[keyno]=rhs.[keyno])) where (lhs.[colorder] is null) or
(rhs.[colorder] is null) )>0) begin print 'The table schema for
'"'+@object+'"' is incompatible between the two databases.' return -100
end
-- check to make sure there is a primary key
select @haspk=count(*) from #lhschema where keyno is not null
if(@haspk=0) begin print 'Warning: table '"'+@object+'"' has no primary
key. Treating all fields as primary, thus no non-key comparison will be
done.' update #lhschema set keyno=1 end
select @hasnk=count(*) from #lhschema where keyno is null
-- build the join 'on' clause for a full join comparison
declare @name varchar(32), @keyno int, @join_on varchar(8000), @lhwhere
varchar(8000), @rhwhere varchar(8000), @nkwhere1 varchar(8000),
@nkwhere2 varchar(8000), @nkwhere3 varchar(8000), @nkselect1
varchar(8000), @nkselect2 varchar(8000)
declare csr_comtable cursor for select [name], [keyno] from #lhschema
order by [colorder] open csr_comtable
select @nkwhere2='', @nkwhere3='', @nkselect2=''
fetch next from csr_comtable into @name, @keyno
while @@fetch_status=0 begin -- build key field join-on and where
clauses if(@keyno is not null) begin -- this column is a key field
if(@join_on is null) select
@join_on='((lhs.['+@name+']=rhs.['+@name+']) or ((lhs.['+@name+'] is
null) and (rhs.['+@name+'] is null)))', @lhwhere='rhs.['+@name+'] is
null', @rhwhere='lhs.['+@name+'] is null' else select
@join_on=@join_on+' and ((lhs.['+@name+']=rhs.['+@name+']) or
((lhs.['+@name+'] is null) and (rhs.['+@name+'] is null)))',
@lhwhere=@lhwhere+' and rhs.['+@name+'] is null', @rhwhere=@rhwhere+'
and lhs.['+@name+'] is null' if(@nkselect1 is null) select
@nkselect1='lhs.['+@name+']' else if(datalength(@nkselect1) < 7900)
select @nkselect1=@nkselect1+', lhs.['+@name+']' else select
@nkselect2=@nkselect2+', lhs.['+@name+']' end else begin -- build
non-key where clause comparison if(@nkwhere1 is null) select
```

```
@nkwhere1='((lhs.['+@name+']!=rhs.['+@name+']) and ((lhs.['+@name+'] is
not null) or (rhs.['+@name+'] is not null)))' else
if(datalength(@nkwhere1) < 7850) select @nkwhere1=@nkwhere1+'
or((lhs.['+@name+']!=rhs.['+@name+'])and((lhs.['+@name+'] is not
null)or(rhs.['+@name+'] is not null)))' else if(datalength(@nkwhere2)
< 7850) select @nkwhere2=@nkwhere2+'
or((lhs.['+@name+']!=rhs.['+@name+'])and((lhs.['+@name+'] is not
null)or(rhs.['+@name+'] is not null)))' else select
@nkwhere3=@nkwhere3+'
or((lhs.['+@name+']!=rhs.['+@name+'])and((lhs.['+@name+'] is not
null)or(rhs.['+@name+'] is not null)))' if(@nkselect1 is null) select
@nkselect1='[lhs.'+@name+']=lhs.['+@name+'],
[rhs.'+@name+']=rhs.['+@name+']' else if(datalength(@nkselect1) <
7900) select @nkselect1=@nkselect1+', [lhs.'+@name+']=lhs.['+@name+'],
[rhs.'+@name+']=rhs.['+@name+']' else select @nkselect2=@nkselect2+',
[lhs.'+@name+']=lhs.['+@name+'], [rhs.'+@name+']=rhs.['+@name+']' end
fetch next from csr_comtable into @name, @keyno end
close csr_comtable
deallocate csr_comtable
/* validate variables up to this point */
if((datalength(@nkwhere3)=8000) or (datalength(@nkselect2)=8000)) begin
select @keyno=count(*) from #lhschema print 'Table '"'+@object+'" has
'+convert(varchar(32),@keyno)+' columns and is not supported by this
procedure.' return -100 end
if(@validate='y') begin print 'Table '"'+@object+'" validated.' return
0 end
/* we must first get counts of rows to determine if we need to output
anything
*/
create table #counts( type char(1), cnt int )
-- find rows that exist only in lh table
insert into #counts EXEC('select ''L'', count(*)'+ 'from
['+@leftdb+'].['+@user+'].['+@object+'] as lhs '+ 'full outer join
['+@rightdb+'].['+@user+'].['+@object+'] as rhs '+ 'on '+@join_on+ '
where '+@lhwhere)
if(@@error!=0) begin print 'Error occured comparing table lhs
'"'+@object+'"'.' return -101 end
-- find rows that exist only in rh table
insert into #counts EXEC('select ''R'', count(*)'+ 'from
['+@leftdb+'].['+@user+'].['+@object+'] as lhs '+ 'full outer join
['+@rightdb+'].['+@user+'].['+@object+'] as rhs '+ 'on '+@join_on+ '
where '+@rhwhere)
if(@@error!=0) begin print 'Error occured comparing table rhs
'"'+@object+'"'.' return -101 end
-- find rows that exist in both with different non-key field values
if((@haspk>0) and (@hasnk>0)) begin insert into #counts
```

```
EXEC('select ''X'', count(*)'+ 'from
['+@leftdb+'].['+@user+'].['+@object+'] as lhs '+ 'join
['+@rightdb+'].['+@user+'].['+@object+'] as rhs '+ 'on '+@join_on+ '
where '+@nkwhere1+@nkwhere2+@nkwhere3) if(@@error!=0) begin print
'Error occured comparing table '''+@object+''' for non-key
equivelance.' return -101 end end
/* now output any rows if found
*/
if(select cnt from #counts where type='L')>0 begin print 'TABLE: LHS
['+@leftdb+'].['+@user+'].['+@object+'] (rows exist only in lhs)'
EXEC('select lhs.*'+ 'from ['+@leftdb+'].['+@user+'].['+@object+'] as
lhs '+ 'full outer join ['+@rightdb+'].['+@user+'].['+@object+'] as rhs
'+ 'on '+@join_on+ ' where '+@lhwhere) end
if(select cnt from #counts where type='R')>0 begin print 'TABLE: RHS
['+@rightdb+'].['+@user+'].['+@object+'] (rows exist only in rhs)'
EXEC('select rhs.* '+ 'from ['+@leftdb+'].['+@user+'].['+@object+'] as
lhs '+ 'full outer join ['+@rightdb+'].['+@user+'].['+@object+'] as rhs
'+ 'on '+@join_on+ ' where '+@rhwhere) end
-- find rows that exist in both with different non-key field values
if(select cnt from #counts where type='X')>0 begin print 'TABLE:
NKEY ['+@rightdb+'].['+@user+'].['+@object+'] !=
['+@rightdb+'].['+@user+'].['+@object+'] (non-equal non-key fields)'
EXEC('select '+@nkselect1+@nkselect2+' '+ 'from
['+@leftdb+'].['+@user+'].['+@object+'] as lhs '+ 'join
['+@rightdb+'].['+@user+'].['+@object+'] as rhs '+ 'on '+@join_on+ '
where '+@nkwhere1+@nkwhere2+@nkwhere3) end
-- get the total number of different rows
select @keyno=sum(cnt) from #counts
if(@keyno is null) begin select @keyno=0 print 'There was a problem
comparing table '''+@object+'''.' end
drop table #counts
drop table #lhschema
drop table #rhschema
return @keyno
end
```