select a.program\_name, a.hostname, a.hostprocesses from

master..sysprocesses a, master..syslogins b

where a.suid=b.suid and b.name=’transvr’

sp\_depends @objname='Trans', @column\_name='ttax'

select s.name, s.crdate, s.type from sysobjects s where s.crdate > '01/19/15'

get space used of all tables

select convert(varchar(30),o.name) AS table\_name,

row\_count(db\_id(), o.id) AS row\_count,

data\_pages(db\_id(), o.id, 0) AS pages,

data\_pages(db\_id(), o.id, 0) \* (@@maxpagesize/1024) AS kbs

from sysobjects o

where type = 'U'

order by table\_name

add users to database

sp\_adduser TBUONOR1, TBUONOR1, vision\_user

sp\_addlogin TBUONOR1, #VISION#

fix a user to be a vision user

sp\_changegroup 'vision-user', 'TBUONOR1'

select cmd, tran\_name, linenum, spid, object\_name(id,dbid) from master..sysprocesses

sp\_spaceused RepoRisk

sp\_repeat 'insert theTable values (###,2)', 10

inserts 10 records, ### is the iteration from 1 to 10

sp\_repeat 'insert theTable values (###,2)', 10, 00:10:00.000 same but sleeps for 10 minutes

to display all subsids on one line

select @s = @s + subsid + ' ' from Subsidiary

select @s

select version from sysprocedures

@@connections contains the number of logins or attempted logins

@@total\_read # of disk reads by Adaptive Server since it was last started.

@@total\_write # of writes

@@boottime the last time a server was rebooted

select rowcount(9,object\_id(‘Trans’))

select name,lockscheme(id) from sysobjects where type=’U’ and name = ‘Trans’

output parameters have specific syntax

declare @nticket char(8)

exec proc ... @nticket=@nticket output

Stored Proc Template

create table #tmptable (

fld1 char(10) null,

etc..

)

insert into #tmptable

( fld1, etc )

select ...

create unique clustered index tmp\_i1 on #tmptable(fld1)

declare tmp\_cursor cursor for

select fields from

#tmp\_table

for update of fld2, fld3

open tmp\_cursor

fetch tmp\_cursor into

@fld1, etc..

while(@@sqlstatus = 0)

begin

update #tmp\_table set fld1 = @fld1 where current of tmp\_cursor

fetch tmp\_cursor into

@fld1, etc..

close tmp\_cursor

deallocate tmp\_cursor

**cursors and transactions**

cursors are a like selecting 1 row at a time

they are useful because they will only page lock where bulk updates might table lock

update cursors REQUIRE a unique index on fields OTHER than the updated fields

the optimizer needs the index, so the open cursor would fail without it

----------------------------- DOFC with a check on sqlstatus after each fetch

DECLARE x-cursor CURSOR

FOR

SELECT fldA from tblA for UPDATE

OPEN x-cursor #create result est

FETCH x-cursor INTO @myfld

while (@@sqlstatus = 0) -- is 1 for error, 2 for EOF

begin

begin tran

IF @@error != 0 BEGIN rollback RETURN –70 END

commit tran

UPDATE table1 set..... where CURRENT OF x-cursor

CLOSE x-cursor #destroy result set, but not the query plan

DEALLOCATE CURSOR x-cursor #destroy cursor, you can re-declare it

----------------------------------- process row sets

DECLARE M\_Cursor CURSOR FOR SELECT sectype from bony\_rules

OPEN M\_Cursor

set cursor rows 10 for M\_Cursor -- allows you to return 10 at a time

fetch M\_Cursor -- fetches to stdout

select @@rowcount -- is a running total

--------------------------------------

by default cursors are READ ONLY, buy you can specify FOR UPDATE

cursors lock much the way a regular select would

**transactions**

puts exclusive lock on data (page) until you commit

select ... SHARED - bypasses the lock

begin transaction ABC

select.. update.. insert.. delete..

commit transaction ABC

**rollbacks**

rollback to outermost tran, "uncommit" any inner tran that has already been committed

-- @@trancount increments with each begin tran, decrements with commit

-- data not really committed until @@trancount = 0

-- trigger with a rollback tran will abort all subsequent code within the transaction block, but a stored procedure will not.

-- rollback trigger will only rollback the code in the trigger and the update that invoked the trigger

**system tables**

useful to use object\_id() and object\_name() functions

select object\_name( instrig ) from sysobjects where id = object\_id( "my\_table" )

to see the program information

select \* from master..sysprocesses where @@SPID = spid

other master db tables to monitor: syslocks, syslogins, sp\_who, sp\_lock

procudures -code is stored in syscomments

-db syb\_systemprocs (or master) stores sp\_ procedures

-these can be run from any db, though they would use objects from your db

sp\_rename (rename a file)

sp\_configure (shows everything about your server)

sp\_help sys\_trade (shows all columns and indexes)

sp\_spaceused sys\_trade (shows space used)

sp\_dboption (shows db options)

sp\_helpuser (shows all users)

sp\_helpuser dbo (shows all users aliased as dbo, or who dbo is aliased as)

sp\_helprotect (show authority to objects)

sp\_helpdevice (show devices)

sp\_depends sys\_trade (show what procs reference sys\_trade)

sp\_depends proc\_pvar (show what tables are referenced by proc\_pvar)

sp\_helptext proc\_pvar (shows the code line by line (may be different line#s))

sp\_sort (shows sorting rules)

sp\_helpdb

sp\_helpindex (constraint, device, group, joins)

sp\_recompile

sp\_monitor (more useful than top as top always shows 100% cpu usage by dataserver)

sp\_rename ‘bony\_rules.buid’, bonyID (rename a column)

sp\_statistics sys\_trade -- more concise then sp\_help

sp\_tables -- lists all tables such as syscolumns, sysindexes, sysusers, etc..

syslogs - records all changes to the db, aka transaction log

master..sysdatabases, sysengines, sysobjects, syscolumns, sysindexes, systypes, sysusers

---- all these files have an id field

select b.name, b.length from sysobjects a, syscolumns b where a.name = 'sys\_fifo\_unmatched' and a.id =b.id

- b.name = field heading a.name = file a.type = u (for user tables)

select \* from sysobjects order by type, name

(type -🡪 P = proc,T=trigger, R=rule, RI=index, S=system file, U=user file, V=view)

select suser\_id(‘TBUONOR1’) -- gives the numerical user id

select suser\_name(1377) -- gives the user name

select user\_id(‘TBUONOR1’) -- database id

sp\_helpuser -- the id is the database id

**grant/revoke**

permissions on tables, views, and procs help control access

sp\_adduser SCALZOR, SCALZOR, repo\_user // step 1, 3rd parm is optional (group)

sp\_addlogin SCALSOR, #GREEN00#, repo\_dev // step 2

sp\_role "grant", "sso\_role", BUONORAT

sp\_password #BUONORAT#, #GREEN00#, SCALZOR -- my password, user's new one, user

**sql**

joins and subqueries

sometimes you have to explicitly reference the full file name

update tblA set col1 = tblB.col1 from tblA, tblB where tblA.col0 = tblB.col0

UPDATE tblA SET status = 'S' where rtid = ( select rtid from tblB b where

b.trans\_date = tblA.trans\_date)

update #\_fo\_trades set proc\_status = 'X' where exists (select 1 from sys\_trade

where #\_fo\_trades.original\_dealid = external\_trade\_id)

update #abc set #abc.fld2 = (select #xyz.fldb from #xyz where #xyz.flda = #abc.fld1)

where exists (select 1 from #xyz where #xyz.flda = #abc.fld1)

select \* from fileA where fldA = (select fldB from fileB where......)

select \* from fileA where fldA in (select fldB from fileB where......)

select \* from fileA where fldA exists (select fldB from fileB where......)

--- a subquery must only return 1 for an update clause unless in or exists is used --

update tranfilp set cmam =0 where tname in (select bname from becfilep where bplno=‘TBA’)

update sys\_trade set dvdt = (select distinct mbs.dvdt from sys\_mbse\_trades mbs where strd.external\_trade\_id = mbs.external\_id

and mbs.curdt = $bus\_date)

where strd.trans\_date = $bus\_date and exists (select 1 from sys\_mbse\_trades where strd.external\_trade\_id = mbs.external\_id)

--- another to update 2 fields with a group from another table

update #ibr set RGSP\_R\_avgrate = ( select avg(#rates.rate) from #rates where #ibr.cusip = #rates.cusip

and #rates.trantype = 0 and #rates.bookcode = 'RGSP' group by #rates.cusip ),

RGSP\_RR\_avgrate = ( select avg(#rates.rate) from #rates where #ibr.cusip = #rates.cusip

and #rates.trantype = 1 and #rates.bookcode = 'RGSP' group by #rates.cusip )

select fields into #tempfile where.... creates table, not logged

insert into #tempfiles select fields where... table must exist, logged

SELECT bxpcb "Par Price" from becfilep -- same thing

select @MidStr = substring(@InputStr, 1, 14)

select @LastByte = right(@InputStr, 1)

select fidoID, min(cusip) from fidoIDs group by fidoID - quick way to grab uniques

select count(distinct \*) from.... -- select # of distinct values

select count(distinct cusip) from.... -- select # of cusips

select fidoID, cusip from fidoIDs f1

where ( select count(\*) from fidoIDs f2

where f1.fidoID = f2.fidoID ) > 1 - shows duplicates plus the cusips

or select \* into tbla from tblb where 1 = 0

create unique index (a,b,c) on tbla with ignore\_dup\_key

select 'entry\_date' = convert(char(7),entry\_date, 111), 'tot' = count(1)

from HistRepoRate group by convert(char(7),entry\_date, 111)

* trick to groups by month

note: to compare fields that might be null, you need to do this

isnull(a.key\_issue\_cdty\_code\_2,'') = isnull(b.key\_issue\_cdty\_code\_2,'')

or

isnull(a.key\_issue\_cdty\_code\_2,0) = isnull(b.key\_issue\_cdty\_code\_2,0)

select ... where ltrim(a.cusip) = ltrim(b.cusip) -only ltrim needed, sybase trims right

select ...where usr is null and convert(numeric(15,8),ask) <> convert(numeric(15,8),aask)

create table tempdb..impact\_prices (name char(31) null, cusip char(9) null, usr char(9) null, bid char(15), ask char(15))

create table tempdb..min\_prices (cusip char(15) null, bid char(15), ask char(15))

process\_code like 'ARCH\_S\_\_S' -positions 5,7 and 8 are any character

process\_code like 'ARCH\_S[TVP]S' -positions 5 is any character, 7 must be T, V or P

process\_code like 'ARCHS[A-Za-z]S' - position 7 must be a letter

c.customer\_name like @cact' + '%' -- ok to use a variable

^ means not, i.e. like 'Tom ^B' will match Tom A but not Tom B

select @msg = convert(char(5), @@rowcount) + “ rows selected”

select db\_name() --gives database name

declare @strvar varchar(255)

select @strvar = @strvar + snp from ratings\_map - concatenates all values into 1

select name, rowcnt(sysindexes.doampg)   from sysindexes     where name in -- get rowcount w/out the table scan  
         (select name from sysobjects where type = "U") order by 2 desc

in a table with 2 rows where fld3 is null in one of them

select count(2) from test1 - returns 2

select count(\*) from test1 - returns 2

select count(fld3) from test1 - returns 1, only non null values returned

sybase processes where, group, having in that order

having is a condition applied after the group, if you condition against a row value, i.e. not a aggregate function, then a row will display for every row that meets the condition

select fidoID, count(\*) from fidoIDs group by fidoID having count(\*)> 1

- shows duplicates

you can add a case statement in a group by list

sybase allows you to have a having condition different fields from the select, these leads to some odd results, sybase will give you extra rows to show what was selected on top of what meets the criteria

null is defined as unknown value, i.e. the value has not been specified or specified as null, it will register as false on any test

this construct would insert null into fld3, if fld3 does not allow null, an error is generated

insert into test1 (fld1, fld2) values ('A', 'B')

where fld1 < 10 would not return null values

in is an UNCORRELATED subquery, it runs once only

exists -- is a join, CORRELATED subquery (might be more efficient to use a temp table and join)

select a.ticket, a.cpty, a.face, a.tran\_date, a.mod\_date, a.subsid from Correct a where a.ticket in (select b.ticket from Trans b where b.subsid = 'LNBR' and b.tran\_date > '20050104')

select a.ticket, a.cpty, a.face, a.tran\_date, a.mod\_date, a.subsid from Correct a where exists (select 1 from Trans b where a.ticket = b.ticket and b.subsid = 'LNBR' and b.tran\_date > '20050104')

\* is it legal to say where something in (select values from tableB where tableA.field=tableB.field) ??

selecting rows N thru M: select all into temp table with identity column, then select where identity btween N and M

a stored proc could accept a page# as a parameter, @row\_start = @page\_num \* @rows\_per\_page

to select the rowcount from a group query (@@rowcount works but only after the sql)

select sectype, count(\*), sum(sign(count(\*))) from bony\_rules group by sectype

- count(\*) wont work since that would be a 1st pass function, but the sign function is evaluated 1st pass – always as 1, the sum is evaluated 2nd pass, thus its a record count

DELETE from tbl1 -logs every record, you can recover it

TRUNCATE TABLE tbl1 -no log, no triggers, resets identity ctr

@@rowcount @@servername @@version @@error

@@nestlevel - increases each time a proc/trigger calls another proc or trig,

cross join -select all records for every record (all possible permutations)

ALTER TABLE dbo.sys\_tp\_accounts modify bonyID char(10)

GRANT REFERENCES ON dbo.dnl\_loan\_rate TO systr\_users

GRANT SELECT ON dbo.dnl\_loan\_rate TO systr\_users

EXEC sp\_rename 'dnl\_loan\_rate.pk\_dlra','pk\_dlra\_10032000221442001'

convert a datetime....fmts 0 - 20 (YY), 100 – 116 (CCYY), convert(char(12), pdate, 112)

1 12/16/02 107 Dec 16, 2002 112 20021216

select @date\_time = convert(char(8), max(mod\_date),112) from ARTSADPRec

select @date\_time = @date\_time + ' 12:30' -- now you can use this

to convert a string convert(char(8),convert(datetime,'$bus\_date',106),112))

select substring(curdt,5,2) + '/' + substring(curdt,7,2) + '/' + substring(curdt,3,2)

(index starts at 1)

--- to test dates

set arithabort off

set arithignore on

go

declare @tst datetime, @msg varchar(12)

select @tst = '20040100'

if @tst is not null

begin

select @msg = convert(char(8), @tst,112)

print @msg

end

**exception handling**

check @@error after anything, selects, updates, cursor actions, fetch, etc

ASE errors invoke a msg # (1700-1999), a severlity level ( 10 thru 24), state#, msg

table master..sysmessages has this data

**ddl**

rules, indexes, triggers, unique key, primary key, foriegn key, default, check constraints, view

syntaxes: 1) creating table—constraint 2) create constraint 3) alter table add constraint

4) after the field itself

CONSTRAINT pk\_xyz PRIMARY KEY NONCLUSTERED (fldxyz)

-null not allowed w/pk

-can only have 1 pk

CONSTRAINT unq\_xyz UNIQUE CLUSTERED (my\_xyz)

-null ok -- default is to not allow null value for non datetimes

fldxyz char(10) PRIMARY KEY UNCLUSTERED

my\_xyz char(3) NOT NULL REFERENCES tableB(b\_xyz)

constraint fk\_sftr\_stra foreign key (trader) references sys\_trader(trader)

constraint chk\_svps\_position\_type check (position\_type in ('DEP', 'TRD', 'SEG'))

CONSTRAINT ck\_zip CHECK zip\_code LIKE '[0-9][0-9][0-9][0-9][0-9]'

CREATE CONSTRAINT or alter table ... add constraint

you can create messages and bind them to constraints, (add them with sp\_addmessage and sp\_bindmsg)

constraint violations will not succeed, but they will not rollback either...

rules amd defaults are independent constraints that are binded to fields across the database via sp\_bindrule and sp\_binddefault

create rule tprule1 as @mdy in ('Aaa', 'Aa1', 'Aa2')

sp\_bindrule tprule1, "Bond.moddy\_rating"

insert into Bond (moody\_rating) values('abc') -- error

CREATE DEFAULT dbo.sysdate AS getdate()

EXEC sp\_bindefault 'dbo.sysdate','cust\_delivery\_instruction.systr\_update\_date'

ALTER TABLE dbo.sys\_general\_ledger ADD CONSTRAINT idx\_sgel\_1 UNIQUE NONCLUSTERED (profit\_center,gl\_account,entity\_code,currency\_code)

ALTER TABLE dbo.dnl\_holiday ADD CONSTRAINT pk\_dhol PRIMARY KEY NONCLUSTERED (dhol\_tag)

CREATE CLUSTERED INDEX idx\_bony\_rules ON dbo.bony\_rules(bonyID,sectype) WITH ALLOW\_DUP\_ROW

ALTER TABLE dbo.sys\_entity\_instrument ADD CONSTRAINT fk\_sein\_sins FOREIGN KEY (sins\_tag) REFERENCES dbo.sys\_instrument (sins\_tag)

ALTER TABLE dbo.dnl\_loan\_rate ADD CONSTRAINT pk\_dlra PRIMARY KEY

NONCLUSTERED (dlra\_tag)

union removes duplicate rows from the result set whereas union all does not

the first query in the union statement may contain an into clause that creates a table to hold the final result set

table level options to handle duplicate rows (DOL table allows them)

allow\_dup\_row - allows them

ignore\_dup\_row - discards them but does not complain

ignore\_dup\_key - if a dupe is attempted, it is ignored with an infomational msg

2 arith errors, truncation and overflow

set arithabort off - do not abort if either error occurs (print warning)

set arithabort arith\_truncation - do not abort if truncation occurs, you can check @@error or null value of result field o look for error

arithignore arith\_overflow ( meaning you divided by 0 or put a # that was too large)

arithignore numeric\_truncation (meaning you put a floater into an int value)

set arithignore on - dont even print the warning

**triggers**

good for cascading deletes or updates, esp. where the update comes from many pgms

good for data integrity checks, can rollback a transaction

create trigger dbo.mytrig on sys\_trade for {insert, update, delete} as

update fintable set fintable.rate = inserted.rate

2 tables, inserted and deleted, used by triggers, -always in memory, not written to disk

they have the same columns as the trigger table, i.e. inserted record in sys\_trade exists in inserted

the table will only have one row at any one time

if update (cusip) and update (trader) begin ... end

update keyword allows for complex conditions

create trigger t3 on tranfilp for insert as

begin update cusfilep set curvdt = inserted.trdt where inserted.tcact = cusfilep.cucact end

create trigger BrokerFeeType\_t2 on BrokerFeeType for delete as

if exists (select Cancel.salescateg from Cancel, deleted where Cancel.salescateg = deleted.code)

begin

select "Unable to delete as BrokerFeeType: "+ deleted.code +" exists in Cancel" from deleted

rollback transaction

end

create trigger Book\_t1 on Book for update as if update (code)

begin

update Cancel

set Cancel.bookcode = inserted.code

from Cancel, inserted, deleted

where Cancel.bookcode = deleted.code

create trigger trg\_fo\_trds\_ins on sys\_fo\_trades for insert

as

begin

update sys\_fo\_trades set last\_amended\_date = getdate(), last\_amended\_process = (select program\_name from

master..sysprocesses where spid = @@spid)

where original\_dealid = (select inserted.original\_dealid from inserted) and revision\_number = (select

inserted.revision\_number from inserted) and event\_type = (select inserted.event\_type from inserted)

end

IF OBJECT\_ID('dbo.trg\_fo\_trds\_ins') IS NOT NULL - tests its existence

**identity column**

automated uniqueness, optimizer is “tuned” to work with them

not row numbers, if a row fails to insert, that identity # is lost

bcp –E (and set identity\_insert MyTable on) allows you to insert into the identity column

CREATE TABLE scratch..int\_fo\_position

( ifop\_tag numeric(10,0) IDENTITY,

batch\_number numeric(18,0) NOT NULL,

book\_id char(20) NOT NULL,

)

SET IDENTITY\_INSERT scratch..int\_fo\_position ON

or... select ident = identity(5) from .... – create an on the fly identity#

select SYB\_IDENTITY - identity column for any table

select @@identity - last identity# selected

**options**

Set nocount on (reduces network traffic if you dont need the rowcounts)

Set dateformat (changes the default)

Set showplan on (enables for the next qry only)

set noexec on (to see showplan w/o running query)

set statistics io on (use this w/other options to see the difference they make)

set statistics time on

set forceplan on (forces the order of your joins)

set prefetch on (lowers your chache, can force the use of a particular index)

set flushmessage on -returns messages immediately, otherwise messages arent returned until the query is finished

set rowcount 0 -- must do this to turn rowcount off!

options vary in their scope, and vary if the go must be immeidately after

**functions**

ABS, CHAR\_LENGTH, CHARINDEX, COUNT, DATEADD, DATEDIFF, GETDATE,LTRIM, LOWER, UPPER, SUBSTRING, RIGHT, CONVERT

waitfor time "16:23" -- waitfor cmd waits until a time

waitfor delay "01:30" -- or for a specific interval

charindex(‘BZW’, ‘ABCBZW’) - returns 4

stuff(‘onetwothree’,,4,3,’---‘) – returns “one---three”

str(5.6,12,2) - converts to ascii, i.e. “ 5.60”

avg date..dateadd(dd, round(avg(datediff(dd, '20000101', b.maturity\_date)),0),'20000101')

char() function translates int values, 10 is \n, 89 is 'Y'

select db\_name(), suser\_name(), show\_role(), host\_id()

select rtrim(trader), reverse(trader), upper(trader), max(trader), min(trader), avg(ctam) from sys\_trader

select price, FLOOR(price), CEILING(price), ROUND(price,4)

108.32153, 108, 109, 108.3215

str(100\*intrate, 5, 3)

can round and display a rate, i.e. 1.299999999999 to 1.30

--date parts are mm,dd,yy, hh,mi,ss

select DATENAME(month, getdate()) returns MAY

select DATEPART(month, getdate()) returns 5

dateadd(DD,10,’10/21/98’) returns 10/31/98

STR(123.45,5,1)

-rounds to 123.5, converts to char(5) and right justifies

CONVERT(char(6), 123.45)

-left justifies, no rounding

if a string overflows its length, you'll see asterixes, ex.

STR(123456,4) result is \*\*\*\*

ISNUMERIC(@fldA) -tests for digits regardless of datatype

select count(\*) from becfilep where ISNUMERIC(CUSIP) = 1

SELECT ISNULL(MAX(zip), 'No Value Found') from .....

-returns a zip, but if none found then

returns the 2nd value

select col\_length("sys\_trade", "trader"), user\_name(), host\_name()

select object\_id("sys\_trade") -the object id is a key in many tables

user defined functions – useful to wrap code into an sql statement

**bcp**

-n is fine if the O/S is the same as the database (native format)

-c is character format, presumable for windows to unix transfers

Warning: Do not use row terminator (-r) or field terminator (-t) parameters with bcp in native format. Results are unpredictable and data may be corrupted.

bcp repo..VolatilityMatrix out vm -n -SDSARTLDD03 -Ureporoot -Paardvark –m0

bcp uat..VolatilityMatrix in vm -n -SDSARTLDT03 -Uuatroot –Paardvark –m0

or use a format file

bcp systr..sys\_pros in sys\_pros -U$USERNAME -P$PASSWORD -S$DSQUERY

-f$format\_file -N -b 10000 –m0

\*you have to insert an empty field in place of the identity column

dates - a space will bcp as Jan 1, 1900, no space will bcp as null

index necessary for slow bcp

no indexes or trigers, no logging = fast bcp

you can bcp a subset of rows thru a view..

-f format file, you can create this via a bcp out, or manually

-b #of rows to transfer at a time

-m maximum# of errors

-t to specify a delimiter (default is tab)

-n to not prompt for input, data will be sent in native format

-L last row (or # of rows) to copy

-F first row to copy

-E recognize the IDENTITY column

-e error file that bcp will store rows that need to be copied

-c bcp should send/receive data in characters only, and is tab delimited and \n is the end of line character

(this is the easy way)

given a field of numeric(5,3)

2230 bcps as 2230.000 2.23 bcps as 2.230

all whole#s bcp as a whole #

1111.40 bcps as 1111.400

1.1234 and 123456.1 will fail

decimal(5,3) does the same

**user stored procedures**

-once optomized, the query plan stays in “cache”

-you can GRANT and REVOKE access to a stored procedure

-- users should have no authority to tables, only procedures

when a proc is created, it is syntax checked and parsed into sysprocedures

when it is executed 1st time, it is optomized based on passed parameters, the query plan is stored in cache

CREATE PROCEDURE nt\_Dir

@path varchar(40) = 'C:\whatever',

@myflag char(1) = 'Y',

@a int =NULL

AS

WHILE THIS = TRUE BEGIN ---- CONTINUE ---- BREAK END

go

- pass parameters in correct order or like a hash, i.e.

exec @retst = sp1 parm1 = @val1 parm3 = @val3

- to pass multiple values between procedures (specify output on both ends)

exec t3 @result1 = @ret1 output, @result2 = @ret2 output

and...

create proc t3 @result1 char(20) output, @result2 char(20) output

passing an variable length array into a stored procedure

- you can break it apart like a hash using the charindex(value, string) command

side note : you can tally grades with charindex

select StudentID, 'A+'=sum(1\*charindex('A+',Grade), 'A'=sum(1\*charindex('A',Grade),

'B'=sum(1\*charindex('B',Grade) from TestGrades group by StudentID

a procedure is compiled on its 1st execution, not when its created

to delete old query plans.....

-restart server, or drop proc, then create proc

to archive old query plans

-create proc WITH RECOMPILE

-exec myproc WITH RECOMPILE

-UPDATE STATISTICS mytable (recreates all procs for mytable)

-sp\_recompile myproc

**misc**

PRINT "The trader is %1! and the spcd is %2!", @trdr, @spcd

RAISERROR (17216, 'my text', @localvar)

- akin to throwing an exception, (sets @@error)

-error #s 17000-19999 come from master..sysmessages

-error #s 20000+ come from localdb..sysmessages

if (@from\_date is null) or (@to\_date is null)

raiserror 99999 "Stored Proc GetFBB\_MonthlyFees requires two valid dates"

in dbi,

$gl\_dbh->err() will now equal 99999

$gl\_dbh->errstr will equal the msg

RAISERROR ('error %1! at %2! seconds',18,1, @sql\_error, %time) ERROR\_DATA

%1!, %2!, etc. used to interpolate varaibles -must be %x!

tempdb -recreated when the server is bounced

modeldb -the template for all new db's including tempdb

temp tables destroyed when job finishes, tempdb tables destroyed when server bounced

man sybperl perldoc SYBASE::SIMPLE

smalldatetime -keeps just the date, hour and minute, takes less disk space

float -precision of 15 real -prec. of 7 money-8 bytes smallmoney -4

other types are numceric (same as numeric), precision, char, varchar, int, etc..

-- or /\* \*/ -- comments

@@ ## -global field and temp obj

@ # -local

ISQL -S, -U, -P server, user, password

-h # of recs to print before headings

-b gets rid of the all-dashes header

-i input file, can also use <

-o output file

exec @sql will work for creating dynamic sqls on 12.0 and higher

select \* ... in a proc is translated to the actual fields, recompiling will not pick up the new fields, must recreate…

typical showplan output...look for indexes used, table scans, # of steps taken, work tables used, locking plan, i.e. allpages

users and user groups are database wide, roles and logins are server wide

Database design - aka Data Modeling, goals reduce IO and reduce duplicated data

- from thinking about it to actual DDL is reffered to logical shcema/physical schema

3rd Normalized form says to the effect that every "function" (i.e. customer, broker, trade, etc.) has its own table, its own primary key, and "trivial" data such as broker adress would only be in the broker table, not, say the trade table