
Advanced SPL
Paul Watson, Oninit
paul@oninit.com
B08 Tuesday - 10:50-11:50
2008 IIUG Informix Conference
www.iiug.org

Agenda

- No such thing as Advanced SPL
- What is SPL
- Free Datablades to make life easy
- Gotchas
- 'Fixing' sysprocedure or sysprocbody tables

Rules

- If in doubt ASK
- If you think I'm wrong tell me

If you KNOW I'm wrong then definitely tell me

What is SPL

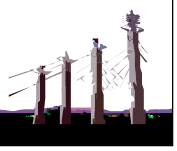
An SPL routine is a user-defined routine written in IBM Informix Stored Procedure Language (SPL). IBM Informix SPL is an extension to SQL that provides flow control, such as looping and branching. Anyone who has the Resource privilege on a database can create an SPL routine.

Routines written in SQL are parsed, optimized as far as possible, and then stored in the system catalog tables in executable format. An SPL routine might be a good choice for SQL-intensive tasks. SPL routines can execute routines written in C or other external languages, and external routines can execute SPL routines.

Advantages of SPL

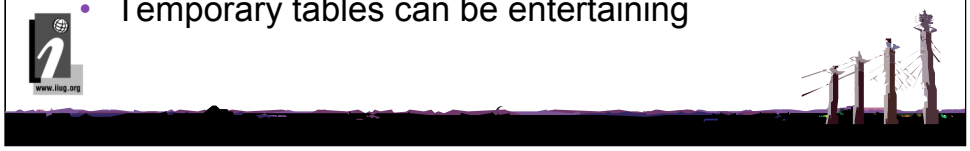
- DB Logic Abstraction
- Ease of Maintenance
- Sometimes quicker than raw SQL

Stops those dumb programmers writing
crap SQL
and hurting OUR databases



Disadvantages of SPL

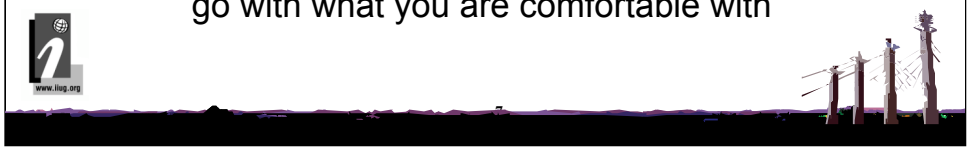
- Overhead on simple queries
- Can't do Dynamic in SPL prior to V9 – the ExecIT Bladelet is your friend.
- Unload doesn't work
- Temporary tables can be entertaining



Function or Procedure

- Procedures don't return data
- Functions must return data

Who cares – you choose,
go with what you are comfortable with



Error Handling

- ON EXCEPTION SET sqlerr, isamerr
  - Unique per Begin/End block

```
INSERT INTO t_error('spname', sqlerr,isamerr,CURRENT);
RETURN  sqlerr, (etc)
```

- RAISE EXCEPTION

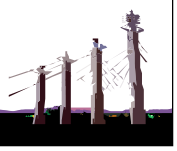
Version Control

- Not built in
- Example using sccs

```
DEFINE sccs  CHAR(255);
LET sccs = “%A%”;
```

Debugging

- SET TRACE FILE TO “xxxxx” WITH APPEND
- TRACE ON
- TRACE <variable>
- TRACE “How Did I get here”
- Significant performance impact



Performance

- Set optimisation high when creating
- Don’t forget Update Statistics if you want to avoid –’sysprocplan lock’ errors
- Check the Query Plan(s)

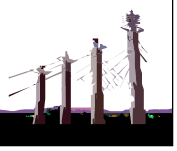
Recursion

- Requires recursion to explode children
- Careful: recurse too deep and you'll run out of stack.
- Nested-sets

Random Numbers

Random Numbers - Functions

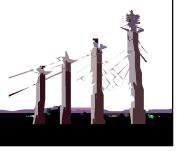
- udr_srand(seed) sets the seed.
- udr_get_randmax() returns whatever RAND_MAX is on your system.
- udr_rand() returns an integer between 0 and RAND_MAX.
- udr_rand_n_m(n, n) returns an integer between n and m.



Random Numbers – The Code

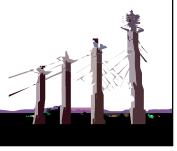
```
#include <stdlib.h>
#include "mi.h"

void udr_srand(mi_integer seed, MI_FPARAM *fparam)
{
    (void) fparam; srand(seed);
}

mi_integer  udr_rand(MI_FPARAM *fparam)
{
    (void) fparam;
    return rand();
}
```

Cont ….



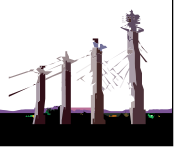
Random Numbers – The Code

```
mi_integer udr_get_randmax(MI_FPARAM *fparam)
{
    (void) fparam;
return RAND_MAX;
}

mi_integer udr_rand_n_m(mi_integer n, mi_integer m, MI_FPARAM *fparam)
{
    int range;
    (void) fparam;

    if ( n == m ) return n;
    if ( n < m ) range = 1 + m - n;
    else { range = 1 + n - m; n = m; }
    return n + (int) ((double)rand() / ((double)RAND_MAX + 1) * range);
}
```



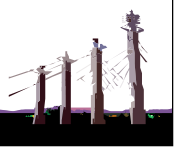
Random Numbers - Makefile

```
default : udr_rand.bld

CC = gcc
CFLAGS = -W -Wall -ansi -pedantic -O2
CBLDINCLS = -I${INFORMIXDIR}/incl/public -I${INFORMIXDIR}/incl/esql -
    I${INFORMIXDIR}/incl
CBLDFLAGS = -DMI_SERVBUILD -fPIC -shared
LDBLDFLAGS = -G #linux needs:
LDBLDFLAGS = -shared -u _etext RM = rm -f .SUFFIXES: .SUFFIXES: .o .c
#linux needs: LDBLDFLAGS = -shared -u _etext

Cont.…
```



Random Numbers - Makefile

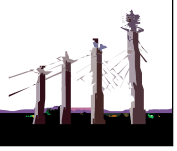
```
.SUFFIXES:
.SUFFIXES: .o .c

.c.o:
    ${CC} ${CFLAGS} ${CBLDFLAGS} ${CBLDINCLS} -c $< FILES.c = udr_rand.c
FILES.o = ${FILES.c:.c=.o}

udr_rand.bld:
    ${FILES.o} ${LD} ${LDBLDFLAGS} -o $@ ${FILES.o}

clean:
    ${RM} ${FILES.o} udr_rand.bld
```



Random Numbers - Register

```
create function udr_get_randmax()
    returns integer
    with (class="rnd") external name
    "$INFORMIXDIR/extend/rnd/udr_rand.bld(udr_get_randmax)"
language c;
```

```
VPCLASS
    VPCLASS=rnd
```



Bit Manipulation

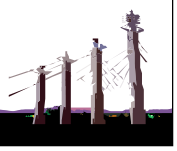
- Sysmaster has Bit manipulation as SPL
- But SPL is a bit slow

Bit Manipulation - Functions

bit_on(mi_integer item, mi_integer mask)
bit_pos_on(mi_integer item, mi_integer bitpos)
bit_or(mi_integer left, mi_integer right)
bit_and(mi_integer left, mi_integer right)
bit_xor(mi_integer left, mi_integer right)
bit_not(mi_integer value)
bit_shiftl(mi_integer left, mi_integer right)
bit_shiftr(mi_integer left, mi_integer right)

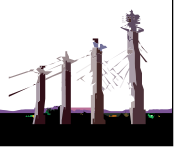
Bit Manipulation - Performance

```
# Check C function speed
    select count(*) from checks
    where bit_pos_on(tablemaskid, 15)
Time: 7.876 seconds

# Check speed of SYSMASTER:bitval
    select count(*) from checks
    where bitval(tablemaskid, 16384) = 1
Time: 12.578 seconds
```



Bit Manipulation - Performance

```
# Check speed of C (various of bit_on)
    select count(*) from checks
    where bit_and(tablemaskid, 16384) = 16384
Time: 7.844 seconds

# Check the speed of SYSMASTER:bitval
    select count(*) from checks
    where SYSMASTER:bitval(tablemaskid,16384) = 1
Time: 19.850 seconds
```



Regular Expressions

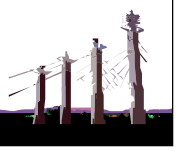
A **regular expression** is a sequence of literal characters and of metacharacters that lets you perform complex text search and modification. The UNIX grep, egrep, awk, sed utilities are examples that use regular expressions to manipulate text data.

For a comprehensive description and history of regular expressions, see *Mastering Regular Expressions* by Jeffrey E. F. Friedl (O'Reilly & Associates, Inc., 1997).

Regular Expressions

RegExp Metacharacter	
^	Beginning of line
$	End of line
|	Or
[abc]	Match any character in list
[^abc]	Match any character not in list
[a-z]	Match a range
.	Any Single Character
()	Regular Expression

Cont …



Regular Expressions

?	Match zero or one of the preceding expression.
*	Match zero, one, or many of the preceding expression
+	Match one or many of the preceding expression.
\	Match one or many of the preceding expression.
&	Reference the entire matched text for string substitution
\n	Reference subgroup *n* within the matched text (*n* can be 1-9).

Regular Expressions - Install

uncompress regexp.1.0.tar

tar xvf regexp.1.0.tar

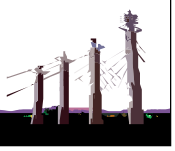
export TARGET=$INFORMIXDIR/incl/dbdk/makeinc.solaris

make -f sol-gcc.mak ← From the README

mkdir $INFORMIXDIR/extend/regexp.1.0
cp ./scripts/* $INFORMIXDIR/extend/regexp.1.0
cp ./src/solaris sparc/regexp.bld $INFORMIXDIR/extend/regexp.1.0

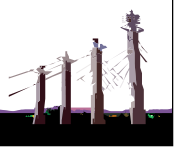
VPCLASS regexp,noyield

blademgr
shm> register regexp.1.0 demodb



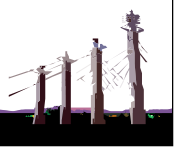
Regular Expressions - Functions

regexp_exract	Return a list of strings that match a regular expression from the source string.
regexp_match	Return TRUE if a source string matches the regular expression.
regexp_replace	Match a regular expression in a string and replace it with something else.
regexp_split	Splits a string into substrings, using the regular expression as the delimiter.
TraceSet_regexp	Enable tracing for **regexp** routines.



sp_is_date

```
create procedure "dvp".sp_is_date ( in_date char(10) ) returning smallint;

        define l_date date;

        on exception
                return 0;
        end exception

       if ( in_date = "" or in_date = " " or in_date is null ) then
          return 0;
       end if;

        let l_date = in_date;

        return 1;

end procedure
```



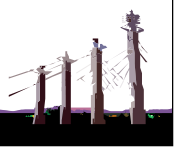
sp_is_date [cont]

```
select myfield,
    case
        when sp_is_date(uinqgroup) = 0 then 'NORMAL'
        when DATE(uinqgroup) < TODAY then 'PAST'
        when DATE(uinqgroup) = TODAY then 'PRESENT'
        else 'FUTURE'
    end case
from usmaster
where userid = ?
```

Moving Procedure Tables

- Why ?
  - Forgetting to resize the next extent size when the database was created
  - Too many extents

THIS IS NOT SUPPORTED



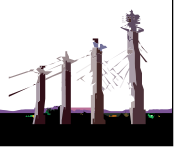
Moving Sysprocbody

```
create table newprocbody
(
        procid integer,
        datakey char(1),
        seqno integer,
        data char(256)
) in dat003dbs
extent size 256 next size 256
lock mode row;

insert into newprocbody
        select *
        from sysprocbody;

create unique index procbnew
on newprocbody(procid, datakey, seqno)
in table;
```

Note: the length of the name of the new table should be the same as original table

Moving Sysprocbody

```
select hex(partnum) partnum, rowid, tabname
from systables
where tabname in ("sysprocbody", "newprocbody")

partnum  0x0110000B <-- the partnums are 'interesting'
rowid    1025       <-- but, it's the rowids you need
tabname  sysprocbody

partnum  0x00E00121
rowid    2055
tabname  newprocbody
```



Moving Sysprocbody

```
oncheck -pp testme:systables 1025
addr              stamp    chksum nslots flag type         frptr frcnt next
14:3724           23105620 9fb6   16     1    DATA         1590  390   0
        slot ptr   len   flg
        1    24    98    0      <-- this is the slot for sysprocbody
        2    122   98    0
        3    220   98    0
        4    318   95    0
        5    413   97    0
        6    510   98    0
        7    608   98    0
        8    706   97    0
        9    803   99    0
        10   902   98    0
        11   1000  100   0
        12   1100  98    0
        13   1198  98    0
        14   1296  98    0
        15   1394  99    0
        16   1493  97    0
slot   1:
    0:  b 73 79 73 70 72 6f 63 62 6f 64 79 69 6e 66 6f  .sysprocbodyinfo
   16: 72 6d 69 78 20 20 20 20 20 20 20 20 20 20 20 20  <snip>
```

Moving Sysprocbody

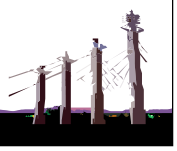
```
oncheck -pp testme:systables 2055
addr              stamp    chksum nslots flag type         frptr frcnt next     prev
14:4199           23157050 4832   7      2001 DATA         699   1317  0         0
        slot ptr   len   flg
        1    24    104   0
        2    128   95    0
        3    223   97    0
        4    320   97    0
        5    417   92    0
        6    509   92    0
        7    601   98    0    <-- this is the slot for newprocbody
[snip]
slot   7:
    0:  b 6e 65 77 70 72 6f 63 62 6f 64 79 69 6e 66 6f  .newprocbodyinfo
   16: 72 6d 69 78 20 20 20 20 20 20 20 20 20 20 20 20   [snip]
```

Moving Sysprocbody

```
onstat -d
[snip]
da9b328  14    14   0    10000  4591  PO--  /ix/IDS940/dbrootdbs
[snip]

informix@olorin :onmode -yuck

Soooo...

3724 is the page for sysprocbody and 24 is its offset,
4199 is the page for newprocbody and 601 is its offset,
/ix/IDS940/dbrootdbs is chunk 14

./swap_table.ksh 3724 24 4199 601 /ix/IDS940/dbrootdbs
```

Gotchas

- File permissions on trace output
- Forgetting to use Trace On
- Leaving Trace On
- Trailing ";" on Foreach statements
- Name overloading
- Update statistics - PDQ
- Define xxx LIKE table.xxx
- CURRENT isn't



Gotchas [cont]

- Temporary Tables are per session, not per procedure. This can be useful, especially with ACE reports.
- Temporary Tables are per session, not per procedure. This can be very unhelpful for Web Apps
- SYSTEM: -668 doesn't necessarily mean 'system command could not be executed'. Use a shell wrapper Env wrong, no home dir, no shell, run but exit with an error.
- Beware of transactions boundaries
- Remember its just SQL – stats, tables, indexes
- Casting can be fun

Cheetah V11

- onspaces, and onparams can be called via SPL functions

Questions ???

Session:
Advanced SPL

Paul Watson

Moving Sysprocbody

```
#!/bin/ksh

let PAGE_A=$1
let OFF_A=$2
let PAGE_B=$3
let OFF_B=$4
CHUNK=$5

dd if=$CHUNK of=p${PAGE_A} bs=2k skip=$PAGE_A count=1

if [ $PAGE_A -ne $PAGE_B ]
then
        dd if=$CHUNK of=p${PAGE_B} bs=2k skip=$PAGE_B count=1
fi
let PARTNUM_A=${OFF_A}+44
let REST_A=${OFF_A}+52

dd if=p${PAGE_A} of=partnum_a bs=1 skip=$PARTNUM_A count=4
dd if=p${PAGE_A} of=rest_a bs=1 skip=$REST_A count=46

let PARTNUM_B=${OFF_B}+44
let REST_B=${OFF_B}+52

dd if=p${PAGE_B} of=partnum_b bs=1 skip=$PARTNUM_B count=4
dd if=p${PAGE_B} of=rest_b bs=1 skip=$REST_B count=46

dd if=partnum_b of=p${PAGE_A} bs=1 seek=$PARTNUM_A conv=notrunc
dd if=rest_b of=p${PAGE_A} bs=1 seek=$REST_A conv=notrunc

dd if=partnum_a of=p${PAGE_B} bs=1 seek=$PARTNUM_B conv=notrunc
dd if=rest_a of=p${PAGE_B} bs=1 seek=$REST_B conv=notrunc
dd if=p${PAGE_A} of=$CHUNK bs=2k seek=$PAGE_A conv=notrunc
```