Cool SQL
aka SQL Lightning Talks

Darryl Priest
Advanced DataTools Corporation

B10
Tuesday, April 29, 2008 • 02:10 p.m. – 03:00 p.m.

2008 IIUG Informix Conference

www.iiug.org

Session B10
Cool SQL

Darryl Priest
Advanced DataTools Corporation
darryl@advancedatatools.com

How It Is Supposed To Work

- This was a session where other people were supposed to talk about their cool/brilliant/wacky SQL
- I was supposed to be the facilitator
- Based on Open Source Conference's Lightning Talks
- ~ 5 minutes each
- Wow your friends and neighbors

Sorting By Case Evaluation

Sorting By Case Evaluation

```
select linvoice, ltradat,
       case
            when llcode matches "U*" then "U"
            when llcode matches "*" and lcdebcr = "D" then "F"
            else "P"
       end transtype,
       sum(lamount)
from ledger, ledcode
where ledger.llcode = ledcode.lccode and
      lzero  <> 'R'
group by 1,2,3
order by 3,2
```

Pesky NULL vs. “ “

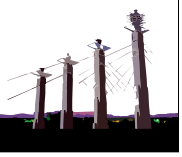
Courtesy of Mike Walker

Finding Those Pesky Nulls & Blanks

```
select "POPULATED" country_name, count(*)
from customer_address
where country_name is not null and country_name != " "
group by 1;

select "BLANK" country_name, count(*)
from customer_address
where country_name = " "
group by 1;

select "NULL" country_name, count(*)
from customer_address
where country_name is null
group by 1;

country_name       (count(*))
POPULATED                1992

country_name       (count(*))
BLANK                      89

country_name       (count(*))
NULL                        7
```

Finding Those Pesky Nulls & Blanks

```
select
   case
      when country_name is null then "NULL"
      when country_name = " "   then "BLANK"
      else "POPULATED"
   end country_name,
   count(*)
from customer_address
group by 1;


country_name       (count(*))

POPULATED                1992
BLANK                      89
NULL                        7
```



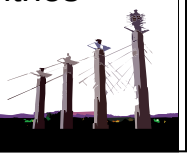
Client Descriptions
Number of Entries & Sizes

Entries & Size Of Client Descriptions

- Given a “normalized” description table

```
create table clidesc
  (
    cdindex serial not null ,
    clnum char(14),
    cdline smallint,
    cddesc char(48)
  );
```

- Want to know number of rows per client
- Also interested in size of client description entries



Entries & Size Of Client Descriptions

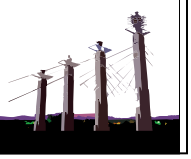
- Build Summary Temporary Table

```
select clnum, count(*) cnt, sum(nvl(length(cddesc),0)) chars
from clidesc
group by 1
into temp t1;
```

Entries & Size Of Client Descriptions

- Get Number Of Rows Per Client

```
select case
     when cnt <= 1                then '000-001 rows'
     when cnt >  1 and cnt <=  10 then '002-010 rows'
     when cnt > 10 and cnt <=  25 then '011-025 rows'
     when cnt > 25 and cnt <=  50 then '026-050 rows'
     when cnt > 50 and cnt <=  75 then '051-075 rows'
     when cnt > 75 and cnt <= 100 then '076-100 rows'
     else '100-xxx rows'
       end case,
       count(*)
from t1
group by 1
order by 1;
```



Entries & Size Of Client Descriptions

- Output Of Rows Per Client

```
case              (count(*))

000-001 rows           10808
002-010 rows            7448
011-025 rows             675
026-050 rows             113
051-075 rows              16
076-100 rows               6

6 row(s) retrieved.
```

Entries & Size Of Client Descriptions

- Get Sizes Of Client Description Entries

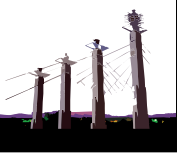
```
select case
     when chars <=   1                  then '0000-0001 chars'
     when chars >    1 and chars <=   25 then '0002-0025 chars'
     when chars >   25 and chars <=   50 then '0026-0050 chars'
     when chars >   50 and chars <=   75 then '0051-0075 chars'
     when chars >   75 and chars <=  100 then '0076-0100 chars'
     when chars >  100 and chars <=  250 then '0101-0250 chars'
     when chars >  250 and chars <=  500 then '0251-0500 chars'
     when chars >  500 and chars <=  750 then '0501-0750 chars'
     when chars >  750 and chars <= 1000 then '0750-1000 chars'
     when chars > 1000 and chars <= 1500 then '1001-1500 chars'
     when chars > 1500 and chars <= 2000 then '1501-2000 chars'
     when chars > 2000 and chars <= 3000 then '2001-3000 chars'
     when chars > 3000 and chars <= 4000 then '3001-4000 chars'
     when chars > 4000 and chars <= 5000 then '4001-5000 chars'
     else                                     '5000-xxxx chars'
       end case,
       count(*)
from t1
group by 1
order by 1;
```

Entries & Size Of Client Descriptions

- Output Of Client Description Sizes SQL

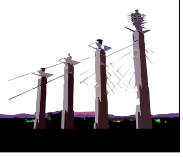
```
case                  (count(*))

0000-0001 chars             2293
0002-0025 chars             4239
0026-0050 chars             4624
0051-0075 chars             1735
0076-0100 chars             1707
0101-0250 chars             3063
0251-0500 chars              960
0501-0750 chars              246
0750-1000 chars              103
1001-1500 chars               63
1501-2000 chars               18
2001-3000 chars               11
3001-4000 chars                4

13 row(s) retrieved.
```

“Count” Ethnicities In PeopleSoft



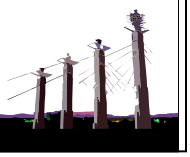
"Count" Ethnicities In PeopleSoft

- PeopleSoft table `PS_DIVERS_ETHNIC`
  - Can have multiple rows for each employee
  - Person may have not reported ethnicity
- HR wanted a report that showed employees as either:
  - More than 1 ethnicity
  - Declined to report ethnicity
  - Reported a single ethnicity



"Count" Ethnicities In PeopleSoft

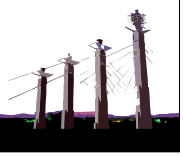
```
SELECT A.EMPLID,
       CASE
          WHEN COUNT(B.ETHNIC_GRP_CD) > 1          THEN MIN('Two or More Races')
          WHEN MIN(nvl(B.ETHNIC_GRP_CD,'x')) = 'x' THEN MIN('NOT ENTERED')
          ELSE MIN(C.DESCR50)
       END CASE
FROM PS_PERSON A
     LEFT OUTER JOIN PS_DIVERS_ETHNIC B ON
        A.EMPLID = B.EMPLID AND
        B.REG_REGION = 'USA'
     LEFT OUTER JOIN  PS_ETHNIC_GRP_TBL C ON
        B.ETHNIC_GRP_CD = C.ETHNIC_GRP_CD AND
        C.EFFDT = (SELECT MAX(C_ED.EFFDT)
                   FROM PS_ETHNIC_GRP_TBL C_ED
                   WHERE C.SETID = C_ED.SETID AND
                         C.ETHNIC_GRP_CD = C_ED.ETHNIC_GRP_CD AND
                         C_ED.EFFDT <= TODAY)
GROUP BY A.EMPLID
```



Validation Of Dates

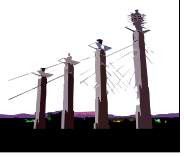
Validation Of Dates

- When dates are stored in non-date fields or entered by users it can be difficult to be sure they are valid dates.
- Wanted an easy way to verify date field format as well as actual date existence.
- Wanted to be able to access in SQL or SPL.
- Desired a boolean return value for readability, specifically of SPL.



Validation Of Dates

```
create procedure sp_is_date ( in_val char(40) ) returning boolean;

define date_check date;

on exception
   return 'f';
end exception;

if ( in_val = '' or in_val = ' ' or in_val is null ) then
   return 'f';
end if;

let date_check = in_val;

return 't';

end procedure;
```



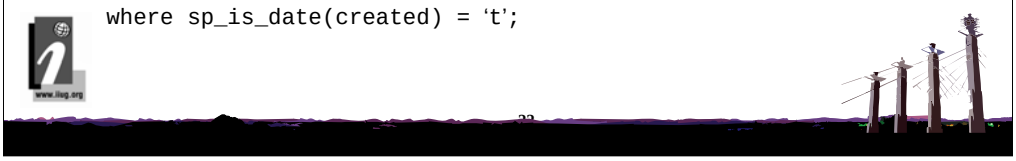
Validation Of Dates

- Utilize Via SPL

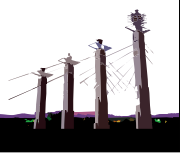
```
create procedure sp_test( in_val char(40) );

if ( sp_is_integer( in_val ) ) then
   .. .. Do something important with the data .. ..
else
   .. .. Return error .. ..
end if;

end procedure;
```

- Or In SQL

```
select tabid, tabname
from systables
where sp_is_date(created) = 't';
```

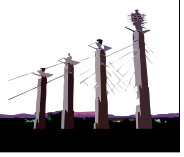


"Dynamic" SQL In SPL



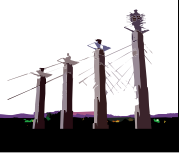
“Dynamic” SQL In SPL

```
create temp table t1_batches ( batch integer );

if in_batch = 0 then
   insert into t1_batches
   select unique batch_daily_load_number
   from faads_ffata_load_history
   where faads_ffata_processing_step_code = 'L';
else
   insert into t1_batches values ( in_batch );
end if

create index t1_batch_idx1 on t1_batches(batch);

update statistics high for table t1_batches;
```

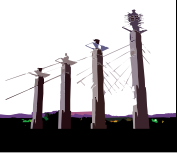


“Dynamic” SQL In SPL

```
select h.faads_ffata_identifier,
       h.entity_zip_code,
       lh.data_source_acronym
into l_faads_ffata_identifier,
     l_entity_zip_code,
     l_data_source_acronym
from faads_ffata_history h,
     faads_ffata_load_history lh
where h.batch_daily_load_number = lh.batch_daily_load_number and
      h.batch_daily_load_number in (select batch from t1_batches)
```

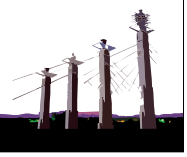


Changing Database Names In SPL



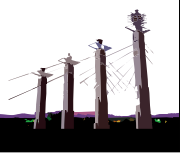
Changing Database Names In SPL

```
removed_by_build_shell_script
create procedure sp_mfo_phase1_validate( source_system char(5),
                                         batch_number integer
) returning char(150)

foreach select internal_id,
               cfda_number,
               sai_number,
               duns_number
        into l_internal_id,
             l_cfda_number,
             l_sai_number,
             l_duns_number
        from <LOAD_DB>:faads_ffata_load
        order by 1
… Inmportant SPL Commands …
end foreach

removed_by_build_shell_script
grant  execute on function sp_mfo_phase1_validate (char,integer)
   to “userx" as "dba_1";
```



Changing Database Names In SPL

```
## Need to replace database names in the stored procedure scripts
cat ${SQL_FILE} | sed -e "/^removed_by_build_shell_script$/d" |
                  sed -e "s/<LOAD_DB>/${LOAD_DB}/g" |
                  sed -e "s/<PROD_DB>/${PROD_DB}/g" |
                  sed -e "s/<MRT_DB>:/${MRT_DB}:mbldba./g" > ${TEMP_SQL}

dbaccess ${PROD_DB} ${TEMP_SQL}
```

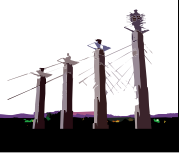


Trapping SQL Errors In Shells

Courtesy of Paul Ruotolo

Trapping SQL Errors In Shells

```
function run_sql
{
   RUN_DB=$1
   RUN_SQL=$2
   IGNORE_ERRORS=$3

   TMP="/tmp/$$"
   rm -f ${TMP}.sql_out

   dbaccess -e ${RUN_DB} ${RUN_SQL} > ${TMP}.sql_out 2>&1
   RET=$?

   if [ $RET != 0 ] && [ $IGNORE_ERRORS = "N" ]
   then
      echo "\nERROR: Error occured while running SQL (${RUN_SQL})..."
      echo "ERROR: Transcript follows..."
      sed 's/^/ERROR:    /' ${TMP}.sql_out | while read ERRLINE
      do
         echo "$ERRLINE"
      done
   fi
   return $RET
}
```



Trapping SQL Errors In Shells

```
echo "drop procedure ${PROC_NAME};" > ${TEMP_SQL}

run_sql ${PROD_DB} ${TEMP_SQL} Y


## Need to replace database names in the stored procedure scripts
cat ${SQL_FILE} | sed -e "/^removed_by_build_shell_script$/d" |
                  sed -e "s/<LOAD_DB>/${LOAD_DB}/g" |
                  sed -e "s/<PROD_DB>/${PROD_DB}/g" |
                  sed -e "s/<MRT_DB>:/${MRT_DB}:mbldba./g" > ${TEMP_SQL}

run_sql ${PROD_DB} ${TEMP_SQL} N
RET=$?

if [ ${RET} != 0 ]
then
   echo "ERROR: Failed to build stored procedure"
   terminate ERR
fi
```

Pipes & dbaccess

Pipes & dbaccess

```
echo "select file_name from faads_ffata_load_history“ | dbaccess mfopd

echo "select file_name from faads_ffata_load_history“ | dbaccess mfopd 2>/dev/null


for file in `echo "select file_name from faads_ffata_load_history"|dbaccess mfopd 2>/dev/null`
do
more $file
done


echo "select * from faads_ffata_load_history“ | dbaccess mfopd | grep file
```

Guaranteeing One Row Returned

Courtesy of John Miller iii

Guaranteeing One Row Returned

- Previously

```
select CURRENT from systables where tabid = 1;
```

- Now You Can Select Without Guessing That You'll Get 1 Row From systables

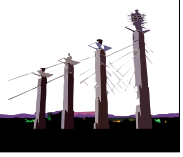
```
select CURRENT from table(set{1});
```

- Or

```
select CURRENT from dual;

From John Miller iii
```

Nifty Administration SQL In Cheetah

Courtesy of John Miller iii



Nifty Administration SQL In Cheetah

- This sql statement will read the last (or most recent) 1KB of the online.log file.
  - Need the skip 1 to remove the partial line.
  - Negative offsets mean backwards from the end of the file.

```
select skip 1 line
from sysmaster:sysonlinelog
where offset > -1024;
```

From John Miller iii

Nifty Administration SQL In Cheetah

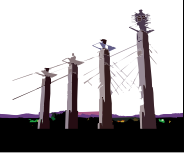
- Search for ‘errors; in the online.log

```
select *
from sysonlinelog
where line matches “*Assert*” or
      line matches “*Warn*” or
      line matches “*Error*”;
```

- Oncheck all tables in a specfic dbspace.

```
select task("check data", partnum) as id,
       trim(dbname) ||"."|| trim(tabname) as table
from sysmaster:systables
where trunc(partnum/1048575) = 1;
```

From John Miller iii



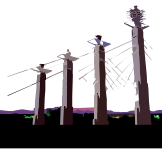
Nifty Administration SQL In Cheetah

- Rolling out a complete system. You put the different dbspaces in the dbspace table and the chunk in the chunk table and the following will unfold the system.

```
database sysadmin;

{ **** Create a table of dbspaces which are to be created ****}
create table dbspaces (
type    varchar(255),
dbspace varchar(255),
path    varchar(255),
offset  varchar(255),
size    varchar(255) );

insert into dbspaces values
("sbspace", "sbspace", "$INFORMIXDIR/CHUNKS/sblob1", 0 , "50 MB" );
insert into dbspaces values
("dbspace", "dbspace1", "$INFORMIXDIR/CHUNKS/dbspace1", 0 , "50 MB" );
insert into dbspaces values
("dbspace", "dbspace2", "$INFORMIXDIR/CHUNKS/dbspace2", 0 , "50 MB" );
insert into dbspaces values
("dbspace", "physdbs", "$INFORMIXDIR/CHUNKS/physdbs", 0 , "50 MB" );
insert into dbspaces values
("dbspace", "logdbs", "$INFORMIXDIR/CHUNKS/logdbs", 0 , "50 MB" );
insert into dbspaces values
("tempdbspace", "tempdbs", "$INFORMIXDIR/CHUNKS/tempdbs", 0 , "10 MB" );
insert into dbspaces values
("blobspace", "bspace1", "$INFORMIXDIR/CHUNKS/blobdbs", 0 , "50 MB" );
```

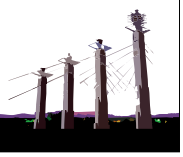
Nifty Administration SQL In Cheetah

```
{ **** Create a table of chunks which are to be created **** }
create table chunks (
dbspace varchar(255),
path    varchar(255),
offset  varchar(255),
size    varchar(255)  );

insert into chunks values
("dbspace1", "$INFORMIXDIR/CHUNKS/chunk",0 , "10 MB" );
insert into chunks values
("dbspace1", "$INFORMIXDIR/CHUNKS/chunk2",0 , "10 MB" );

{**** Create all the dbspaces ****}
select task( "create "|| type , dbspace, path, size, offset)
from dbspaces;

{**** Add the chunks to the dbspaces ****}
select task("add chunk", dbspace, path, size, offset)
from chunks;

{**** Add 3 logical logs ****}
execute function task("add log","logdbs","5 MB",3,"true");
execute function task("checkpoint");
```



Nifty Administration SQL In Cheetah

```
{**** Drop all logical logs in the rootdbs but the current log ****}
select task("drop log", number)
from sysmaster:syslogfil
where chunk = 1 and
     sysmaster:bitval(flags,"0x02") == 0;

execute function task("checkpoint");

select task("onmode", "l")
from sysmaster:syslogfil
where chunk = 1 and
       sysmaster:bitval(flags,"0x02") > 0;

execute function task("checkpoint");

{**** Drop the current logical log in the rootdbs ****}
select task("drop log", number)
from sysmaster:syslogfil
where chunk = 1;

execute function task("alter plog","physdbs","49 MB");

execute function task("checkpoint");
```



Key Thoughts

- CASE
- Temporary Tables
- Pipe(s) To & From dbaccess

Session B10
Cool SQL

Darryl Priest
Advanced DataTools Corporation
darryl@advancedatatools.com