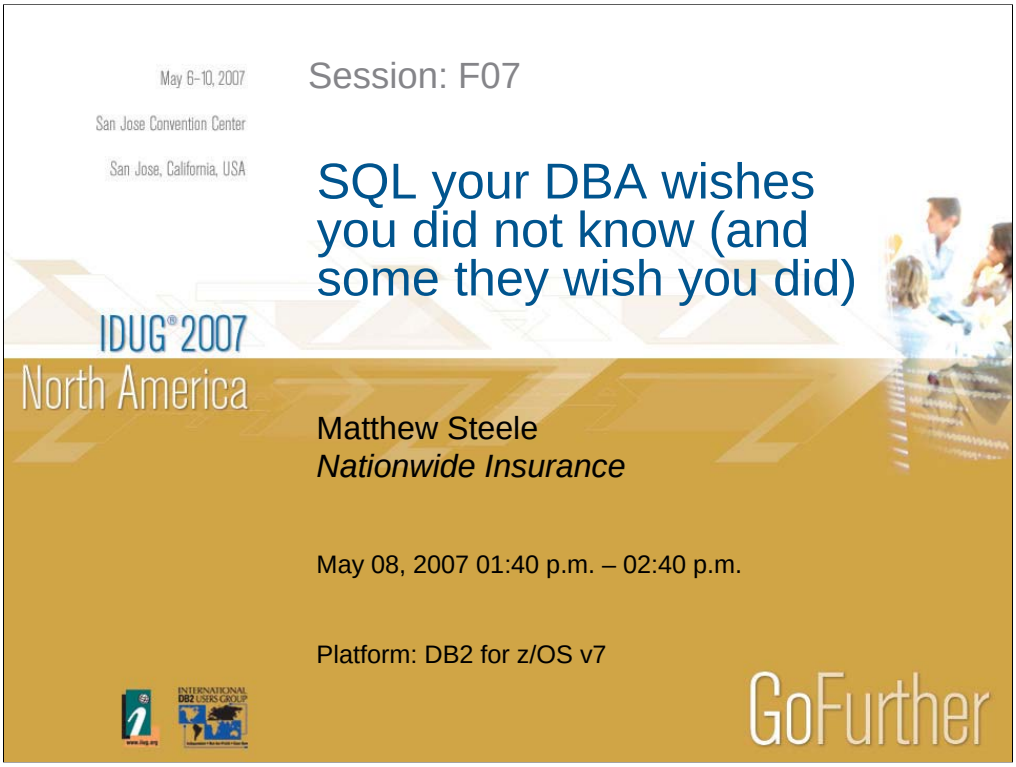


My goal is for everyone to come out with at least one SQL statement they can use that will make their life or the life of the DBA easier.

DB2 Organization at Nationwide

- Centralized DB2 DBAs
- DBAs are responsible for multiple Areas
- No one outside of the centralized DB2 DBA Team has access to administer DB2 Objects

My Area

- In Nationwide, we are a new, minor player that has a DBA that is responsible for multiple areas most bigger then we are we are
- We only have 6 scheduled production releases during the year and during production night that DBA has to handle all DB2 incidents for all his area
- Since users use our system 24 x 7, we have a very short implementation window (typical 12:30am – 2:00am)

Put it all together

- Our DBAs are very, very busy during implementation nights and everyday so whatever the application areas can do to help them, the better
- If we can find the issue before them, the better it is for everyone
- We try to keep our requests limited to object changes and use SQL and system tables whenever possible

No rights to administer objects in DB2 except in our development environment

I do not have access to administer objects or use DB2 utilities in the production environment

Things you need to run the SQL:

MOST IMPORTANT:

- Database Name (DBNAME, DSNAME)

Additional Items:

- RACF group you are associated with (GRANTEE)
- Name of Package Collection (COLLID)
- Schema Name

How to get the database name:

- You should know one of the table names:

```
select creator, dbname
from sysibm.systables
where name = 'THOMREVW'
```

WARNING:

- Most of these slides are only as good as the statistics in the catalog tables.

Section 1:

SQL to determine if DBAs are doing their job.

SQL you can run to hold your DBAs accountable.

RUNSTATS

- The utility in DB2 that updates the catalog statistics that is used by the optimizer to determine the fastest path to your data.

The strategy your company employs to determine when to run RUNSTATS is not something that I will be going over in this presentation. I will just point out how you can check to make sure RUNSTATS has been ran and when the last time it was ran.

Runstats

- Last time RUNSTATS was ran:

```
select creator, name, cardf, statstime
from sysibm.systables
where dbname = 'DHOMNR01'
and type = 'T'
order by name
```

Output

CREATOR	NAME	CARDF	STATSTIME
DBHOMNR	THOMADDR	3164778	2/11/2007 8:33
DBHOMNR	THOMAGNT	26369	2/11/2007 8:00
DBHOMNR	THOMAGRI	511	2/11/2007 8:34
DBHOMNR	THOMCLASS	12128	2/11/2007 8:34
DBHOMNR	THOMCMCT	20500576	2/11/2007 8:00
DBHOMNR	THOMCMTS	3956200	2/11/2007 8:07
DBHOMNR	THOMCOAC	13513	2/11/2007 8:34
DBHOMNR	THOMCONT	2264783	2/11/2007 8:08
DBHOMNR	THOMCRDT	-1	
DBHOMNR	THOMCTNB	2257048	2/11/2007 8:34
DBHOMNR	THOMCTPT	3625680	2/11/2007 8:32

Runstats Needed - Tables

- What tables have never had RUNSTATS ran:

```
select creator, name, cardf, statstime
from sysibm.systables
where dbname = 'DHOMNR01'
and type = 'T‘
and cardf = -1
order by name
```


Not recommended because you lose the important information of the last time RUNSTATS was ran. This just tells you what tables need RUNSTATS. Can be used if adding a lot of new tables and want to make sure they have been included in the RUNSTATS job or the database is new.

Cardf may also be -1 for a view, alias, or created temporary tablespace

By limiting checking for RUNSTATS on sysibm.systables what are we missing?

Runstats on Indexes

- Last time RUNSTATS was ran:

```
select idx.creator, idx.name, idx.tbname,
   keys.colname, keys.colseq, idx.statstime
from sysibm.sysindexes idx
, sysibm.syskeys keys
where idx.dbname = 'DHOMNR01'
and keys.ixname = idx.name
and keys.ixcreator = idx.creator
order by idx.name, keys.colseq
```

Output

CREATOR	NAME	TBNAME	COLNAME	COLSEQ	STATSTIME
DBHOMNR	XHOMCOAC	THOMCOAC	CO_ACCT_POL_NUM	1	2/11/2007 8:34
DBHOMNR	XHOMCOA2	THOMCOAC	HCR_ID	1	2/11/2007 8:34
DBHOMNR	XHOMCOA3	THOMCOAC	UNDERWRIT_ID	1	2/11/2007 8:34
DBHOMNR	XHOMCON3	THOMCONT	CT_PT_ID	1	2/11/2007 8:08
DBHOMNR	XHOMCRD0	THOMCRDT	CRDT_ID	1	
DBHOMNR	XHOMCRD1	THOMCRDT	ST_CD	1	
DBHOMNR	XHOMCTNB	THOMCTNB	CONT_NUM_ID	1	2/11/2007 8:34
DBHOMNR	XHOMCTN2	THOMCTNB	CT_PT_ID	1	2/11/2007 8:34

Runstats Needed - Indexes

- What indexes have never had RUNSTATS ran:

```
select idx.creator, idx.name, idx.tbname, keys.colname,
    keys.colseq, idx.statstime
from sysibm.sysindexes idx
, sysibm.syskeys keys
where idx.dbname = 'DHOMNR01'
and nleaf = -1
and keys.ixname = idx.name
and keys.ixcreator = idx.creator
order by idx.name, keys.colseq
```


Not recommended because you lose the important information of the last time RUNSTATS was ran. This just tells you what tables need RUNSTATS. Can be used if adding a lot of new indexes and want to make sure they have been included in the RUNSTATS job or the database is new.

Nleaf is the number of active leaf pages in the index.

REORGs

- The REORG utility reorganizes a table or index to improve access performance and to reclaim fragmented space.
- DB2's optimizer may not choose to use an index if the cluster ratio of the index is low. Instead, it may choose to do a table scan.

Important Note

- Some of the SQL statements are ONLY as good as the statistics they use.
- If the statistics are old, these queries will be reporting on OLD statistics

- The next section is on REORGs and the SQL on the next pages should only be used after statistics are in a desired state
  - Can use the previous slides to see the last time statistics was ran

This leads us to the next slides

A good example is the SQL used to mimic reorgchk. Since these SQL statements rely solely upon statistics, they will only be as good as the statistics at that time.

REORGs Needed

- Many people and organizations have scheduled REORG jobs
- Many other people and organizations have come up with threshold driven REORG drives
- There are numerous thresholds you can use and I am sure there are many differing opinions
- Your DBA probably already has a REORG strategy in place but if you want to check to see if you agree with his strategy you can run some of the following

xxxx.SDSNSAMP(DSNTESP)

- Dataset that has IBM Supplied Sample SQL to determine if a REORG needs to be done
- Can run Queries 15 – 21 to determine if a REORG is needed

DSNTESP – Query 15

- Returns a list of tablespaces that may need to be REORG'd
- Added DBNAME = 'DHOMNR01' to limit to only our area's tablespaces
- Added Additional Columns

```
SELECT TSNAME, CARDF, FARINDREF, NEARINDREF
, PERCDROP
, (NEARINDREF + FARINDREF) * 100 / CARD as CALCX
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'DHOMNR01'
AND ((CARD > 0 AND (NEARINDREF + FARINDREF) * 100 / CARD
    > 10)
OR PERCDROP  > 10)
```


Can run with only DBNAME = 'DHOMNR01' clause to see how close tables are to needing to be REORG'd

NEARINDREF + FARINDREF will give you the total number of rows that have been relocated from their original page. NEARINDREF is the number of rows that have been relocated near their original page. FARINDREF is the number of rows that have been relocated far from their original page.

PERCDROP is the percentage of space occupied by rows of dropped tables.

Output

TSNAME	CARDF	FARINDREF	NEARINDREF	PERCDROP	CALCX

Query 15 – Con't

- If you want to know how close some tablespaces are to needing to be reorg'd, you can run the following query:

```
SELECT TSNAME, CARD, FARINDREF
, NEARINDREF,PERCDROP
, (NEARINDREF + FARINDREF) * 100 / cast(CARD as
   double) as CALCX
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'DHOMNR01'
AND (CARD <> -1) AND (CARD <> 0)
ORDER BY TSNAME
```


I used CAST for the denominator so the result would be a double.

Output

TSNAME	CARD	FARINDREF	NEARINDREF	PERCDROP	CALCX
SHOMCMCT	20500576	268184	494	0	1.31058756592985
SHOMCOTP	12128	0	0	0	0
SHOMLCQU	1462	2	3	0	0.341997264021888
SHOMLCRV	372488	0	0	0	0
SHOMLKUP	557	12	30	0	7.54039497307002
SHOMMDRC	90	0	0	0	0
SHOMRVHZ	652432	0	0	0	0
SHOMRVQU	49	0	0	0	0
SHOMRVQU	21502345	0	0	0	0

DSNTESP – Query 16

- Returns a list of index spaces that may need to be REORG'd
- Added Join to SYSIBM.SYSINDEXES to be able to include DBNAME = 'DHOMNR01'
- Added Additional Columns

```
SELECT IXP.PARTITION, IXP.IXNAME, IX.TBNAME, IXP.CARDF
, IXP.LEAFDIST, IX.CLUSTERING
FROM SYSIBM.SYSINDEXPART IXP
INNER JOIN SYSIBM.SYSINDEXES IX
ON IX.NAME = IXP.IXNAME
WHERE DBNAME = 'DHOMNR01'
AND LEAFDIST > 200
ORDER BY IX.TBNAME, IXP.IXNAME,IXP.PARTITION
```


LEAFDIST = 100 * the average number of leaf pages between successive active leaf pages of the index. Helps determine the efficiency of the index.

The more intervening pages, the less efficient the index will be.

Output

PARTITION	IXNAME	TBNAME	CARDF	LEAFDIST	CLUSTERING
0	XNEWADDR	THOMADDR	3180871	283936	Y
0	XNEWADD2	THOMADDR	3180871	303805	N
0	XNEWADD3	THOMADDR	3180871	136286	N
0	XHOMCLASS	THOMCLASS	12621	602	Y
0	XHOMCMC2	THOMCMCT	20695873	13966	N
0	XHOMCMC3	THOMCMCT	20695873	15528	Y
0	XHOMCMC4	THOMCMCT	20695873	9889	N
0	XHOMCMT2	THOMCMTS	3984789	3895	Y
0	XHOMCMT3	THOMCMTS	3984789	202	N
0	XHOMCTNB	THOMCTNB	2268658	223263	Y
0	XHOMCTN2	THOMCTNB	2268658	171216	N

Reorg Indexes – DB2 Admin Guide

- The rule of thumb IBM gives on page 784 of the DB2 Admin Guide for when an index needs reorg'd
- LEAFFAR/NLEAF > 10%

```
SELECT IXP.IXNAME, IX.TBNAME, IXP.LEAFFAR, IX.NLEAF
, (IXP.LEAFFAR/cast(IX.NLEAF as double)) * 100 as perc
FROM SYSIBM.SYSINDEXPART IXP
INNER JOIN SYSIBM.SYSINDEXES IX
ON IX.NAME = IXP.IXNAME
WHERE IX.DBNAME = 'DHOMNR01'
and (ixp.leaffar <> 0) and (ixp.leaffar <> -1)
and (ix.nleaf <> 0) and (ix.nleaf <> -1)
and  (IXP.LEAFFAR/cast(IX.NLEAF as double) * 100) > 10
ORDER BY IX.TBNAME, IXP.IXNAME
```


LEAFFAR = Number of leaf pages located physically far away from previous leaf pages for successive active leaf pages accessed in an index scan.

NLEAF = Number of active leaf pages in the index

LEAFFAR/NLEAF = Percentage of pages in disorganization. Percentage of pages that are located far away from previous leaf pages.

Output

IXNAME	TBNAME	LEAFFAR	NLEAF	PERC
XNEWADDR	THOMADDR	10838	27054	40.0606195017373
XNEWADD2	THOMADDR	11620	29003	40.0648208805986
XNEWADD3	THOMADDR	2243	5228	42.9035960214231
XHOMCMC3	THOMCMCT	12767	125287	10.1902032932387
XHOMCTNB	THOMCTNB	7730	19347	39.9545149118726
XHOMCTN2	THOMCTNB	6475	15882	40.7694245057298
XHOMCTPT	THOMCTPT	12562	31076	40.4234779250869
XHOMCTP2	THOMCTPT	12986	21391	60.7077742976018

DSNTESP – Query 21

- Checks to see if LOB table space needs to be REORG'd
- ORGRATIO greater than 2 generally indicates a LOB table space that needs REORG'd
- Added DBNAME = 'DHOMNR01' to limit to our area

```
SELECT NAME, ORGRATIO
FROM SYSIBM.SYSLOBSTATS
WHERE DBNAME = 'DHOMNR01'
AND ORGRATIO > 2
```


ORGRATIO – Ratio of organization in the LOB tablespace. Value 1 indicating perfect organization. The greater the value exceeds 1, the greater the disorganization.

DSNTESP – Queries 17 - 20

- Can use Queries 17 -20 to monitor different useful statistics **over time** to see if a REORG is desired
- Look at documentation found in the sample dataset DSNTESP to see how to use the information found in these queries
- Numerous other thresholds that can be monitored with SQL to see if a REORG needs to be done
  - The internet has numerous examples of other statistics that can be used

Query 17 –

```
SELECT CARD, NEARINDREF, FARINDREF
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'xxx'
AND TSNAME = 'yyy'
```

Query 18 –

```
SELECT PERCDROP
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'xxx'
AND TSNAME = 'yyy'
```

Query 19 –

```
SELECT NEAROFFPOS, FAROFFPOS
FROM SYSIBM.SYSINDEXPART
WHERE IXCREATOR = 'zzz'
AND IXNAME = 'www'
```

Query 20 –

```
SELECT LEAFDIST
FROM SYSIBM.SYSINDEXPART
WHERE IXCREATOR = 'zzz'
AND IXNAME = 'www'
```

Status of a Table

- The CHECKFLAG column in the sysibm.systables table can tell you if a table is in a check pending status
- The STATUS column in the sysibm.systablespace table can tell you if a tablespace is in a check pending status

Table Status - Check Pending

- To find tables that are in a check pending status:

```
select creator, name, tsname, checks
from sysibm.systables
where dbname = 'DHOMNR01'
and checkflag = 'C'
```


One of the following conditions is true:

- There are rows in the table that violate referential constraints, table check constraints, or both
- The table is a materialized query table that might contain inconsistent data

Tablespace Status – Check Pending

- To find tablespaces in check pending status

```
select ts.name, ts.ntables, tab.name, ts.status, tab.checkflag
from sysibm.systablespace ts
inner join sysibm.systables tab
on tab.tsname = ts.name
where ts.dbname = 'DHOMNR01'
and ts.status in ('P', 'S')
order by ts.name, tab.name
```

33

P - if the ENTIRE tablespace is in a check pending status

S - if only a portion of the tablespace is in a check pending status

Recovery

- Are your DBAs taking full image copies on a regular basis?
- Image copies gives DB2 a starting place in case recovery needs to be done

Last time Image Copy was ran

Last time a full image copy was taken on a tablespace:

```
select copy.tsname, copy.dsnum, copy.dsname, copy.icbackup
, copy.timestamp
from sysibm.syscopy copy
where copy.dbname = 'DHOMNR01'
and (copy.tsname, copy.icbackup, copy.timestamp) in (
select tsname, icbackup, max(timestamp)
from sysibm.syscopy
where dbname = 'DHOMNR01'
and ictype = 'F' and stype = ''
group by tsname, icbackup
)
and copy.ictype = 'F' and copy.stype = ''
order by copy.tsname, copy.icbackup
```

35

ICTYPE = F, Full Image Copy

STYPE = blank, specifies a DB2 Image Copy if ICTYPE = F

ICBACKUP can be the following:

Blank – local site primary copy

LB – Local site Backup copy

RP – Recovery site primary copy

RB – Recovery site backup copy

Output

TSNAME	DSNUM	DSNAME	ICBACKUP	TIMESTAMP
SHOMADDR	0	DB2BKPG1.D07042.T1548.DHOMNR01.SHOMADD		2/11/2007 15:54
SHOMAGNT	0	DB2BKPG1.D07042.T1548.DHOMNR01.SHOMAGN		2/11/2007 16:04
SHOMAGNT	0	DRPDB2G1.DHOMNR01.SHOMAGNT.P00000.D04	RP	2/10/2007 2:09
SHOMAGRI	0	DB2BKPG1.D07042.T1548.DHOMNR01.SHOMAGR		2/11/2007 16:15
SHOMCMCT	0	DB2BKPG1.D07042.T1548.DHOMNR01.SHOMCMC		2/11/2007 16:05
SHOMCMTS	0	DB2BKPG1.D07042.T1548.DHOMNR01.SHOMCMT		2/11/2007 15:55
SHOMCOAC	0	DB2BKPG1.D07042.T1548.DHOMNR01.SHOMCOA		2/11/2007 16:13
SHOMCOTP	0	DB2BKPG1.D07042.T1548.DHOMNR01.SHOMCOT		2/11/2007 15:51
SHOMCOTP	0	DRPDB2G1.DHOMNR01.SHOMCOTP.P00000.D04	RP	2/10/2007 8:22

No image copy in x days

- If you want to make sure your DBA is taking an image copy at least once a week on a table:

```
select tab.name, tab.tsname, max(copy.timestamp) as
   timestamp
from sysibm.systables tab
left outer join sysibm.syscopy copy
on copy.tsname = tab.tsname and copy.ictype = 'F'
where tab.dbname = 'DHOMNR01'
group by tab.name, tab.tsname
having max(copy.timestamp) < current_timestamp - 7 days
```

37

Without the having statement, can check to see if timestamp is null:

Image copy never taken

Permissions

- Many different ways a user can get authority to a table
- If user has SYSADM authority, any table can be accessed

Permissions – Direct Access

- To see if DBA gave you permission to a specific table or view:

```
Select stname, selectauth, updateauth, insertauth, deleteauth
, updatecols, alterauth, indexauth, grantor, grantedts
From sysibm.systabauth
Where dbname = 'DHOMNR01'
And ttname = 'THOMREVW'
And grantee = <USER>
```

- *Must know what username <USER> you access the table with

39

If a match is found with this query, you can access the table directly.

'DBHOMUP1' is the RACF group that I am in.

Permissions (con't)

- To see all the tables a user has direct access to:

```
Select stname, selectauth, updateauth, insertauth, deleteauth
, updatecols, grantor
From sysibm.systabauth
Where dbname = 'DHOMNR01'
And grantee = <USER>
order by stname
```

- *Must know what username or RACF groups you use

Output

STNAME	SELECTAUTH	UPDATEAUTH	INSERTAUTH	DELETEAUTH	UPDATECOLS	GRANTOR
THOMAGNT	Y					DBHOMCT
THOMCMCT	Y					DBHOMCT
THOMCMTS	Y					DBHOMCT
THOMCTNB	Y	Y	Y	Y		DBHOMCT
THOMCTPT	Y	Y	Y	Y		DBHOMCT
THOMCVAR	Y					DBHOMCT
THOMEPR	Y					DBHOMCT
THOMHCCA	Y					DBHOMCT
THOMHCRT	Y	Y	Y	Y		DBHOMCT
THOMHZDT	Y					DBHOMCT
THOMHZRD	Y					DBHOMCT
THOMLKUP	Y					DBHOMCT
THOMLOCK	Y					DBHOMCT

Permissions to a column

- If updatecols = 'Y' from previous query, will have a row in this table
- Could also have been given UPDATE or READ ONLY privileges to specific columns. To find that:

```
select tname, colname, grantor, creator, granteetype
from sysibm.syscolauth
where grantee = <USER>
or collid = <ID>
```


The GRANTEETYPE may be either an auth id or an application plan or package. If it is an application plan or package, you need to use COLLID instead of GRANTEE.

Access using Plan

- To see if you can access a table or view using a plan

```
select distinct dep.bname, dep.btype, auth.grantor,
   auth.authhowgot
from sysibm.sysplanauth auth
inner join sysibm.sysplandep dep
on dep.dname = auth.name
where grantee = <USER>
and (dep.btype = 'T' or dep.btype = 'V')
and auth.executeauth <> ''
order by dep.bname
```

43

SYSIBM.SYSPLANAUTH – records the privileges held by users over a plan

BTYPE = T, Table

BTYPE = V, View

Access using Package

- To see if you can access a table or view using a package

```
select distinct dep.bname, dep.btype, auth.grantor,
    auth.authhowgot
from sysibm.syspackauth auth
inner join sysibm.syspackdep dep
on dep.dname = auth.name
where grantee = <USER>
and (dep.btype = 'T' or dep.btype = 'V')
and auth.executeauth <> ''
order by dep.bname
```

44

SYSIBM.SYSPLANAUTH – records the privileges held by users over a package

BTYPE = T, Table

BTYPE = V, View

Section 2:

SQL your DBAs wish you knew.

SQL you should know that can help out your DBAs

PLAN_TABLE

- Contains at least one row for each table referenced in each SQL statement being explained
- Typically, this table contains only rows for STATIC SQL that was bound with EXPLAIN=YES

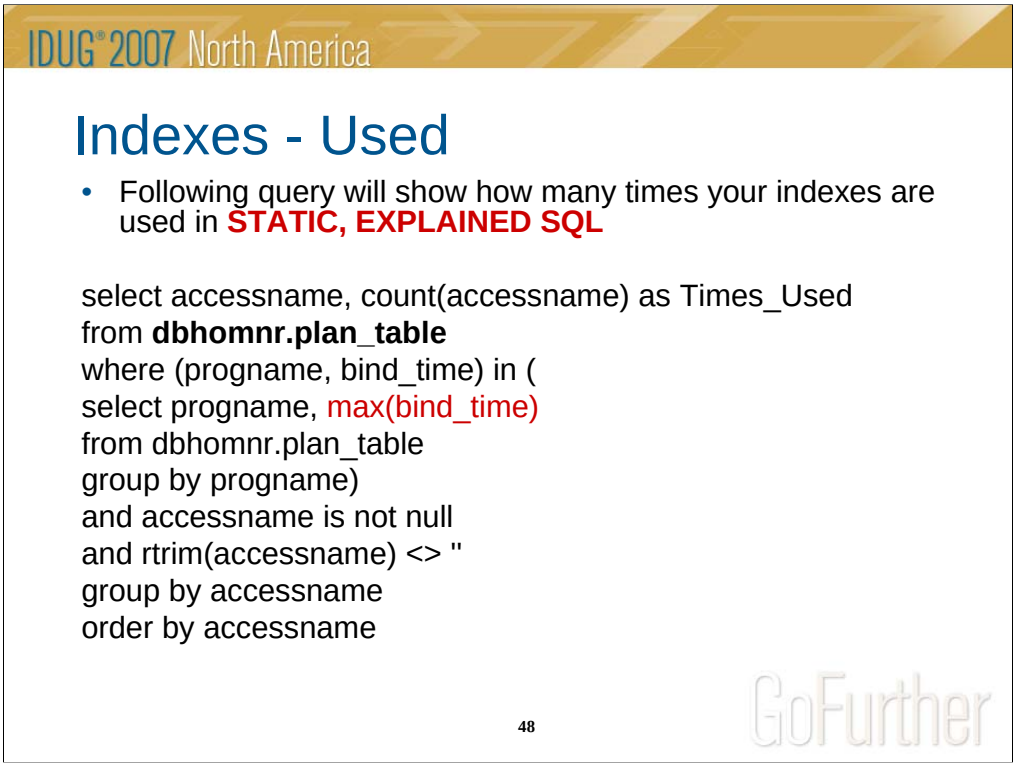
- Can also be used for Dynamic SQL

Dynamic SQL

- One way to obtain access path information for individual dynamic sql statements is to use the EXPLAIN PLAN command

```
EXPLAIN PLAN
SET QUERYNO = 888 FOR
<Select statement>
```


In this example, I also used SET QUERYNO, that is optional.



Must know the qualifier of the plan_table where your EXPLAINED SQL is inserted.

Instead of bind_time, you may need/want to use version. Using bind_time gives you just the last time it was bound. It may not be the latest version. Someone could have bound an older version.

Also, if you are using an older version of the PLAN_TABLE (prior to DB2 v6), it may not even contain the column bind_time so you may have to use version.

```
select accessname, count(accessname) as Times_Used
from dbcornr.plan_table
where (progname, version, left(timestamp, 8)) in (
select progname, max(version), max(left(timestamp, 8))
from dbcornr.plan_table
group by progname)
and accessname is not null
and rtrim(accessname) <> ''
group by accessname
order by accessname
```

Output

ACCESSNAME	TIMES_USED
XHOMCMCT	2
XHOMCMTS	1
XHOMCON3	1
XHOMCTN2	6
XHOMCTP2	12
XHOMCVA2	2
XHOMEMA2	1
XHOMHCCA	2
XHOMHCC2	1
XHOMLCQU	1

Indexes – Not Used

- To make sure all your indexes are being used, find the ones that are not

```
select name, tbname, clustering, uniquerule, colcount, statstime
from sysibm.sysindexes
where dbname = 'DHOMNR01'
and name not in (
select accessname
from dbhomnr.plan_table
where (progname,bind_time) in (
select progname, max(bind_time)
from dbhomnr.plan_table
group by progname))
```


Using Version:

```
select name, tbname, clustering, uniquerule, colcount, statstime
from sysibm.sysindexes
where dbname = 'DCORNR01'
and name not in (
select accessname
from dbcornr.plan_table
where (progname, version, left(timestamp, 8)) in (
select progname, max(version), max(left(timestamp, 8))
from dbcornr.plan_table
group by progname))
```

Output

NAME	TBNAME	CLUSTERING	UNIQUERULE	COLCOUNT	STATSTIME
XNEWADDR	THOMADDR	Y	P	1	1/21/2007 8:29
XHOMAGRI	THOMAGRI	Y	P	2	1/21/2007 8:31
XHOMAGR2	THOMAGRI	N	U	1	1/21/2007 8:31
XHOMCLASS	THOMCLASS	Y	P	2	1/21/2007 8:31
XHOMCOA2	THOMCOAC	N	D	1	1/21/2007 8:31
XNEWCONT	THOMCONT	Y	U	3	1/21/2007 8:06
XNEWCON2	THOMCONT	N	P	1	1/21/2007 8:06
XHOMCRD0	THOMCRDT	N	P	1	
XHOMCTNB	THOMCTNB	Y	P	1	1/21/2007 8:30
XHOMCVRG	THOMCVRG	Y	P	2	1/21/2007 8:31
XHOMEMAIL	THOMEMAIL	Y	P	1	1/21/2007 8:31

Locate tablescans and indexscans

- Search plan_table for tablescans and non-matching index scans

```
select queryno, progname, tname, correlation_name as alias
, tab.cardf , accessname, accesstype, matchcols, method, bind_time
from dbhomnr.plan_table pt
left outer join sysibm.systables tab
on tab.name = pt.tname
where (progname, bind_time) in (
    select progname, max(bind_time)
    from dbhomnr.plan_table
    group by progname)
and ((accesstype = 'R')
or (accesstype = 'I' and matchcols = 0))
order by progname, timestamp, queryno, qblockno, planno
```


For older versions of the PLAN_TABLE may need to use the following:

```
select queryno, progname, tname, tab.cardf , accessname, accesstype,
matchcols, method
from dbcornr.plan_table pt
left outer join sysibm.systables tab
on tab.name = pt.tname
and tab.dbname = 'DCORNR01'
where (progname, version, left(timestamp,8)) in (
                select progname, max(version), max(left(timestamp,8))
                from dbcornr.plan_table
                group by progname)
and ((accesstype = 'R')
or (accesstype = 'I' and matchcols = 0))
order by progname, queryno
```

Output

QUERYNO	PROGNAME	TNAME	ALIAS	CARDF	ACCESSNAME	ACCESSTYPE	MATCHCOLS	METHOD	BIND_TIME
958	ALPIREVW	THOMPARTY	E	3405048	XHOMPRT2	I	0	1	2/12/2007 9:47
343	ALPISLOT	THOMHCRP	HCRP	1061		R	0	0	2/12/2007 9:47
377	ALPISLOT	THOMHCRP	HCRP	1061		R	0	0	2/12/2007 9:47
420	ALPISLOT	THOMHCRP	HCRP	1061		R	0	0	2/12/2007 9:47
1051	ALPISRCH	THOMUNDW	UNDW	1007		R	0	0	2/12/2007 9:47
1060	ALPISRCH	THOMHCRP	HCRP	1061		R	0	0	2/12/2007 9:47
1206	ALPISRCH	THOMUNDW	UNDW	1007	XHOMUNDW	I	0	0	2/12/2007 9:47
1310	ALPISRCH	THOMUNDW	UNDW	1007	XHOMUNDW	I	0	0	2/12/2007 9:47
1419	ALPISRCH	THOMCOAC	COAC	13513		R	0	0	2/12/2007 9:47
1521	ALPISRCH	THOMUNDW	UNDW	1007	XHOMUNDW	I	0	0	2/12/2007 9:47
1624	ALPISRCH	THOMUNDW	UNDW	1007	XHOMUNDW	I	0	0	2/12/2007 9:47
1733	ALPISRCH	THOMUNDW	UNDW	1007	XHOMUNDW	I	0	0	2/12/2007 9:47
1841	ALPISRCH	THOMHCRP	HCRP	1061		R	0	0	2/12/2007 9:47
272	ALPIUND	THOMUNDW	UNDW	1007		R	0	0	2/12/2007 9:47

Long Running Queries – Step I

- Another Option besides tablescans and index scans
- Find queries with the highest procms

```
select queryno, progname, cost_category, procms
, procsu, reason
from dbhomnr.dsn_statemnt_table
where (progname, explain_time) in (
select progname, max(explain_time)
from dbhomnr.dsn_statemnt_table
group by progname)
order by procms desc
```

54

DSN_STATEMNT_TABLE – contains a row with cost estimate, in service units and milliseconds for processing of an explainable statement.

The DSN_STATEMNT_TABLE may not exist so may not be able to use this. Prior to Version 6, this table did not exist.

Output

QUERYNO	PROGNAME	COST_CATEGORY	PROCMS	PROCSU	REASON
182	ALPURPT	A	5176	90799	
421	HOMMAIL2	A	2508	43983	
460	HOMEMAIL	A	2455	43063	
1402	HOMMLFMT	B	2407	42221	HOST VARIABLES
4890	HOMREVWS	B	1743	30564	HOST VARIABLES
3529	HOMREVWS	A	1697	29757	
457	HOMARENL	A	1615	28323	
436	HOMARENL	A	1486	26053	
3718	HOMREVWS	A	1165	20430	
629	HOMCLN	B	911	15976	TRIGGERS
4133	HOMREVWS	A	494	8665	
911	DSNESM68	A	354	3402	
811	HOMMSRCH	B	152	2659	HOST VARIABLES
586	HOMCLN	B	126	2194	TRIGGERS
330	HOMSHCRP	A	93	1629	

Long Queries – Step II

- Find the SQL for the statement that has a high cost:

```
select name, seqno, stmt, status, explainable,
   version
from sysibm.syspackstmt
where name = 'HOMMAIL2'
and queryno = 421
order by version desc, seqno
```


Using the progname and queryno found in previous query.

Output

NAME	SEQNO	STMT	STATUS	EXPLAINABLE	VERSION
HOMMAIL2	5	*f* DECLARE CMCT_SELECT CI	C	Y	2006-06-26-14.34.58.442167
HOMMAIL2	6	E 'CM%' ORDER BY MAIL_TYPE , M	C	Y	2006-06-26-14.34.58.442167

Long Queries – Step III

- Could go back to the plan_table to see the path that the optimizer chose:

```
select progname, tname, planno, method, accesstype,
    accessname, version
from dbhomnr.plan_table
where (progname, bind_time) in (
select progname, max(bind_time)
from dbhomnr.plan_table
group by progname)
and progname = 'HOMMAIL2'
and queryno = 421
```

58

Using version:

```
select progname, tname, planno, method, accesstype, accessname, version
from dbcornr.plan_table
where (progname, version, left(timestamp, 8)) in (
select progname, max(version), max(left(timestamp, 8))
from dbcornr.plan_table
group by progname)
and progname = 'CORBALHR'
and queryno = 629
```

Output

PROGNAME	TNAME	PLANNO	METHOD	ACCESSTYPE	ACCESSNAME	VERSION
HOMMAIL2	THOMCMCT	1	0	I	XHOMCMC3	2006-06-26-14.34.58.442167
HOMMAIL2		2	3			2006-06-26-14.34.58.442167

Should have also included matchcols to see if it is a nonmatching indexscan.

History for Visual EXPLAINs

- How long do you keep the data in the EXPLAIN tables
- When does it get cleaned up?
- What can you do with the information?

History – Con't

- How far do the records actually go back in the dsn_statemnt_table

Select min(explain_time)

From dbhomnr.dsn_statemnt_table

10/29/2004 10:33:30 AM

This also happens to be the first time our programs were put into production

History – Con't

- To find out when a program has been explained :

```
select distinct date(explain_time) as dnsdate
from dbhomnr.dsn_statemnt_table
where progname = 'HOMIHCRP'
order by dnsdate desc
```


For our area, the number of times would correspond to the number of times the program has been put into production since we do records are only added to this time when they go into production.

Output

DNSDATE
11/17/2006
7/22/2006
5/13/2006
5/12/2006
4/21/2006
4/3/2006
4/2/2006
3/17/2006
12/3/2005
12/2/2005
9/6/2005
6/17/2005
4/28/2005
11/21/2004
11/8/2004
11/7/2004
11/3/2004

What ELSE can you do with information?

- Can look at trends for each program (cost_category, procms, procsu, reason)
  - May need to spend some time matching up queries between history in dsn_statemnt_table
  - queryno may not match, new queries can be added or queryno changed
- Can export to another table to store history before information is purged

There is another session on why keeping EXPLAIN history is good.

Queryno

- To make using the history easier, you can always add QUERYNO to the end of your SQL Statements to keep your QUERYNO the same

```
SELECT *
FROM T1
WHERE C1 = 10
AND C2 BETWEEN 10 AND 20
QUERYNO 100
```

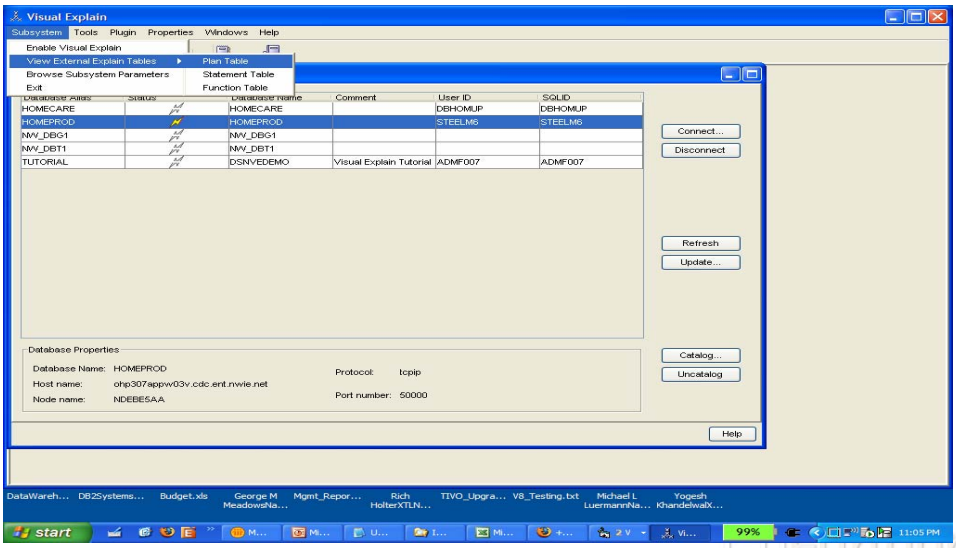
Easy way to get SQL for Visual Explain

- As already shown in a previous slide can use sysibm.syspackstmt

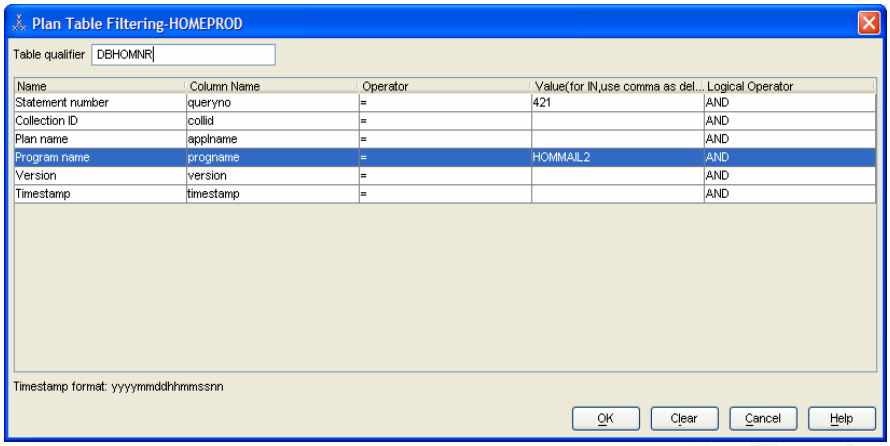
```
select name, stmt, status, explainable, version
from sysibm.syspackstmt
where name = 'HOMMAIL2'
and queryno = 421
order by seqno
```


The text does not come out the best but you can easily get the SQL out of it.

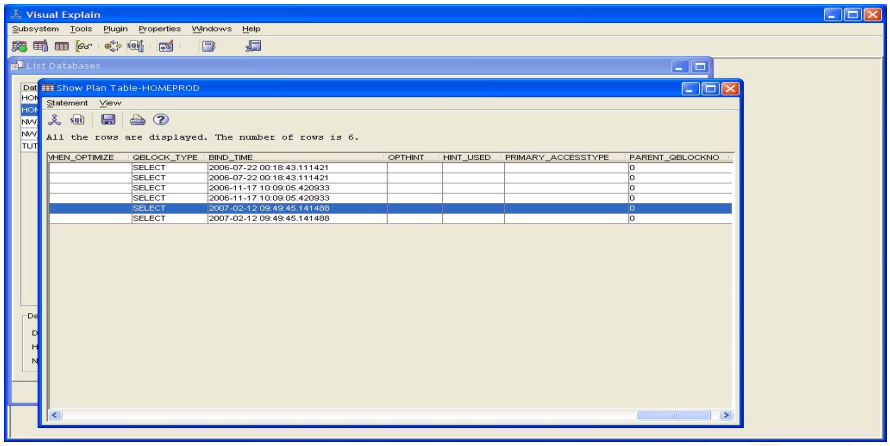
Use Visual Explain to get SQL



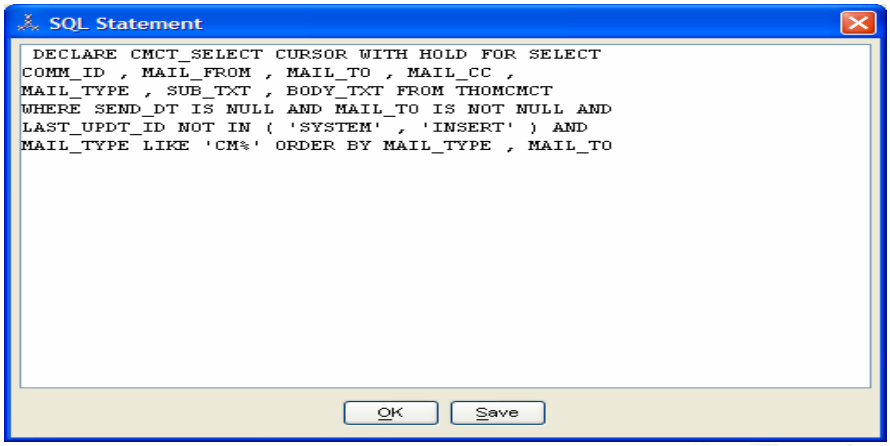
Use Visual Explain to get SQL – Con't



Use Visual Explain – Con't



Use Visual Explain – Con't



Section 3:

SQL you should know as an Application Developer.

Start out Easy – Tables in a Database

- List all tables that exist in a database

```
select name, type, cardf, colcount, reclength
from sysibm.systables
where dbname = 'DHOMNR01‘
order by name
```


Make sure DBA has added the table

Can do the same with sysibm.sysview, sysibm.systriggers, etc. (Just need to check columns available)

Output

NAME	TYPE	CARDF	COLUMNS	RECLENGTH
APDSADR1	A	-1	0	0
APDSELID	A	-1	0	0
DSN_STATEMNT_TABLE	T	13070	11	335
PLAN_TABLE	T	39263	51	810
THOMADDR	T	3052497	17	486
THOMAGNT	T	25917	7	161
THOMAGRI	T	385	12	108
THOMCALT	G	-1	27	719
THOMCLASS	T	9571	6	98
THOMCMCT	T	19673737	14	10783
THOMCMTS	T	3764505	9	643
THOMCOAC	T	10609	19	289
THOMCONT	T	2200829	6	88
THOMCTNB	T	2177673	7	102

Columns in a table

- List all columns and some information about the columns from a specified table

```
select cols.name, cols.coltype, cols.length, cols.scale
, cols.nulls, cols.colcard
from sysibm.syscolumns cols
inner join sysibm.systables tab
on tab.name = cols.tbname
where tab.dbname = 'DHOMNR01'
and cols.tbname = 'THOMREVW'
order by colno
```


Added join to sysibm.systables just to use dbname = ‘DHOMNR01’ this is not needed

Output

NAME	COLTYPE	LENGTH	SCALE	NULLS	COLCARD
REVIEW_ID	CHAR	20	0	N	1652601
CONF_TXT	CHAR	50	0	Y	1652601
POLICY_NUM	CHAR	20	0	Y	1397487
HCR_ID	CHAR	20	0	Y	895
INSP_STATUS	CHAR	10	0	Y	16
DCSN_PNT_RVW_IND	CHAR	1	0	Y	4
AGENT_REBUTTAL_RES	CHAR	50	0	Y	4864
INSP_DRVD_INSR_AMT	DECIMAL	15	0	Y	327680
CREATOR_NOTES_ID	CHAR	20	0	Y	12160
INSP_ELPSD_HRS	DECIMAL	3	2	Y	12
EFF_DT	DATE	4	0	Y	2095

Indexes on a Table

- List all indexes on a table

```
select idx.name, idx.tbname, keys.colname, keys.colseq
, idx.uniquerule, idx.clustering, idx.fullkeycard, idx.createdts
from sysibm.sysindexes idx
, sysibm.syskeys keys
where idx.dbname = 'DHOMNR01'
and idx.tbname = 'THOMCMTS'
and keys.ixname = idx.name
and keys.ixcreator = idx.creator
order by idx.name, keys.colseq
```


Make sure DBA has added the index and that it is correct (right unique attribute, is it the clustered index, etc.)

I would run the following SQL and KEEP A COPY OF IT ON YOUR WALL for easy access:

```
select idx.tbname, idx.name, idx.tbname, keys.colname, keys.colseq
, idx.uniquerule, idx.clustering, idx.fullkeycard, idx.createdts
from sysibm.sysindexes idx
, sysibm.syskeys keys
where idx.dbname = 'DHOMNR01'
and keys.ixname = idx.name
and keys.ixcreator = idx.creator
order by idx.tbname, idx.name, keys.colseq
```

Output

NAME	TBNAME	COLNAME	COLSEQ	UNIQUERULE	CLUSTERING	FULLKEYCARD	CREATEDTS
XHOMCMTS	THOMCMTS	COMMENT_ID	1	P	N	3956200	8/27/2004 22:01
XHOMCMT2	THOMCMTS	REVIEW_ID	1	D	Y	2307967	6/16/2005 22:33
XHOMCMT2	THOMCMTS	HAZ_DETAIL_ID	2	D	Y	2307967	6/16/2005 22:33
XHOMCMT2	THOMCMTS	COMMENT_TYP_LKUP	3	D	Y	2307967	6/16/2005 22:33
XHOMCMT3	THOMCMTS	CO_ACCT_POL_NUM	1	D	N	12861	2/9/2007 23:11

Foreign Keys on a Table

- To find all foreign keys associated with a table:

```
select rels.relname, rels.tbname, rels.reftbname
, fk.colname, fk.colseq, rels.deleterule, rels.enforced, rels.timestamp
from sysibm.sysforeignkeys  fk
inner join sysibm.sysrels rels
on fk.relname = rels.relname
and fk.tbname = rels.tbname
and fk.creator = rels.creator
where fk.tbname = 'THOMCMTS'
and fk.creator = 'DBHOMNR'
order by rels.tbname, rels.relname, fk.colseq
```

78

Make sure DBA has created all necessary foreign keys.

RELNAME – Name of the constraint

TBNAME – Name of dependent table of relationship

REFTBNAME – Name of parent table in the relationship

COLNAME – Name of the column from the dependent table that the constraint is based on

COLSEQ – Numerical place of the column in the foreign key

DELETERULE – Type of delete rule for referential constraint

ENFORCED – Enforced by the system or not

TIMESTAMP – Date and time constraint was defined

Output

RELNAME	TBNAME	REFTBNAME	COLNAME	COLSEQ	DELETERULE	TIMESTAMP
COM$LKUP	THOMCMTS	THOMLKUP	COMMENT_TYP_LKUP	1	R	2004-08-27 22:03
HAZ$DTAL	THOMCMTS	THOMHZDT	HAZ_DETAIL_ID	1	R	2004-08-27 22:03
REVW$ID	THOMCMTS	THOMREVW	REVIEW_ID	1	R	2006-04-01 23:38

Values for DELETERULE:

C – Cascade

R – Restrict

N – Set Null

A – No Action

Previous slide I also used rels.enforced but due to space, that was not included:

ENFORCED = Y, Enforced by the system

Stored Procedures in a Schema

- List all Stored Procedures in a Schema

```
select name, routinetype, language, parm_count
, wlm_environment, result_sets
from sysibm.sysroutines
where schema = 'DBHOMNR'
order by name
```


I used this at Nationwide in an application that I wrote to test stored procedures. This query populates a drop-down box to select the stored procedure you want to test.

Could also use COLLID = instead of SCHEMA

Output

NAME	ROUTINETYPE	LANGUAGE	PARM_COUNT	WLM_ENVIRONMENT	RESULT_SETS
ALPILTTR	P	COBOL	7	DBHOMNR	2
ALPIREVW	P	COBOL	4	DBHOMNR	7
ALPISLOT	P	COBOL	5	DBHOMNR	1
ALPISRCH	P	COBOL	14	DBHOMNR	5
ALPIUND	P	COBOL	4	DBHOMNR	1
ALPIVEND	P	COBOL	2	DBHOMNR	5
ALPUACK	P	COBOL	4	DBHOMNR	0
ALPUBLDG	P	COBOL	8	DBHOMNR	0
ALPUCUQU	P	COBOL	8	DBHOMNR	1
ALPUCURC	P	COBOL	8	DBHOMNR	1
ALPULOC	P	COBOL	11	DBHOMNR	1
ALPUREVW	P	COBOL	9	DBHOMNR	0
ALPUSTAT	P	COBOL	8	DBHOMNR	0
ALPUSTQU	P	COBOL	8	DBHOMNR	0
ALPUSTRC	P	COBOL	8	DBHOMNR	0
ALPUUND	P	COBOL	11	DBHOMNR	0
ALPUVEND	P	COBOL	18	DBHOMNR	0

Stored Procedure Parameters

- Input/Output parameters for a Stored Procedure

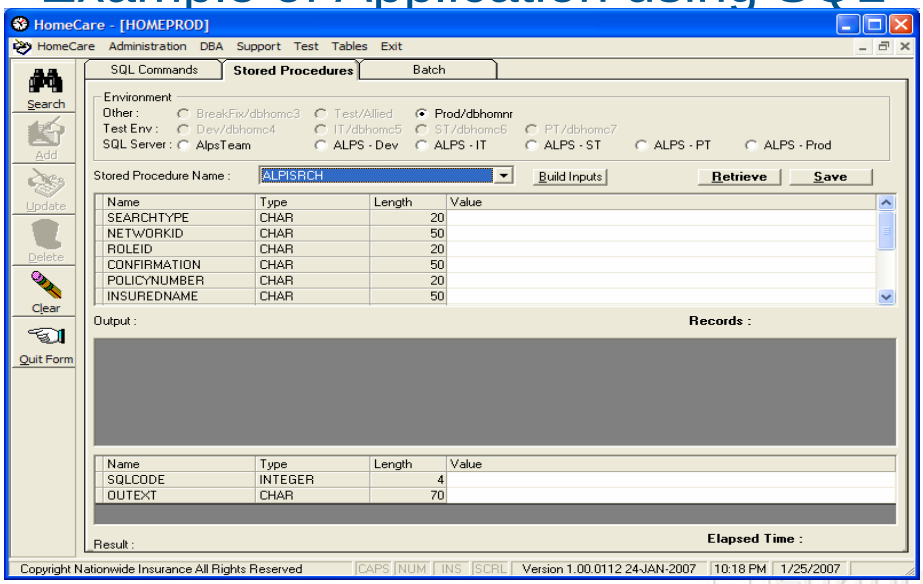
```
select parmname, rowtype, typename, length, scale
, encoding_scheme
from sysibm.sysparms
where schema = 'DBHOMNR'
and   name = 'ALPISRCH'
order by name, ordinal
```


I use this procedure to retrieve all the parameters (inputs/outputs) in a stored procedure for the tester application I wrote.

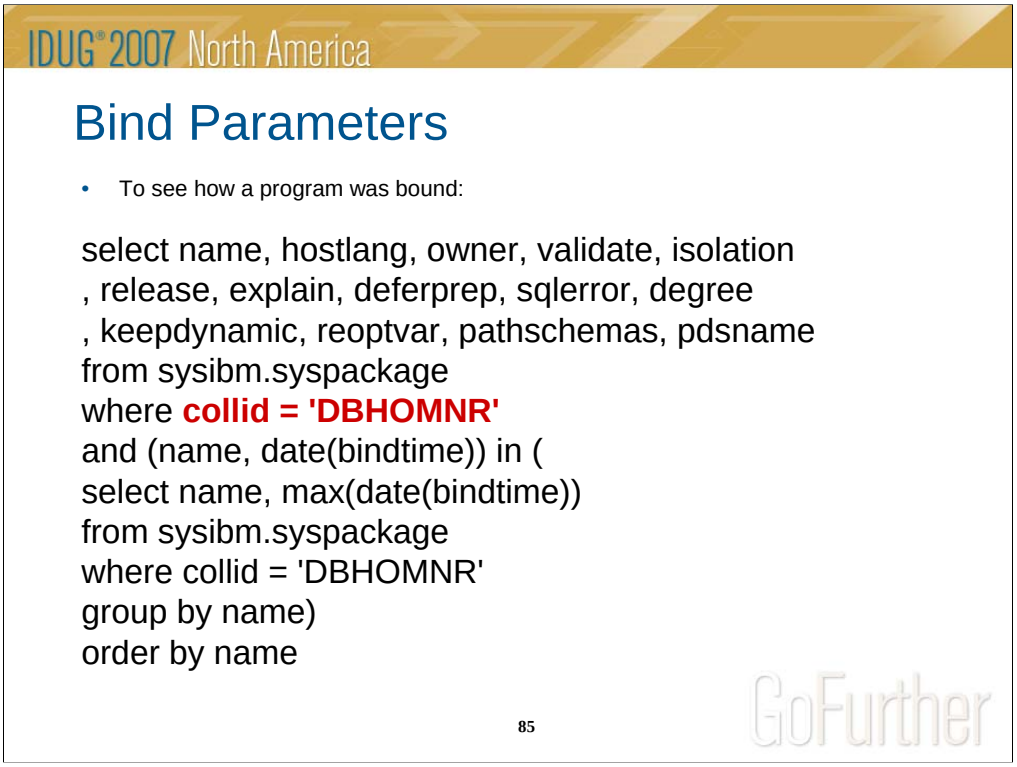
Output

PARMNAME	ROWTYPE	TYPENAME	LENGTH	SCALE	ENCODING_SCHEME
SEARCHTYPE	P	CHAR	20	0	E
NETWORKID	P	CHAR	50	0	E
ROLEID	P	CHAR	20	0	E
CONFIRMATION	P	CHAR	50	0	E
POLICYNUMBER	P	CHAR	20	0	E
INSUREDNAME	P	CHAR	50	0	E
LOCATIONADDRESS	P	CHAR	50	0	E
LOCATIONCITY	P	CHAR	50	0	E
LOCATIONSTATE	P	CHAR	2	0	E
LOCTIONZIP	P	CHAR	10	0	E
HCRID	P	CHAR	20	0	E
UNDERWRITID	P	CHAR	20	0	E
SQLCODE	O	INTEGER	4	0	
OUTEXT	O	CHAR	70	0	E

Example of Application using SQL



This is an example of the Stored Procedure that I wrote that uses the last two queries to fill in the list of Stored Procedures and the List of Parameters.



This can be used by Application Developers and DBAs to see how a program was bound.

Here is the PKG file that we used to bind ALPILTTR:

```
BIND PACKAGE (DBHOMNR)   MEMBER (ALPILTTR)
    OWNER (DBHOMNR)     QUALIFIER (DBHOMNR)
    ACTION (REPLACE)    VALIDATE (BIND)
    ISOLATION (CS)      RELEASE (COMMIT)
    EXPLAIN (YES)       CURRENTDATA (YES)
    SQLERROR(NOPACKAGE) DEGREE(1)
    NOREOPT(VARS)       KEEPDYNAMIC(NO)
    PATH (SCHEMA)
    FLAG(I)
```

There are a lot of other sessions that tell what bind parameters should be used. This just shows you what they were.

Output

NAME	OWNER	ISOLATION	RELEASE	EXPLAIN	REOPTVAR	PDSNAME
ALPILTTR	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002310.DBR
ALPIREVW	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002310.DBR
ALPISLOT	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002035.DBR
ALPISLOT	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002111.DBR
ALPISLOT	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002221.DBR
ALPISRCH	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002310.DBR
ALPIUND	DBHOMNR	S	C	Y	N	OCHM.FIRE.#001896.DBR
ALPIUND	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002035.DBR
ALPIUND	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002111.DBR
ALPIVEND	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002310.DBR
ALPRPT1	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002351.DBR
ALPRPT2	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002273.DBR
ALPRPT3	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002273.DBR
ALPSPLT	DBHOMNR	S	C	Y	N	OCHM.FIRE.#002351.DBR

This does not show all of the columns from the previous query due to space on slide.

Triggers in a Schema

- Find all Triggers and how the triggers are set up

```
select trig.name, trig.tbname, trig.seqno, trig.trigtime
, trig.trigevent, trig.granularity
, ltrim(trig.text) as trigtext, trig.createdts
from sysibm.systriggers trig
inner join sysibm.sysdatabase db
on db.dbid = trig.dbid
where db.name = 'DHOMNR01'
```


I used TBOWNER instead of OWNER or another column because TBOWNER is the owner of the table which the trigger applies. OWNER in our case would not show all of the triggers since different OWNER IDs were used.

Output

NAME	TBNAME	SEQNO	TRIGTIME	TRIGEVENT	GRANULARITY	TEXT
ADDHIST	THOMREVW	1	A	U	R	CREATE TRIGGER DBHOMC
HCRPHIST	THOMHCRP	1	A	U	R	CREATE TRIGGER DBHOMC

TRIGTIME – Time when triggered actions are applied to base table, relative to the event that activated the trigger:

B – Trigger is applied before the event

A – Trigger is applied after the event

TRIGEVENT – Operation that activated the trigger:

I – Insert

D – Delete

U – Update

GRANULARITY – Trigger is executed once per statement or row:

S – Statement

R – Row

TEXT – Full text of the CREATE TRIGGER statement

Section 4: Additional useful SQL

Packages Accessing a Table

- Can use to find all packages that are dependent on a certain table, view, synonym, tablespace, alias, function or stored procedure

```
select distinct dname, dcollid, bqualifier, dtype
from sysibm.syspackdep
where bname = 'THOMREVW'
```


This SQL can be used if a table is changing and you need to know what other programs are affected.

Output

DNAME	DCOLLID	BQUALIFIER	DTYPE
ADDHIST	DBHOMCT	DBHOMNR	T
ALPILTTR	DBHOMNR	DBHOMNR	
ALPIREVW	DBHOMNR	DBHOMNR	
ALPISRCH	DBHOMNR	DBHOMNR	
ALPIVEND	DBHOMNR	DBHOMNR	
ALPUACK	DBHOMNR	DBHOMNR	
ALPUCUQU	DBHOMNR	DBHOMNR	
ALPUCURC	DBHOMNR	DBHOMNR	
ALPULOC	DBHOMNR	DBHOMNR	
ALPUREVW	DBHOMNR	DBHOMNR	
ALPURPT	DBHOMNR	DBHOMNR	

DTYPE – Package Type

T – Trigger package

Blank – Not trigger package

Package Dependencies

- Find tables, views, synonyms, tablespaces, aliases, indexes, functions or stored procedures that a package is dependent on

```
select distinct bname, bqualifier, btype, dtype
from sysibm.syspackdep
where dname = 'ALPISRCH'
order by btype, bname
```

Output

BNAME	BQUALIFIER	BTYPE	DTYPE
XHOMCOAC	DBHOMNR	I	
XHOMCOA3	DBHOMNR	I	
XHOMCTN2	DBHOMNR	I	
SHOMADDR	DHOMNR01	R	
SHOMCMTS	DHOMNR01	R	
SHOMCOAC	DHOMNR01	R	
SHOMCOTP	DHOMNR01	R	
SHOMCTNB	DHOMNR01	R	
THOMADDR	DBHOMNR	T	
THOMCLASS	DBHOMNR	T	
THOMCMTS	DBHOMNR	T	
THOMCOAC	DBHOMNR	T	
THOMCTNB	DBHOMNR	T	
THOMCTPT	DBHOMNR	T	
THOMCVRG	DBHOMNR	T	
THOMHCRP	DBHOMNR	T	
THOMPARTY	DBHOMNR	T	
THOMREVW	DBHOMNR	T	
THOMSTRU	DBHOMNR	T	
THOMUNDW	DBHOMNR	T	

BNAME – Name of an object the package is dependent on

BQUALIFIER – If BNAME identifies a tablespace, the name of its database. Otherwise, the authorization ID of the owner of BNAME.

BTYPE – Object identified by BNAME and BQUALIFIER

A – Alias
F – User defined function
I – Index
M – Materialized query table
O – Stored Procedure
P – Partitioned tablespace
Q – Sequence object
R – Tablespace
S – Synonym
T – Table
V - View

Table Size

- To find the size of a table:

```
select tab.name, tab.card, tab.npagesf, tab.spacef
, sum(col.length) as reclength
, tab.card * sum(col.length) as Size
from sysibm.systables tab
inner join sysibm.syscolumns col
on col.tbname = tab.name
where dbname = 'DHOMNR01'
group by tab.name, tab.card, tab.npagesf, tab.spacef
order by name
```

Output

NAME	CARD	NPAGESF	SPACEF	RECLENGTH	SIZE
THOMADDR	3119364	87390	282742	465	1450504260
THOMAGNT	26204	1049	4069	152	3983008
THOMAGRI	483	6	22	98	47334
THOMCLASS	11131	141	527	88	979528
THOMCOAC	12449	282	1074	275	3423475
THOMCONT	2239451	51638	188079	78	174677178
THOMCRDT	-1	-1	-1	48	-48
THOMCTNB	2225890	28600	97450	93	207007770
THOMCTPT	3573637	41119	139767	78	278743686
THOMCVAR	219468	4573	17575	73	16021164
THOMCVRG	26159	306	1156	72	1883448

Extents

- Can find the number of extents in Tablespaces, Indexes, and Partitions

Extents - Tablespace

- To find the number of extents for a tablespace:

```
Select tsname, partition, extents, card, dsnum,
   spacef, statstime
From sysibm.systablepart
Where dbname = 'DHOMNR01'
Order by extents desc
```


Extents – Number of data set extents

DSNUM – Number of data sets

SPACEF – Kilobytes of DASD storage

Output

TSNAME	PARTITION	EXTENTS	CARD	DSNUM	SPACEF	STATSTIME
SHOMPLAN	0	89	40295	1	22752	4/25/2006 3:57:28 PM
SHOMRVQU	10	9	13461110	1	2455200	2/25/2007 8:14:56 AM
SHOMRVQU	2	7	21597028	1	2966400	2/25/2007 8:14:56 AM
SHOMRVQU	4	5	6693900	1	813600	2/25/2007 8:14:56 AM
SHOMRVQU	9	5	7970360	1	989280	2/25/2007 8:14:56 AM
SHOMRVQU	7	5	6692603	1	815040	2/25/2007 8:14:56 AM
SHOMRVHT	0	5	8533853	3	5662800	2/25/2007 8:11:15 AM
SHOMCMTS	0	5	4014427	1	642240	2/25/2007 8:07:57 AM

Extents - Indexes

- To find the number of extents for an index:

```
Select idxpart.ixname, idxpart.partition, idxpart.extents
, idxpart.cardf, idxpart.dsnum, idxpart.spacef, idxpart.statstime
From sysibm.sysindexpart idxpart
inner join sysibm.sysindexes idx
on idx.name = idxpart.ixname
Where idx.dbname = 'DHOMNR01'
Order by extents desc
```

Output

IXNAME	PARTITION	EXTENTS	CARDF	DSNUM	SPACEF	STATSTIME
XHOMORD1	0	8	36103	1	1584	2/25/2007 8:38:49 AM
XHOMORD0	0	6	36103	1	1440	2/25/2007 8:38:49 AM
XHOMRVQU	0	6	84581974	3	5562000	2/25/2007 8:14:56 AM
XHOMEPR	0	4	1287845	1	62640	2/25/2007 8:09:16 AM
XHOMCTP2	0	3	3668362	1	108720	2/25/2007 8:35:54 AM
XHOMSCHT	0	3	1694436	1	62640	2/25/2007 8:35:31 AM
XHOMRVQ1	10	2	13461110	1	114480	2/25/2007 8:14:56 AM
XHOMRVQ1	9	2	7970360	1	68400	2/25/2007 8:14:56 AM
XHOMRVQ1	7	2	6692603	1	59040	2/25/2007 8:14:56 AM

Free Space

- Amount of space currently on pages for tablespaces:

```
Select tsname, percactive, pctfree, freepage
, percdrop
From sysibm.systablepart
Where dbname = 'DHOMNR01'
Order by tsname
```


PERCACTIVE – Percentage of space occupied by rows of data from active tables

PCTFREE - What percentage of each page in a table space or index is left free when loading or reorganizing the data.

FREEPAGE – Number of each page left as free space

PERCDROP – Percentage of space occupied by rows of dropped tables.

Output

TSNAME	PERCACTIVE	PCTFREE	FREEPAGE	PERCDROP
SHOMADDR	77	20	10	0
SHOMAGNT	93	5	0	0
SHOMAGRI	42	10	5	0
SHOMCMCT	73	30	25	0
SHOMCMTS	69	25	20	0
SHOMCOAC	74	10	5	0
SHOMCOTP	71	10	5	0
SHOMCRDT	0	10	10	0
SHOMCTNB	82	15	10	0

Free Space – Allocated/Used

- Compare space allocated with space used

```
Select name, space, nactivef
, space - (nactivef * pgsize) as space_not_used
, nactivef * pgsize * 100/space as perc_space_used
From sysibm.systablespace
Where dbname = 'DHOMNR01'
And space > 0
order by name
```


SPACE – Number of kilobytes of DASD storage allocated to table space (from STOSPACE).

NACTIVEF – Number of active pages in table space formatted for rows.

PGSIZE – Size of pages in table space in kilobytes.

Output

NAME	SPACE	NACTIVEF	SPACE_NOT_USED	PERC_SPACE_USED
SHOMADDR	391680	93769	16604	95.7608251633987
SHOMAGNT	5040	1101	636	87.3809523809524
SHOMAGRI	96	14	40	58.3333333333333
SHOMCMCT	6082560	177607	399136	93.4380260942761
SHOMCMTS	642240	149700	43440	93.2361733931241
SHOMCOAC	1728	424	32	98.1481481481481
SHOMCOTP	960	222	72	92.5

Section 5: System Tables

Catalog Tables Used

- systables
- sysindexes
- syskeys
- systablepart
- sysindexpart
- systablespace
- syscopy
- systabaut
- syscolauth
- sysplanauth
- sysplandep
- syspackauth
- syspackdep
- sysdbauth
- syspackstmt
- syscolumns
- sysforeignkeys
- sysrels
- sysroutines
- sysparms
- syspackage
- systriggers
- sysdatabase

User Tables Used

- plan_table
- dsn_statemnt_table

- Must know qualifier to use these tables

Reference

- To find all the columns in the catalog tables and their definitions:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.sqlref/bjnrmstr602.htm

Session: F07

SQL your DBA wishes you did not know (and some they wish you did)

Matt Steele

Nationwide Insurance

SteeleM6@Nationwide.com

GoFurther