# Informix: Exploring the Sysmaster Database

When you list all the databases on your INFORMIX server, you will see one called "sysmaster". This is a special database and is one of the new features that first appeared in INFORMIX-OnLine DSA 6.x and 7.x. This is a database that contains tables that can be used for monitoring your system. These are referred to as the System Monitoring Interface (SMI) tables. In this chapter we will explore some of the tables and views that are in this database.

The sysmaster database is described as a pseudo database. That means most of its tables are not normal tables on disk, but pointers to shared memory structures in the OnLine engine. The sysmaster database contains over 120 tables. Only 18 of these tables are documented in the INFORMIX-OnLine Dynamic Server Administrator's Guide, Volume 2, Chapter 38. The rest are undocumented and described by Informix as for internal use. The examples and references in this article are based on OnLine 7.23. I have also tested some of the examples with versions 7.10, 7.12, and 7.22. There are some minor changes between versions in the undocumented features and structures of these tables.

A warning: *Some of the features discussed in this article are based on undocumented SMI tables and may change or not work in future versions of INFORMIX OnLine DSA.*

This article will focus on users, server configuration, dbspaces, chunks, tables, and monitoring IO using the sysmaster database. We will present how to create scripts to monitor the following:

* List who is using each database.
* Display information about your server configuration.
* Display how much free space is available in each dbspace in a format like the Unix df command.
* List the status and characteristics of each chunk device.
* Display blocks of free space within a chunk. This allows you to plan where to put large tables without fragmenting them.
* Display IO statistics by chunk devices.
* Display IO usage of chunk devices as a percent of the total IO, and show which chunks are getting used the most.
* Display tables and the number of extents, and number of pages used.
* Present a layout of dbspace, databases, tables, and extents similar to the command "tbcheck -pe".
* Show table usage statistics sorted by which tables have the most reads, writes, or locks.
* Show statistics of users sessions.
* Show locks and users who are waiting on locks.

1. A Practical Example - Who is Using What Database

Let's begin with a very practical example of the sysmaster database's value.

My interest in this database started a couple of years ago, while consulting on a project for a development group where I needed to know who had a database open and which workstation they were using to connect to the database. This was a development environment and there were continual changes to the database schemas. In order to make updates to the database schema, I would have to get the developers to disconnect from the database. The "onstat -u" utility would tell me which users were connected to the server, but not what database and what workstation they were using. "Onstat -g ses" told me the user and workstation, but not the database. "Onstat -g sql told me the session id and database, but not the user name and workstation. After some debugging, I found all the information I wanted in the sysmaster database. And, because it was a database, I could retrieve it with SQL queries. The following query shows the database, who has it open, the workstation they are connected from, and the session id.

Figure 1. Dbwho SQL script

-- dbwho.sql

select sysdatabases.name database, -- Database Name

syssessions.username, -- User Name

syssessions.hostname, -- Workstation

syslocks.owner sid -- Informix Session ID

from syslocks, sysdatabases , outer syssessions

where syslocks.tabname = "sysdatabases" -- Find locks on sysdatabases

and syslocks.rowidlk = sysdatabases.rowid -- Join rowid to database

and syslocks.owner = syssessions.sid -- Session ID to get user info

order by 1;

Every user that opens a database opens a shared lock on the row in the sysdatabases table of the sysmaster database that points to that database. First we need to find all the locks in syslocks on the sysdatabases table. This gives us the rowid in sysdatabase which has the database name. Finally, we join with the table syssessions to get the username and hostname. I put all this together in a shell script that can be run from the unix prompt and called it dbwho. Figure 2 contains the shell script.

Figure 2. Dbwho shell script

:

###########################################################################

# Program: dbwho

# Author: Lester Knutsen

# Date: 10/28/1995

# Description: List database, user and workstation of all db users

###########################################################################

echo "Generating list of users by database ..."

dbaccess sysmaster - <<EOF

select

sysdatabases.name database,

syssessions.username,

syssessions.hostname,

syslocks.owner sid

from syslocks, sysdatabases , outer syssessions

where syslocks.rowidlk = sysdatabases.rowid

and syslocks.tabname = "sysdatabases"

and syslocks.owner = syssessions.sid

order by 1;

EOF

One of the first things you will notice is that this script is slow. This led me to start digging into what was causing the slow performance. Running this query with set explain turned on (this shows the query optimizer plan) shows that there is a lot of work going on behind the scenes. Syslocks is a view, and it takes a sequential scan of six tables to produce the view. A temp table is created to hold the results of the syslocks view, and this is then joined with the other two tables. The tables sysdatabase and syssessions are also views. And the view syssessions uses a stored procedure, called bitval. Figure 3 contains the output from turning set explain on. In spite of these queries sometimes being a bit slow, these tables are a tremendous value and make it much easier to monitor your database server.

Figure 3: Output from "set explain on" for dbwho.sql

QUERY:

------

create view "informix".syslocks

(dbsname,tabname,rowidlk,keynum,type,owner,waiter)

as select x1.dbsname ,x1.tabname ,x0.rowidr ,x0.keynum ,

x4.txt [1,4] ,x3.sid ,x5.sid

from "informix".syslcktab x0 ,

"informix".systabnames x1 ,

"informix".systxptab x2 ,

"informix".sysrstcb x3 ,

"informix".flags\_text x4 ,

outer("informix".sysrstcb x5 )

where ((((((x0.partnum = x1.partnum )

AND (x0.owner = x2.address ) )

AND (x2.owner = x3.address ) )

AND (x0.wtlist = x5.address ) )

AND (x4.tabname = 'syslcktab' ) )

AND (x4.flags = x0.type ) ) ;

Estimated Cost: 713

Estimated # of Rows Returned: 51

1) informix.syslcktab: SEQUENTIAL SCAN

2) informix.flags\_text: SEQUENTIAL SCAN

Filters: informix.flags\_text.tabname = 'syslcktab'

DYNAMIC HASH JOIN

Dynamic Hash Filters: informix.syslcktab.type = informix.flags\_text.flags

3) informix.systxptab: SEQUENTIAL SCAN

DYNAMIC HASH JOIN

Dynamic Hash Filters: informix.syslcktab.owner =

informix.systxptab.address

4) informix.systabnames: SEQUENTIAL SCAN

Filters: informix.systabnames.tabname = 'sysdatabases'

DYNAMIC HASH JOIN

Dynamic Hash Filters: informix.syslcktab.partnum

informix.systabnames.partnum

5) informix.sysrstcb: SEQUENTIAL SCAN

DYNAMIC HASH JOIN (Build Outer)

Dynamic Hash Filters: informix.systxptab.owner = informix.sysrstcb.address

6) informix.sysrstcb: SEQUENTIAL SCAN

DYNAMIC HASH JOIN

Dynamic Hash Filters: informix.syslcktab.wtlist =

informix.sysrstcb.address

QUERY:

------

select sysdatabases.name database,

syssessions.username,

syssessions.hostname,

syslocks.owner sid

from syslocks, sysdatabases, outer syssessions

where syslocks.rowidlk = sysdatabases.rowid

and syslocks.tabname = "sysdatabases"

and syslocks.owner = syssessions.sid

order by 1

Estimated Cost: 114

Estimated # of Rows Returned: 11

Temporary Files Required For: Order By

1) (Temp Table For View): SEQUENTIAL SCAN

2) informix.sysdbspartn: INDEX PATH

(1) Index Keys: ROWID

Lower Index Filter: informix.sysdbspartn.ROWID = (Temp Table For

View).rowidlk

3) informix.sysscblst: INDEX PATH

(1) Index Keys: sid (desc)

Lower Index Filter: informix.sysscblst.sid = (Temp Table For

View).owner

4) informix.sysrstcb: AUTOINDEX PATH

Filters: informix.bitval(informix.sysrstcb.flags ,'0x80000' )= 1

(1) Index Keys: scb

Lower Index Filter: informix.sysrstcb.scb = informix.sysscblst.address

2. How the Sysmaster Database is Created

The sysmaster database keeps track of information about the database server just like the system tables keep track of information in each database. This database is automatically created when you initialize OnLine. It includes tables for tracking two types of information: the System Monitoring Interface (SMI) tables, and the On-Archive catalog tables. This article will focus on the SMI tables. There is a warning in the documentation not to change any information in these tables as it may corrupt your database server. Also there is a warning that OnLine does not lock these tables, and that all selects from this database will use an isolation level of dirty read. This means that the data can change dynamically as you are retrieving it. This also means that selecting data from the sysmaster tables does not lock any of your users from processing their data. *As mentioned above,* the SMI tables are described as pseudo-tables which point directly to the shared memory structures in OnLine where the data is stored. That means they are not actually on disk. However, because many of the SMI tables are really views, selecting from them does create temporary tables and generate disk activity.

A script located in your directory $INFORMIXDIR/etc. named sysmaster.sql contains the SQL statements to create the sysmaster database. The process of creating it is interesting and outlined as follows:

* First the script creates real tables with the structures of the pseudo tables.
* Then, the table structures of the real tables are copied to temp tables.
* The real tables are then dropped.
* The column in systables that contains partnum is updated to indicate they point to pseudo tables in shared memory.
* The flags\_text table is created which has the interpretations for all the text descriptions and flags used in the SMI tables.
* The stored procedures are created that are used to create the views, two of which may be interesting:

- bitval() is a stored procedure for getting the boolean flag values

- l2date() is a stored procedure for converting unix time() long values to dates

* Finally the script creates the SMI views.
* After the sysmaster script is run the system will execute another script to create the on-archive tables and views in the sysmaster database.

Warning: The sysmaster database is created the first time you go into online mode after you first initialize your system. Do NOT start creating any other database until this process is complete or you may corrupt your sysmaster database. You will need 2000 KB of logical log space to create the sysmaster database. If there are problems creating the sysmaster database, shut your OnLine server down and restart it. This will re-create the sysmaster database. Monitor your online.log file until you see the messages showing the successful completion of building the sysmaster database in the online.log (Figure 4).

Figure 4. Online.log messages showing successful creation of sysmaster database

12:10:24 On-Line Mode

12:10:24 Building 'sysmaster' database ...

12:11:02 Logical Log 1 Complete.

12:11:03 Process exited with return code 1: /bin/sh /bin/sh -c

/u3/informix7/log\_full.sh 2 23 "Logical Log 1 Complete." "Logical Log 1 Complete."

12:11:22 Logical Log 2 Complete.

12:11:23 Process exited with return code 1: /bin/sh /bin/sh -c

/u3/informix7/log\_full.sh 2 23 "Logical Log 2 Complete." "Logical Log 2 Complete."

12:11:26 Checkpoint Completed: duration was 3 seconds.

12:11:40 Logical Log 3 Complete.

12:11:41 Process exited with return code 1: /bin/sh /bin/sh -c

/u3/informix7/log\_full.sh 2 23 "Logical Log 3 Complete." "Logical Log 3 Complete."

12:11:59 Logical Log 4 Complete.

12:12:00 Process exited with return code 1: /bin/sh /bin/sh -c

/u3/informix7/log\_full.sh 2 23 "Logical Log 4 Complete." "Logical Log 4 Complete."

12:12:25 'sysmaster' database built successfully.

Supported SMI Tables

There are 18 supported SMI tables in release 7.23 of INFORMIX-OnLine DSA. We will discuss the more important ones and a few unsupported ones in this chapter.

|  |
| --- |
| Sysmaster ERDiagram |

Figure 5. Supported SMI tables

Supported tables and views: (OnLine 7.23)

sysadtinfo Auditing configuration table

sysaudit Auditing event masks table

syschkio Chunk I/O statistics view

syschunks Chunk information view

sysconfig Configuration information view

sysdatabases Database information view

sysdbslocale Locale information view

sysdbspaces Dbspace information view

sysdri Data replication view

sysextents Table extent allocation view

syslocks Current lock information view

syslogs Logical Log status view

sysprofile Current system profile view

sysptptof Current table profile view

syssessions Current user sessions view

sysseswts Session wait times view

systabnames Table information table

sysvpprof Current VP profile view

Differences From Other Databases

There are several key differences between the sysmaster database and other databases you might create. Reminder that this is a database that points to the server's shared memory structures and not to tables that are stored on disk. Some of the differences are:

* You cannot update the sysmaster database. Its purpose is to allow you to read information about the server. Trying to update its tables should generate an error message but may corrupt the server.
* You cannot run dbschema on these table to get their structure. This will generate and error message.
* You cannot drop the sysmaster database or any tables within it. Again, this should generate an error message.
* The data is dynamic and may change while you are retrieving it. The sysmaster database has an effective isolation level of dirty read even though it looks like a database with unbuffered logging. This prevents your queries from locking users and slowing down their processing.
* However, because the sysmaster database uses unbuffered logging, its temp tables are logged.
* You can create triggers and stored procedures on the sysmaster database, but the triggers will never be executed. Again, this is because this is not a real database but pointers to shared memory.

The sysmaster database reads the same shared memory structures read by the command line utility "onstat". The statistical data is reset to zero when OnLine is shut down and restarted.

It is also reset to zero when the "onstat -z" command to reset statistics is used. Individual user statistical data is lost when a user disconnects from the server.

Now, let's examine some of the more interesting tables in the sysmaster database and what else can be done with them.

3. Server Information

This first section will look at how you determine the state and configuration of your INFORMIX-OnLine server from the sysmaster database. We will look at four tables and how to use them.

Server configuration and statistics tables:

* sysconfig - ONCONFIG File
* sysprofile - Server Statistics
* syslogs - Logical Logs
* sysvpprof - Virtual Processors

Server Configuration Parameters: sysconfig

The view sysonfig contains configuration information from the OnLine server. This information was read from the ONCONFIG file when the server was started. Have you ever needed to know from within a program how your server was setup? Or, what TAPEDEV is set to?

View sysconfig

Column Data Type Description

cf\_id integer unique numeric identifier

cf\_name char(18) config parameter name

cf\_flags integer flags, 0 = in view sysconfig

cf\_original char(256) value in ONCONFIG at boottime

cf\_effective char(256) value effectively in use

cf\_default char(256) value by default

*Example queries:*

To find out what the current tape device is:

select cf\_effective from sysconfig where cf\_name = "TAPEDEV";

To find the server name:

select cf\_effective from sysconfig where cf\_name =

"DBSERVERNAME";

To find out if data replication is turned on:

select cf\_effective from sysconfig where cf\_name = "DRAUTO";

Server Profile Information: sysprofile

The sysprofile table is a view based on values in a table called syshmhdr. Syshmhdr points to the same shared memory area as the onstat utility with the -p option. When you zero out the statistics with "onstat -z", all values in the syshmhdr table are reset to zero.

View sysprofile

Column Data Type Description

name char(16) profile element name

value integer current value

One of the best uses of this data is for developing alarms when certain values fall below acceptable levels. The Informix documentation says that tables in the sysmaster database do not run triggers. This is because the updates to these tables take place within OnLine shared memory and not through SQL which activates triggers. However, you can create a program to poll this table at specified intervals to select data and see if it falls below your expectations.

Logical Logs Information: syslogs

Syslogs is a view based on the table syslogfil. This is an example where the SMI views are a great tool in presenting the data in a more understandable format. Syslogfil has a field called flags which contains status information encoded in boolean smallint. The view syslogs decodes that data into six fields: is\_used, is\_current, is\_backed\_up, is\_new, is\_archived, and is\_temp, with a 1 if true or a 0 if false.

View syslogs

Column Data Type Description

number smallint logfile number

uniqid integer logfile uniqid

size integer pages in logfile

used integer pages used in logfile

is\_used integer 1 for used, 0 for free

is\_current integer 1 for current

is\_backed\_up integer 1 for backuped

is\_new integer 1 for new

is\_archived integer 1 for archived

is\_temp integer 1 for temp

flags smallint logfile flags

Virtual Processor Information and Statistics: sysvpprof

Sysvpprof is another view that is more readable than the underlying table sysvplst. As with the view syslogs in the above paragraph, this view has data that is converted to make it more understandable. This time the flags are converted to text descriptions from the flags\_text table.

View sysvpprof

Column Data Type Description

vpid integer VP id

txt char(50) VP class name

usecs\_user float number of unix secs of user time

usecs\_sys float number of unix secs of system time

The following query on the base table sysvplst achieves the same results as the view.

Figure 6. SQL script to display VP status

-- vpstat.sql

select vpid,

txt[1,5] class,

pid,

usecs\_user,

usecs\_sys,

num\_ready

from sysvplst a, flags\_text b

where a.flags != 6

and a.class = b.flags

and b.tabname = 'sysvplst';

SQL Output

vpid class pid usecs\_user usecs\_sys num\_ready

1 cpu 335 793.61 30.46 0

2 adm 336 0.02 0.11 0

3 lio 337 1.15 5.98 0

4 pio 338 0.19 1.13 0

5 aio 339 0.94 4.27 0

6 msc 340 0.15 0.14 0

7 aio 341 0.81 5.72 0

8 tli 342 1.79 3.02 0

9 aio 343 0.52 2.50 0

10 aio 344 0.28 1.16 0

11 aio 345 0.09 0.86 0

12 aio 346 0.16 0.48 0

4. Dbspace and Chunk Information

Now let's look at the SMI tables that contain information about your disk space, chunks, and dbspace. There are four tables that contain this data.

* sysdbspaces - DB Spaces
* syschunks - Chunks
* syschkio - I/O by Chunk
* syschfree\* - Free Space by Chunk

\* Note: Syschfree is not a supported table.

Dbspace Configuration: sysdbspaces

The sysmaster database has three key tables containing dbspace and chunk information. The first one is sysdbspaces. This is a view that interprets the underlying table sysdbstab. Sysdbspaces serves two purposes: it translates a bit field containing flags into separate columns where 1 equals yes and 0 equals no, and, it allows the underlying table to change between releases without having to change code. The view is defined as follows:

View sysdbspaces

Column Data Type Description

dbsnum smallint dbspace number,

name char(18) dbspace name,

owner char(8) dbspace owner,

fchunk smallint first chunk in dbspace,

nchunks smallint number of chunks in dbspace,

is\_mirrored bitval is dbspace mirrored, 1=Yes, 0=No

is\_blobspace bitval is dbspace a blob space, 1=Yes, 2=No

is\_temp bitval is dbspace temp, 1=Yes, 2=No

flags smallint dbspace flags

The columns of type bitval are the flags that are extracted from the flags column by a stored procedure called bitval when the view is generated.

Chunk Configuration: syschunks

The syschunks table is also a view based on two actual tables, one for primary chunk information, syschktab, and one for mirror chunk information, sysmchktab. The following is the layout of syschunks:

View syschunks

Column Data Type Description

chknum smallint chunk number

dbsnum smallint dbspace number

nxchknum smallint number of next chunk in dbspace

chksize integer pages in chunk

offset integer pages offset into device

nfree integer free pages in chunk

is\_offline bitval is chunk offline, 1=Yes, 0=No

is\_recovering bitval is chunk recovering, 1=Yes, 0=No

is\_blobchunk bitval is chunk blobchunk, 1=Yes, 0=No

is\_inconsistent bitval is chunk inconsistent, 1=Yes, 0=No

flags smallint chunk flags converted by bitval

fname char(128) device pathname

mfname char(128) mirror device pathname

moffset integer pages offset into mirror device

mis\_offline bitval is mirror offline, 1=Yes, 0=No

mis\_recovering bitval is mirror recovering, 1=Yes, 0=No

mflags smallint mirror chunk flags

Displaying Free Dbspace

Now, we will take a look at several ways to use this dbspace and chunk information. One capability I have always wanted is a way to show the amount of dbspace used and free in the same format as the Unix "df -k" command. The sysmaster database contains information about the dbspaces and chunks, so this can be generated with an SQL script. The following is an SQL script to generate the amount of free space in a dbspace. It uses the sysdbspaces and syschunks tables to collect its information.

Figure 7. SQL script to display free dbspace

-- dbsfree.sql - display free dbspace like Unix "df -k " command

database sysmaster;

select name[1,8] dbspace, -- name truncated to fit on one line

sum(chksize) Pages\_size, -- sum of all chunks size pages

sum(chksize) - sum(nfree) Pages\_used,

sum(nfree) Pages\_free, -- sum of all chunks free pages

round ((sum(nfree)) / (sum(chksize)) \* 100, 2) percent\_free

from sysdbspaces d, syschunks c

where d.dbsnum = c.dbsnum

group by 1

order by 1;

Sample output

dbspace pages\_size pages\_used pages\_free percent\_free

rootdbs 50000 13521 36479 72.96

dbspace1 100000 87532 12468 12.47

dbspace2 100000 62876 37124 37.12

dbspace3 100000 201 99799 99.80

Displaying Chunk Status

The next script lists the status and characteristics of each chunk device.

Figure 8. SQL script showing chunk status

-- chkstatus.sql - display information about a chunk

database sysmaster;

select

name dbspace, -- dbspace name

is\_mirrored, -- dbspace is mirrored 1=Yes 0=No

is\_blobspace, -- dbspace is blobspace 1=Yes 0=No

is\_temp, -- dbspace is temp 1=Yes 0=No

chknum chunknum, -- chunk number

fname device, -- dev path

offset dev\_offset, -- dev offset

is\_offline, -- Offline 1=Yes 0=No

is\_recovering, -- Recovering 1=Yes 0=No

is\_blobchunk, -- Blobspace 1=Yes 0=No

is\_inconsistent, -- Inconsistent 1=Yes 0=No

chksize Pages\_size, -- chunk size in pages

(chksize - nfree) Pages\_used, -- chunk pages used

nfree Pages\_free, -- chunk free pages

round ((nfree / chksize) \* 100, 2) percent\_free, -- free

mfname mirror\_device, -- mirror dev path

moffset mirror\_offset, -- mirror dev offset

mis\_offline , -- mirror offline 1=Yes 0=No

mis\_recovering -- mirror recovering 1=Yes 0=No

from sysdbspaces d, syschunks c

where d.dbsnum = c.dbsnum

order by dbspace, chunknum

Displaying Blocks of Free Space in a Chunk: syscchfree

In planning expansions, new databases, or when adding new tables to an existing server, I like to know what blocks of contiguous free space are available. This allows placing new tables in dbspaces where they will not be broken up by extents. One of the sysmaster tables tracks the chunk free list, which is the available space in a chunk.

Table syschfree

Column Data Type Description

chknum integer chunk number

extnum integer extent number in chunk

start integer physical addr of start

leng integer length of extent

The next script uses this table to create a list of free space and the size of each space that is available.

Figure 9. SQL script showing free space on chunks

-- chkflist.sql - display list of free space within a chunk

database sysmaster;

select

name dbspace, -- dbspace name truncated to fit

f.chknum, -- chunk number

f.extnum, -- extent number of free space

f.start, -- starting address of free space

f.leng free\_pages -- length of free space

from sysdbspaces d, syschunks c, syschfree f

where d.dbsnum = c.dbsnum

and c.chknum = f.chknum

order by dbspace, chknum

Sample Output

dbspace chknum extnum start free\_pages

rootdbs 1 0 11905 1608

rootdbs 1 1 15129 34871

IO Statistics by Chunk Devices: syschkio

Informix uses a view, syschkio, to collect information about the number of disk reads and writes per chunk. This view is based on the tables syschktab and symchktab.

View syschkio

Column Data Type Description

chunknum smallint chunk number

reads integer number of read ops

pagesread integer number of pages read

writes integer number of write ops

pageswritten integer number of pages written

mreads integer number of mirror read ops

mpagesread integer number of mirror pages read

mwrites integer number of mirror write ops

mpageswritten integer number of mirror pages written

The following script displays IO usage of chunk devices. It uses the base tables so the mirror chunks can be displayed on separate rows. It also joins with the base table that contains the dbspace name.

Figure 10. SQL script displaying chunk I/O

-- chkio.sql - displays chunk IO status

database sysmaster;

select

name[1,10] dbspace, -- truncated to fit 80 char screen line

chknum,

"Primary" chktype,

reads,

writes,

pagesread,

pageswritten

from syschktab c, sysdbstab d

where c.dbsnum = d.dbsnum

union all

select

name[1,10] dbspace,

chknum,

"Mirror" chktype,

reads,

writes,

pagesread,

pageswritten

from sysmchktab c, sysdbstab d

where c.dbsnum = d.dbsnum

order by 1,2,3;

Sample Output

dbspace chknum chktype reads writes pagesread pageswritten

rootdbs 1 Primary 74209 165064 209177 308004

rootdbs 1 Mirror 69401 159832 209018 307985

A better view of your IO is to see the percent of the total IO that takes place per chunk. This next query collects IO stats into a temp table, and then uses that to calculate total IO stats for all chunks. Then each chunk's IO is compared with the total to determine the percent of IO by chunk. The following script uses the one above as a basis to show IO by chunk as a percent of the total IO.

Figure 11. SQL script chunk I/O summary

-- chkiosum.sql - calculates percent of IO by chunk

database sysmaster;

-- Collect chunk IO stats into temp table A

select

name dbspace,

chknum,

"Primary" chktype,

reads,

writes,

pagesread,

pageswritten

from syschktab c, sysdbstab d

where c.dbsnum = d.dbsnum

union all

select

name[1,10] dbspace,

chknum,

"Mirror" chktype,

reads,

writes,

pagesread,

pageswritten

from sysmchktab c, sysdbstab d

where c.dbsnum = d.dbsnum

into temp A;

-- Collect total IO stats into temp table B

select

sum(reads) total\_reads,

sum(writes) total\_writes,

sum(pagesread) total\_pgreads,

sum(pageswritten) total\_pgwrites

from A

into temp B;

-- Report showing each chunks percent of total IO

select

dbspace,

chknum,

chktype,

reads,

writes,

pagesread,

pageswritten,

round((reads/total\_reads) \*100, 2) percent\_reads,

round((writes/total\_writes) \*100, 2) percent\_writes,

round((pagesread/total\_pgreads) \*100, 2) percent\_pg\_reads,

round((pageswritten/total\_pgwrites) \*100, 2) percent\_pg\_writes

from A, B

order by 11;-- order by percent page writes

Sample output for 1 chunk

dbspace datadbs

chknum 9

chktype Primary

reads 12001

writes 9804

pagesread 23894

pageswritten 14584

percent\_reads 0.33

percent\_writes 0.75

percent\_pg\_reads 37.59

percent\_pg\_writes 1.86

5. Database and Table Information

The next five tables we will look at store information on your tables and extents. They are:

* sysdatabases - Databases
* systabnames - Tables
* sysextents - Tables extents
* sysptprof - Tables I/O

Information on All Databases on a Server: sysdatabases

This view has data on all databases on a server. Have you ever needed to create a pop-up list of databases within a program? This table now allows programs to give users a list of databases to select from without resorting to ESQL/C. The following is the definition of this view:

View sysdatabases

Column Data Type Description

name char(18) database name

partnum integer table id for systables

owner char(8) user name of creator

created integer date created

is\_logging bitval unbuffered logging, 1=Yes, 0= No

is\_buff\_log bitval buffered logging, 1=Yes, 0= No

is\_ansi bitval ANSI mode database, 1=Yes, 0= No

is\_nls bitval NLS support, 1=Yes, 0= No

flags smallint flags indicating logging

The following is a script to list all databases, owners, dbspaces, and logging status. Notice the function dbinfo is used. This is a new function in 7.X with several uses, one of which is to convert the partnum of a database into its corresponding dbspace. This function will be used in several examples that follow.

Figure 12. SQL script listing all databases on the server

-- dblist.sql - List all databases, owner and logging status

database sysmaster;

select

dbinfo("DBSPACE",partnum) dbspace,

name database,

owner,

is\_logging,

is\_buff\_log

from sysdatabases

order by dbspace, name;

Sample Output

dbspace database owner is\_logging is\_buff\_log

rootdbs central lester 0 0

rootdbs datatools lester 0 0

rootdbs dba lester 0 0

rootdbs roster lester 0 0

rootdbs stores7 lester 0 0

rootdbs sunset linda 0 0

rootdbs sysmaster informix 1 0

rootdbs zip lester 1 1

Information About Database Tables: systabnames, sysextents, and sysptprof

Three tables contain all the data you need from the sysmaster database about tables in your database. The first of these is a real table defined as follows:

Table systabnames - All tables on the server

Column Data Type Description

partnum integer table id for table

dbsname char(18) database name

owner char(8) table owner

tabname char(18) table name

collate char(32) collation assoc with NLS DB

View sysextents - Tables and each extent on the server

Column Data Type Description

dbsname char(18) database name

tabname char(18) table name

start integer physical addr for this extent

size integer size of this extent

The view sysextents is based on a table, sysptnext, defined as follows:

Table sysptnext

Column Data Type Description

pe\_partnum integer partnum for this partition

pe\_extnum smallint extent number

pe\_phys integer physical addr for this extent

pe\_size integer size of this extent

pe\_log integer logical page for start

View sysptprof - Tables IO profile

Column Data Type Description

dbsname char(18) database name

tabname char(18) table name

partnum integer partnum for this table

lockreqs integer lock requests

lockwts integer lock waits

deadlks integer deadlocks

lktouts integer lock timeouts

isreads integer reads

iswrites integer writes

isrewrites integer rewrites

isdeletes integer deletes

bufreads integer buffer reads

bufwrites integer buffer writes

seqscans integer sequential scans

pagreads integer disk reads

pagwrites integer disk writes

These tables allow us to develop scripts to display tables, the number of extents, and pages used. We can also present a layout of dbspace, databases, tables, and extents similar to the command "tbcheck -pe". And finally, we can show table usage statistics sorted by which tables have the most hits based on reads, writes, or locks. These scripts will enable a DBA to monitor and tune the database server.

Extents are created when a table's initial space has been filled up and it needs more space. OnLine will allocate additional space for a table. However, the table will no longer be contiguous, and performance will start to degrade. Informix will display warning messages when a table reaches more than 8 extents. Depending on a number of factors, at approximately 180-230 extents a table will not be able to expand and no additional rows can be inserted. The following script lists all tables sorted by the number of extents. The tables that show up with many extents may need to be unloaded and rebuilt.

Figure 13. SQL script showing tables and extents

-- tabextent.sql - List tables, number of extents and size of table.

database sysmaster;

select dbsname,

tabname,

count(\*) num\_of\_extents,

sum( pe\_size ) total\_size

from systabnames, sysptnext

where partnum = pe\_partnum

group by 1, 2

order by 3 desc, 4 desc;

Sample Output

dbsname tabname num\_of\_extents total\_size

rootdbs TBLSpace 8 400

sysmaster syscolumns 6 56

sunset inventory 3 376

sunset sales\_items 3 96

sunset sales\_header 3 48

sunset parts 3 48

sunset customer 3 40

sunset syscolumnext 3 32

sunset employee 3 32

Sometimes it is helpful to see how the tables are interspersed on disk. The following script lists by dbspace each table and the location of each extent. This is similar to the output from "oncheck -pe".

Figure 14. SQL script showing table layout on chunks

-- tablayout.sql - Show layout of tables and extents

database sysmaster;

select dbinfo( "DBSPACE" , pe\_partnum ) dbspace,

dbsname[1,10],

tabname,

pe\_phys start,

pe\_size size

from sysptnext, outer systabnames

where pe\_partnum = partnum

order by dbspace, start;

Sample output

dbspace dbsname tabname start size

rootdbs rootdbs TBLSpace 1048589 50

rootdbs sysmaster sysdatabases 1050639 4

rootdbs sysmaster systables 1050643 8

rootdbs sysmaster syscolumns 1050651 16

rootdbs sysmaster sysindexes 1050667 8

rootdbs sysmaster systabauth 1050675 8

rootdbs sysmaster syscolauth 1050683 8

rootdbs sysmaster sysviews 1050691 8

rootdbs sysmaster sysusers 1050699 8

rootdbs sysmaster sysdepend 1050707 8

rootdbs sysmaster syssynonyms 1050715 8

IO Performance of Tables

Have you ever wanted to know which tables have the most reads, writes, or locks? The last script in this article shows the performance profile of tables. By changing the columns displayed and the sort order of the script, you can display the tables with the most reads, writes, or locks first.

Figure 15. SQL script show table I/O activity

-- tabprof.sql

database sysmaster;

select

dbsname,

tabname,

isreads,

-- pagwrites

-- uncomment the following to show locks

-- lockreqs,

-- lockwts,

-- deadlks

from sysptprof

order by isreads desc; -- change this sort to whatever you need to monitor.

Sample Output

dbsname tabname isreads bufreads pagreads

zip zip 334175 35876509 1111

sysmaster sysviews 259712 634102 1119

sysmaster systables 60999 240018 1878

zip systables 3491 8228 543

sysmaster sysusers 2406 8936 87

sysmaster sysprocauth 1276 5104 12

sunset systables 705 2251 26

sysmaster sysprocedures 640 2562 21

sysmaster syscolumns 637 1512 49

stores7 systables 565 1361 16

sysmaster sysdatabases 534 2073 902

bufreads,

pagreads

-- uncomment the following to show writes

-- iswrites,

-- bufwrites,

6. User Session Information

This last set of SMI tables deals with users and information about their sessions. These tables were used in our example script "dbwho" at the beginning of this chapter.

* syssessions - Session data
* syssesprof - User statistics
* syslocks - User Locks
* syseswts - Wait times

User Session and Connection Information: syssessions

This view contains information from two shared memory structures, the user control and thread control table. This tells you who is logged in to your server and some basic data about their session.

View syssessions

Column Data Type Description

sid integer Session id number

username char(8) User name

uid smallint User unix id

pid integer User process id

hostname char(16) Hostname

tty char(16) TTY port

connected integer Time user connected

feprogram char(16) Program name

pooladdr integer Pointer to private session pool

is\_wlatch integer Flag 1=Yes, 0=No, wait on latch

is\_wlock integer Flag 1=Yes, 0=No, wait on lock

is\_wbuff integer Flag 1=Yes, 0=No, wait on buffer

is\_wckpt integer Flag 1=Yes, 0=No, wait on checkpoint

is\_wlogbuf integer Flag 1=Yes, 0=No, wait on log buffer

is\_wtrans integer Flag 1=Yes, 0=No, wait on a transaction

is\_monitor integer Flag 1=Yes, 0=No, a monitoring process

is\_incrit integer Flag 1=Yes, 0=No, in crtical section

state integer Flags

The following is a quick query to tell who is using your server.

Figure 16. SQL script showing user sessions

-- sessions.sql

select sid,

username,

pid,

hostname,

l2date(connected) startdate -- convert unix time to date

from syssessions

Sample Output

sid username pid hostname startdate

47 lester 11564 merlin 07/14/1997

This next query list all users and their session status. The objective is to show who is blocked waiting on another user, lock, or some other OnLine process. The five fields are yes/no flags where 1 = yes and 0 = no. If all the fields are 0, then none of the sessions are blocked. In the following example, one session is blocked waiting on a locked record.

Figure 17. SQL script users waiting on resources

-- seswait.sql

select username,

is\_wlatch, -- blocked waiting on a latch

is\_wlock, -- blocked waiting on a locked record or table

is\_wbuff, -- blocked waiting on a buffer

is\_wckpt, -- blocked waiting on a checkpoint

is\_incrit -- session is in a critical section of transaction

-- (e.g writting to disk)

from syssessions

order by username;

Sample Output

username is\_wlatch is\_wlock is\_wbuff is\_wckpt is\_incrit

lester 0 1 0 0 0

lester 0 0 0 0 0

lester 0 0 0 0 0

User Session Performance Statistics: syssesprof

This view syssesprof provides a way to find out at a given point in time how much of your server resources each user is using. The view contains the following information.

View syssesprof

Column Data Type Description

sid integer, Session Id

lockreqs decimal(16,0) Locks requested

locksheld decimal(16,0) Locks held

lockwts decimal(16,0) Locks waits

deadlks decimal(16,0) Deadlocks detected

lktouts decimal(16,0) Deadlock timeouts

logrecs decimal(16,0) Logical Log records written

isreads decimal(16,0) Reads

iswrites decimal(16,0) Writes

isrewrites decimal(16,0) Rewrites

isdeletes decimal(16,0) Deletes

iscommits decimal(16,0) Commits

isrollbacks decimal(16,0) Rollbacks

longtxs decimal(16,0) Long transactions

bufreads decimal(16,0) Buffer reads

bufwrites decimal(16,0) Buffer writes

seqscans decimal(16,0) Sequential scans

pagreads decimal(16,0) Page reads

pagwrites decimal(16,0) Page writes

total\_sorts decimal(16,0) Total sorts

dsksorts decimal(16,0) Sorts to disk

max\_sortdiskspace decimal(16,0) Max space used by a sort

logspused decimal(16,0) Current log bytes used

maxlogsp decimal(16,0) Max bytes of logical logs used

This table contains data since the user logged on. Each time a user disconnects their data is lost so you cannot use this data for charging the user for server usage. Also, when a DBA resets the server statistics with the command "tbstat -z", all profile data is reset to zero.

I like to monitor the number of locks used by each user and their buffer usage. The following is an example query.

Figure 19. SQL script to monitor resource usage by user

-- sesprof.sql

select username,

syssesprof.sid,

lockreqs,

bufreads,

bufwrites

from syssesprof, syssessions

where syssesprof.sid = syssessions.sid

order by bufreads desc

Active Locks on the Server: syslocks

This view contains information about all active locks on your server. It can be very large; if you have a lot of users and your server is configured to handle a large number of locks, you could end up with hundreds of thousands or more records in this view. This view is composed of six tables, and queries on this view will create a temp table which is logged to your logical log. The performance may be a bit slow because of the sheer volume of data produced by this view. However, the data this view contains can be very helpful to understanding how your system is performing.

View syslocks

Column Data Type Description

dbsname char(18) Database name

tabname char(18) Table name

rowidlk integer Rowid for index key lock

keynum smallint Key number of index key lock

owner integer Session ID of lock owner

waiter integer Session ID of first waiter

type char(4) Type of Lock

Types of Locks

B - byte lock

IS - intent shared lock

S - shared lock

XS - repeatable read shared key

U - update lock

IX - intent exclusive lock

SIX - shared intent exclusive

X - exclusive lock

XR - repeatable read exclusive

Basically there are three types of locks: a shared lock (S), an exclusive lock (X), and an update lock(U). A shared lock allows other users to also read the data but none may change it. An exclusive lock does not allow anyone else to lock that data even in shared mode. An update lock prevents other users from changing data while you are changing it.

There are six objects that can be locked in OnLine.

* Database - Every user that opens a database places a shared lock on the database to prevent someone else from dropping the database while it is in use. This shows up as a lock on the sysmaster database and the sysdatabase tables, and the rowid will point to the record containing database name.
* Table - A table lock shows up as a lock on a table with a rowid of 0 and a keynum of 0.
* Page - A page level lock shows as a rowid ending in 00. This means all the rows on that page are locked.
* Row - A row level lock will show with an actual rowid (not ending in 00).
* Key - A key lock will show with a keynum. If a row has indexes that need to be updated this will place locks on the indexes for that row.

One of the key data elements missing from this view is the username and session id (sid) of the user who has a lock. The following query adds the user's name and session id and uses the underlying tables to improve performance. It also puts the data into a temp table from which you can select subsets of data much more quickly than if you were to repeat the query.

Figure 20. SQL script to show all locks

-- locks.sql

select dbsname,

b.tabname,

rowidr,

keynum,

e.txt type,

d.sid owner,

g.username ownername,

f.sid waiter,

h.username waitname

from syslcktab a,

systabnames b,

systxptab c,

sysrstcb d,

sysscblst g,

flags\_text e,

outer ( sysrstcb f , sysscblst h )

where a.partnum = b.partnum

and a.owner = c.address

and c.owner = d.address

and a.wtlist = f.address

and d.sid = g.sid

and e.tabname = 'syslcktab'

and e.flags = a.type

and f.sid = h.sid

into temp A;

select dbsname,

tabname,

rowidr,

keynum,

type[1,4],

owner,

ownername ,

waiter,

waitname

from A;

Example SQL Output

dbsname sysmaster

tabname a

rowidr 0

keynum 0

type X

owner 47

ownername lester

waiter

waitname

The above example SQL output shows the row from syslocks that displays the exclusive lock I created on the temp table "A" while running the query.

A more important use of this query is to find out when one user is waiting on the lock owned by another user. When a user has a database object locked, the first user waiting on the object can be displayed. (This will only occur when a user has set lock mode to WAIT). The following script displays only the users that have locks where someone else is waiting on their process. There is one key difference between this script and the one above. The tables sysrstcb and sysscblst in this script do not use an outer join, so only rows that have waiters will be returned. In this example "linda" has an update lock on a row and "lester" is waiting for that update to complete.

Figure 21. SQL script to show users waiting on locks

-- lockwaits.sql

database sysmaster;

select dbsname,

b.tabname,

rowidr,

keynum,

e.txt type,

d.sid owner,

g.username ownername,

f.sid waiter,

h.username waitname

from syslcktab a,

systabnames b,

systxptab c,

sysrstcb d,

sysscblst g,

flags\_text e,

sysrstcb f , sysscblst h

where a.partnum = b.partnum

and a.owner = c.address

and c.owner = d.address

and a.wtlist = f.address

and d.sid = g.sid

and e.tabname = 'syslcktab'

and e.flags = a.type

and f.sid = h.sid

into temp A;

select dbsname,

tabname,

type[1,4],

owner,

ownername ,

waitname

from A;

SQL Output

dbsname tabname type owner ownername waitname

stores7 items U 29 linda lester

Wait Status and Times on Objects: sysseswts

This is a supported view that shows all sessions that are blocked and waiting on a database object. It shows the amount of time a user has been waiting. On a well tuned system this table should be empty. However, when the table is not empty, it provides useful information on what is causing your performance to slow down.

View sysseswts

Column Data Type Description

sid integer Session ID

reason char(50) Description of reason for wait

numwaits integer Number of waits for this reason

cumtime float Cumulative wait time for this reason

maxtime integer Max wait time for this reason

7. Some Unsupported Extras

Several of the SMI tables are not documented and not officially supported. These could change in future releases. Two additional unsupported tables I have found helpful are systrans and syssqexplain.

User Transactions: systrans

Three of the fields in systrans are very helpful to determine what logical log number a transaction began in, and the current logical log number in use by a transaction.

Key systrans fields

Column Data Type Description

tx\_id integer pointer to transaction table

tx\_logbeg integer transaction starting logical log

tx\_loguniq integer transaction current logical log number

This can be used to create a script to determine what logical log files have active transactions. The output of this will tell you what logical logs are free and available for reuse. This first script lists all user transactions and what logs they are using.

Figure 22. SQL script to display transactions and logs used

-- txlogpos.sql

select

t.username,

t.sid,

tx\_logbeg,

tx\_loguniq,

tx\_logpos

from systrans x, sysrstcb t

where tx\_owner = t.address

SQL Output

username sid tx\_logbeg tx\_loguniq tx\_logpos

informix 1 0 16 892952

informix 0 0 0 0

informix 8 0 0 0

lester 53 0 0 0

informix 12 0 0 0

lester 51 14 16 0

This shows that my logical logs numbered 14 to 16 are in use by transactions.

Another helpful use of this view is to summarize the transactions by logical logs. This next script show my transaction status by logical log.

Figure 23. SQL script to view logical logs status

-- logstat.sql

database sysmaster;

-- select transaction data into a temp table

select tx\_logbeg, tx\_loguniq

from systrans

into temp b;

-- count how may transactions begin in each log

select tx\_logbeg, count(\*) cnt

from B

where tx\_logbeg > 0

group by tx\_logbeg

into temp C;

-- count how many transactions currently are in each log

select tx\_loguniq, count(\*) cnt

from B

where tx\_loguniq > 0

group by tx\_loguniq

into temp D;

-- join data from counts with syslogs

select

uniqid,

size,

is\_backed\_up, -- 0 = no, 1 = yes log is backed up

is\_archived, -- 0 = no, 1 = yes log is on last archive

c.cnt tx\_beg\_cnt,

d.cnt tx\_curr\_cnt

from syslogs, outer c, outer D

where uniqid = c.tx\_logbeg

and uniqid = d.tx\_loguniq

order by uniqid

SQL Output

uniqid size is\_backed\_up is\_archived tx\_beg\_cnt tx\_curr\_cnt

10 500 1 1

11 500 1 1

12 500 1 1

13 500 1 1

14 500 1 1

15 500 1 1

16 500 0 1 1 2

This shows that all logs are backed up except the current one, and it has two active transactions.

User Queries: syssqexplain

Have you ever wanted to run a query to see what your users were doing? The view syssqexplain contains some of the data from a user's session, including the sql that they are currently executing. Try this query on your system sometime to see your user's SQL.

Figure 24. SQL to view current executing SQL

-- syssql.sql

select username,

sqx\_sessionid,

sqx\_conbno,

sqx\_sqlstatement

from syssqexplain, sysscblst

where sqx\_sessionid = sid

SQL Output

username lester

sqx\_sessionid 55

sqx\_conbno 2

sqx\_sqlstatement select username,sqx\_sessionid, sqx\_conbno, sqx\_sqlstatement

from syssqexplain, sysscblst

where sqx\_sessionid = sid

username lester

sqx\_sessionid 51

sqx\_conbno 0

sqx\_sqlstatement update items set total\_price = 300 where item\_num = 1

Conclusion

The sysmaster database is a great tool for a DBA to monitor the Informix server. If you have any questions or suggestions please send me E-mail at [lester@advancedatatools.com](mailto:lester@advancedatatools.com). Also, if you have any creative scripts for monitoring your server with the sysmaster database, please send them in and I may include them in the future publications