April 23-27, 2017
Raleigh, NC, USA

Creating Custom OAT Monitoring and Reports

C01 – April 24 - 9:45am

Jeff Filippi

Integrated Data Consulting, LLC

jeff.filippi@itdataconsulting.com

Introduction

- 26 years of working with Informix products
- 22 years as an Informix DBA
- Worked for Informix for 5 years 1996 – 2001
- Certified Informix DBA
- Started my own company in 2001 specializing in Informix Database Administration consulting
- IBM Business Partner
- OLTP and Data warehouse systems
- Informix 4, 5, 7, 9, 10, 11.10, 11.50, 11.70, 12.10

Topics

- Show how to create customized monitoring and reports in OAT.
- Demonstrate using the custom monitoring and reports that I have created for clients.
- Show how to add alerts to Health Advisor.

Create Custom Monitoring & Reports

- OAT allows you to create your own monitoring and reports for customized ways you want to see the view of your system.
- I started creating some custom reports for items that were not readily available in OAT.
- It is a great way to be able to extend the functionality of OAT.

Creating Monitoring

- Create a table in the sysadmin database where you want to store the information.
- Create a script/store procedure/etc that will populate your table with the data.
- Create a new record in the “ph_task” table to add the new task with items like when it is to run, frequency, purging, etc.
- Set the new task as enabled.

Examples of Custom Tasks

Below are examples of custom tasks I have created to enhance the monitoring of Informix through OAT.

- Kill user session using too many locks
- Send an alert if a chunk was extended
- Send an alert if the backup has not occurred in a number of days
- Check days since last backup
- Monitor number of sessions for a user
- Monitor number of locks
- Monitor SAR information
- Monitor space used for file system
- Monitor SQL Operations per second
- Monitor Checkpoint Times
- Save SQL Trace (Run time greater than X seconds)

Kill User Session using too many Locks

- Here is a task that will check the user sessions to see if they are using too many locks on a system and to kill the session so that it does not cause an issue with the system.
- Set the task to run every 30 seconds to check if the number of locks for a user session exceeded the threshold.

Kill User Session using too many Locks

create function kill_session_highlocks(task_id INTEGER, task_seq INTEGER) RETURNING INTEGER

```
define locks_allowed INTEGER;
define sys_hostname CHAR(256);
define sys_username CHAR(32);
define sys_sid      INTEGER;
define rc           INTEGER;

SELECT value::integer
INTO locks_allowed
FROM ph_threshold
WHERE name = "KILL HIGH LOCKS";
```

Kill User Session using too many Locks

```
FOREACH highlocks_cur for
  select admin("onmode","z",A.sid), A.username, A.sid, hostname
  into rc, sys_username, sys_sid, sys_hostname
  from sysmaster:sysrstcb A, sysmaster:systcblst B, sysmaster:sysscblst C
  where A.tid = B.tid
    and C.sid = A.sid
    and lower(name) in ("sqlexec")
    and nlocks > locks_allowed

  IF rc > 0 THEN
    INSERT INTO ph_alert (ID, alert_task_id, alert_task_seq, alert_type, alert_color, alert_state,
                          alert_object_type, alert_object_name, alert_message, alert_action)
    VALUES (0, task_id, task_seq, "WARNING", "YELLOW", "NEW", "USER", "TIMEOUT", "User
"||TRIM(sys_username)||"@"||TRIM(sys_hostname)||"sid("||sys_sid||")"||" terminated due
to large number of locks.",NULL);
  END IF
end foreach;
  RETURN 0;
end function;
```

Kill User Session using too many Locks

unl_kill.unl

0|kill_session_highlocks|Terminate sessions with high number of
locks.|TASK|0|||sysadmin|kill_session_highlocks| 30 00:00:00|00:00:00|| 0 00:00:30|2012-09-12 16:00:00|0|0|t|t|t|t|t|t|t|4|SERVER|t|0|

load from unl_kill.unl insert into ph_task;

insert into ph_threshold values (0,'KILL HIGH LOCKS','kill_session_highlocks',500000,'NUMERIC','Kill sessions with high number of locks');

Kill User Session using too many Locks

There is now another option starting with Informix 12.10xC4 where you can limit the number of locks for a session .

- You can prevent users from acquiring too many locks by limiting the number of locks for each user without administrative privileges for a session.
- Set the SESSION_LIMIT_LOCKS configuration parameter or the IFX_SESSION_LIMIT_LOCKS option to the SET ENVIRONMENT statement.

Send Alert if a Chunk was Extended

- Created task that would email the DBA if a chunk was automatically extended due to running out of space.
- This was helpful for the case that there were many instances running an application that the DBA did not actively log in to.
- This gave the DBA a heads up on how much space was being used.

Send Alert if a Chunk was Extended

```
create table alert_chunk_ext
(
last_offset int8
) lock mode row;

insert into alert_chunk_ext values (0);

create function alert_chunk_extend(task_id integer, task_seq integer) returning
integer

define a_offset int8;
define a_last_offset int8;
define a_new_offset int8;
define a_line   char(200);

select last_offset into a_last_offset from alert_chunk_ext;
```

Send Alert if a Chunk was Extended

```
FOREACH alert_cur for
   select next_offset, line[1,200]
   into a_new_offset, a_line
   from sysmaster:sysonlinelog
   where line matches '*Chunk*in*space*has*been*extended*by*'
     and offset >= a_last_offset
   order by offset

   insert into ph_alert(ID,alert_task_id,alert_task_seq,alert_type,alert_color,
                        alert_state,alert_object_type,alert_object_name,alert_message)
   VALUES (0,task_id,task_seq,"INFO","YELLOW","NEW","CHUNK","EXPAND CHUNK",a_line);

   update alert_chunk_ext set last_offset = a_new_offset;

END FOREACH;
RETURN 0;
END FUNCTION;
```

Check Days since Last Backup

- Created a task to alert the DBA if there had not been an Informix level 0 backup for “X” number of days.

Check Days since Last Backup

```
create table check_backup_days
(backup_days integer) lock mode row;

CREATE FUNCTION "informix".check_backup_days(task_id INT, task_seq INT) RETURNING INTEGER
DEFINE dbspace_num INTEGER;
DEFINE dbspace_name CHAR(257);
DEFINE req_level INTEGER;
DEFINE req_level0 INTEGER;
DEFINE level_0 INTEGER;
DEFINE arcdist char(10);
DEFINE arcdist1 INTEGER;

 {*** Select the configuration values ***}
 select value::integer INTO req_level FROM ph_threshold where name   = "REQUIREDLEVEL BACKUP";
 select value::integer INTO req_level0 FROM ph_threshold where name   = "REQUIRED LEVEL 0 BACKUP";

 {*** If not found or are bad values then set better values ***}
 IF req_level < 1 THEN
   LET req_level = 1;
 END IF
 IF req_level0 < 1 THEN
   LET req_level0 = 1;
 END IF
```

Check Days since Last Backup

```
delete from check_backup_days where 1=1;
{*** Check root dbspaces backup time ***}
FOREACH SELECT level0
  INTO level_0
  FROM sysmaster:sysdbstab
  WHERE dbsnum > 0
  AND name = 'rootdbs'
  AND sysmaster:bitval(flags, '0x2000')=0
   AND
   ( ((CURRENT - DBINFO("utc_to_datetime",level0) > req_level units day ) AND
     (CURRENT - DBINFO("utc_to_datetime",level1) > req_level units day ) AND
     (CURRENT - DBINFO("utc_to_datetime",level2) > req_level units day ) )
   OR
    (CURRENT - DBINFO("utc_to_datetime",level0) > req_level0 units day ) )
    LET arcdist = (TODAY - DBINFO("utc_to_datetime",level_0));
    LET arcdist1 = arcdist;
    INSERT INTO check_backup_days (backup_days) VALUES (arcdist1);
 END FOREACH
RETURN 0;
END FUNCTION;
```

Monitor Number of Sessions for a User

- Collect count of how many sessions are running under specific user id's.
- This is then used to create a report graphing the number of sessions for each user id.

Monitor Number of Sessions for a User

```
-- ADD TASK TO COUNT USERS

create table "informix".mon_cnt_users
  (
    id integer,
    time datetime year to minute,
    username varchar(16),
    user_cnt integer

  ) extent size 2000 next size 2000 lock mode row;

revoke all on "informix".mon_cnt_users from "public" as "informix";

grant select on "informix".mon_cnt_users to "public" as "informix";
grant update on "informix".mon_cnt_users to "public" as "informix";
grant insert on "informix".mon_cnt_users to "public" as "informix";
grant delete on "informix".mon_cnt_users to "public" as "informix";
grant index on "informix".mon_cnt_users to "public" as "informix";

load from unl_mon_cnt_users.unl insert into ph_task;
```

Monitor Number of Locks

- Collect the number of locks to be able to report any large changes in number of locks held.
- Created a report to graph the number of locks that were being used on the system

Monitor Number of Locks

```
-- ADD TASK LOCK COUNT
create table "informix".mon_locks
  (
    id integer,
    time datetime year to minute,
    lock_cnt integer

  ) extent size 2000 next size 2000 lock mode row;

revoke all on "informix".mon_locks from "public" as "informix";
grant select on "informix".mon_locks to "public" as "informix";
grant update on "informix".mon_locks to "public" as "informix";
grant insert on "informix".mon_locks to "public" as "informix";
grant delete on "informix".mon_locks to "public" as "informix";
grant index on "informix".mon_locks to "public" as "informix";
```

Monitor Number of Locks

unl_mon_locks.unl

0|mon_locks|Gather count of how many locks used|SENSOR|9829|mon_locks|create table informix.mon_locks (id integer, time datetime year to minute, lock_cnt integer) extent size 2000 next size 2000 lock mode row; grant select on mon_locks to 'db_monitor' as informix;|sysadmin|insert into mon_locks select $DATA_SEQ_ID, CURRENT, count(*) from sysmaster:syslocks| 30 00:00:00|00:00:00|| 0 00:20:00|2014-02-12 14:20:00|9829|86.09572725670445|t|t|t|t|t|t|t|401|PERFORMANCE|t|0|

load from unl_mon_locks.unl insert into ph_task;

Monitor SAR Information

- Created a task and report to monitor the SAR information for the server.
- This would allow the DBA to be able to do quick comparisons of Informix data and SAR data without having to get a separate report from the Unix admin.
- May have to change the stored procedure since different OS's display the information in different formats.

Monitor SAR Information

```
create table mon_sar
(
        sar_id serial,
        sar_time datetime year to second,
        pct_user smallint,
        pct_sys smallint,
        pct_wio smallint,
        pct_idle smallint
) extent size 2000 next size 2000 lock mode row;
create index idx_mon_sar_1 on mon_sar(sar_id,
sar_time);
create index idx_mon_sar_2 on mon_sar(sar_time);
```

Monitor SAR Information

```
create external table tmp_sar
(
sar_output char(79)
)
USING (DATAFILES ("DISK:/prod/bin/dba/sar.unl"),
FORMAT "DELIMITED"
);

create procedure mon_sar_proc()
define tmp_sar_output  char(80);
define tmp_time_sar  char(8);
define tmp_user  smallint;
define tmp_sys  smallint;
define tmp_wio  smallint;
define tmp_idle  smallint;
```

Monitor SAR Information

```
system "sar -u 1 1 |grep : |grep -v sys | grep -v Average |sed 's/$/|/' >
/prod/bin/dba/sar.unl";

foreach sar_cur for select sar_output into tmp_sar_output from tmp_sar
    let tmp_time_sar = tmp_sar_output[1,8];
    let tmp_user = tmp_sar_output[23,29];
    let tmp_sys = tmp_sar_output[43,49];
    let tmp_wio = tmp_sar_output[53,59];
    let tmp_idle = tmp_sar_output[73,79];

    insert into mon_sar values
(0,CURRENT,tmp_user,tmp_sys,tmp_wio,tmp_idle);
end foreach
end procedure;
```

Monitor SAR Information

sar_task.unl

0|mon_sar|Monitor server information from SAR output|SENSOR|19654|mon_sar|create table informix.mon_sar (id integer,sar_id serial not null , sar_time datetime year to second, pct_user smallint, pct_sys smallint, pct_wio smallint, pct_idle smallint) extent size 2000 next size 1000 lock mode row; create index informix.idx_mon_sar_1 on informix.mon_sar (sar_id, sar_time) using btree in sysadmindbs; create index informix.idx_mon_sar_2 on informix.mon_sar (sar_time) using btree in sysadmindbs; grant select on mon_sar to "db_monitor" as informix;|sysadmin|mon_sar_proc| 30 00:00:00|00:00:00|| 0 00:10:00|2014-02-12 14:20:00|19652|20764.198599732768|t|t|t|t|t|t|401|SERVER |t|0|

load from sar_task.unl insert into ph_task;

Monitor Space Used for File System

- Created a task to alert the DBA if a file system that the Informix chunks resided in was filling up.
- This was useful in a case where a customer had many of instances in remote locations where they did not actively manage.
- This would give them a heads up to a need to add more space to the file system.

Monitor Space Used for File System

freespace_task.unl

0|mon_used_file_system|Check percent free for /infx_data file system|TASK|3|||sysadmin|execute procedure mon_used_file_system()| 0 01:00:00|23:50:00|| 1 00:00:00||1|0.0|t|t|t|t|t|t|t|400|SERVER|t|0|

load from freespace_task.unl insert into ph_task;

Monitor Space Used for File System

```
create external table "informix".used_file_system
 (
 used_fs smallint
  )
using
 (datafiles ("DISK:/prod/bin/dba/used_fs.unl“),
format "delimited",
escape on
);

create procedure "informix".mon_used_file_system()
system "df -k /infx_data |grep '\/infx_data' |awk '{print $4}' |sed
's/%/|/' > /prod/bin/dba/used_fs.unl";
end procedure;
```

Monitor SQL Operations Per Second

- Created a task and report to track the number of operations per second for an instance.
- There is a real time option to see this information in OAT, but not one that would save the information for historical viewing.

Monitor SQL Operations Per Second

```
create table mon_oper_per_sec
(
curr_date datetime year to second,
ops integer,
ops_diff integer,
num_sec integer,
ops_psec integer
) extent size 2000 next size 1000 lock mode row;
```

Monitor SQL Operations Per Second

```
create procedure "informix".mon_oper_per_sec_proc()
define s_dt datetime year to second;
define s_ops integer;
define last_ops integer;
define numsec integer;
define last_dt datetime year to second;

   let s_dt = NULL;
   let s_ops = 0;
   let last_ops = 0;
   let last_dt = NULL;
   let numsec = 0;
```

Monitor SQL Operations Per Second

```
  select sum(value) as ops
  into s_ops
  from sysmaster:sysshmhdr
  where number in (53, 54, 55, 56);
  select ops, current year to second as dt, ((current - curr_date)::interval second(9) to second)::char(10)::int
  into last_ops, s_dt, numsec
  from sysadmin:mon_oper_per_sec
  where curr_date = (select max(curr_date) from sysadmin:mon_oper_per_sec);

  if last_ops > s_ops then
     insert into mon_oper_per_sec values (s_dt, s_ops, s_ops, numsec ,(s_ops / numsec));
  else
     insert into mon_oper_per_sec values (s_dt, s_ops, (s_ops - last_ops), numsec, ((s_ops - last_ops) / numsec));
  end if
end procedure;
```

Monitor SQL Operations Per Second

ld_task.unl

0|mon_oper_per_second|Monitor SQL Operations Per Second|SENSOR|11452|mon_oper_per_sec|create table informix.mon_oper_per_sec (id integer, curr_date datetime year to second, ops integer, ops_diff integer, num_sec integer, ops_psec integer) extent size 2000 next size 1000 lock mode row; create index "informix".idx_mon_oper_1 on "informix".mon_oper_per_sec (curr_date) using btree ; grant select on mon_oper_per_sec to "db_monitor" as informix;|sysadmin|mon_oper_per_sec_proc| 30 00:00:00|00:00:00|| 0 00:10:00|2014-02-12 14:20:00|11452|62.158475794845053|t|t|t|t|t|t|401|SERVER |t|0|

load from ld_task.unl insert into ph_task;

Monitor SQL Operations Per Second

Create first row in the table so graph is not skewed.

insert into mon_oper_per_sec values ('2013-11-19 18:25:44',146084526,0,0,0);

Monitor Checkpoint Time

- Created a report to display the checkpoint information over time.
- The existing table "**mon_checkpoints**" is used for the report.

Save SQL Trace

- Created a task to save SQL trace information ONLY if it is greater than “X” seconds.
- We do not want all the SQL traces but just the longer running ones.

Save SQL Trace

```
create raw table "informix".save_sqltrace
  (
    date_time datetime year to second,
    sql_id int8,
    sql_runtime float,
    sql_sid int8,
    sql_uid int8,
    sql_statement char(11000),
    sql_database char(30)
  ) in sqltrace  extent size 99996 next size 99996 lock mode row;

create index "informix".idx_savesql1 on "informix".save_sqltrace
    (date_time) using btree  in sqltrace;
create index "informix".idx_savesql2 on "informix".save_sqltrace
    (sql_runtime) using btree  in sqltrace;
create index "informix".idx_savesql3 on "informix".save_sqltrace
    (sql_id) using btree  in sqltrace;
```

NOTE: Create a new dbspace so that if the table fills up the dbspace it does not affect any other process.

Save SQL Trace

Here is the row to insert into the "ph_task" table.

0|save_trace|Saves SQL Trace when run time greater than set value.|TASK|9251|||sysadmin|insert into save_sqltrace select current, sql_id, sql_runtime, sql_finishtime, sql_sid, sql_uid, sql_statement, sql_database from sysmaster:syssqltrace where (sql_runtime > 5 and (sql_finishtime > (select max(sql_finishtime) from save_sqltrace)) or (sql_runtime > 5 and ((select count(*) from save_sqltrace) = 0)))| 30 00:00:00|00:00:00|| 0 00:01:00|2014-03-27 14:54:17|9237|645.80983534882432|t|t|t|t|t|t|t|400|PERFORMANCE|t|0|

Create Plug-In For OAT

Now that we created the tasks and procedures to capture the information in OAT we now want to create reports to display the information.

We will then need to create a plug-in to install in OAT.

For detailed information on creating plug-in for OAT, see Erika Von Bargen's presentation:

http://www.ibm.com/developerworks/data/library/techarticle/dm-0808vonbargen/

myreport.php – Menu

```
 function run()
{
      /**
       * By convention, OAT modules use a 'do' parameter to determine what should be executed.
       */
       switch($this->idsadmin->in['do'])
       {
……….
            $this->sessioncntReport();
            break;
                        case "sarReport":
                              // Run the SAR report
            $this->sarReport();
            break;
                        case "checkpointReport":
                              // Run the Checkpoint report
            $this->checkpointReport();
…….
       }
   } // end of function run
```

myreports.php - SAR

```
function sarReport()
  {
     /**
      * Set the page title and the menu item that should be highlighted for this report
      */
     $this->idsadmin->html->set_pagetitle("Sar");
     $this->idsadmin->setCurrMenuItem("sarReport");

     /**
      * For our SAR report, we want to select some information from the syadmin   mon_sar_proc table.
      *
      * Therefore, we first need a 'connection' to the database which we will get again  from the $this-
>idsadmin object.
      */
     $db = $this->idsadmin->get_database("sysadmin");

    $qry = "select sar_time as category "
                                  . ", pct_user as series1, '# Percent User' as series1_label "
                                  . ", pct_sys as series2, '# Percent System' as series2_label "
                                  . ", pct_wio as series3, '# Percent Wait' as series3_label "
                                  . ", pct_idle as series4, '# Percent Idle' as series4_label "
         . "from sysadmin:mon_sar order by 1";
```

myreports.php - SAR

```
/**
 * Now we'll create the graph using the Charts library class.  We create the Charts object, set the attributes which includes
 * our data array, and then we call the chart render function to create the graph.
 */
require_once ("lib/Charts.php");
$this->idsadmin->Charts = new Charts($this->idsadmin);
        $this->idsadmin->Charts->setShowZoom(true);
$this->idsadmin->Charts->setType("LINE");
$this->idsadmin->Charts->setTitle("Sar");
$this->idsadmin->Charts->setLegendDir("vertical");
$this->idsadmin->Charts->setWidth("100%");
$this->idsadmin->Charts->setDbname("sysadmin");
$this->idsadmin->Charts->setSelect(urlencode($qry));
$this->idsadmin->Charts->setHeight(500);

$this->idsadmin->Charts->Render();
```

myreports.php - SAR

```
/**
 * Underneath our pie graph, we also want to display a table listing the  memory usage per session.  So here is the query for this table.
 */
$qry = "select sar_time, pct_user, pct_sys, pct_sys, pct_wio, pct_idle "
    . "from sysadmin:mon_sar";
/**
 *  And here is the 'count' query for the # of rows returned from the previous query.
 */
$qrycnt = " select count(*) from sysadmin:mon_sar ";
/**
 * We'll use the 'gentab' API again to create the output for us.   So load the gentab class and create a new instance of the gentab class.
 */
require_once("lib/gentab.php");
$tab = new gentab($this->idsadmin);
        /**
```

myreports.php - SAR

```
/**
     * Now we call the display_tab_by_page function of the gentab class and  pass the required arguments.
     *   arg1:  Title.
     *   arg2:  Array of 'column' headings.
     *         We use the idsadmin lang function to get our string to use as a heading.
     *   arg3:  The query.
     *   arg4:  The count query.
     *   arg5:  How many rows to display per page.
     */
    $tab->display_tab_by_page("Sar",
    array(
            "1" => $this->idsadmin->lang("time"),
            "2" => $this->idsadmin->lang("pct_user"),
            "3" => $this->idsadmin->lang("pct_sys"),
            "4" => $this->idsadmin->lang("pct_wio"),
            "5" => $this->idsadmin->lang("pct_idle"),
    ),
    $qry,$qrycnt,500);
    /**
     * It's just that easy again.  Now we have our second report for Session Memory which
     * includes both a table and a pie chart.  This report can be accesses once the plugin is installed at ...
     *     http://HOSTNAME/OATINSTALL/index.php?act=myPlugin/myReports&do=sarReport
     */          }
// end of function sarReport
```

lang/en_us/lang_myreports

Here you put in the columns that will be in your report so that they will display correctly in OAT.

```
<?xml version="1.0" encoding="UTF-8"?>
<lang module="lang_myReports">
<sessionid><![CDATA[Session ID]]></sessionid>
<username><![CDATA[User Name]]></username>
<lockcnt><![CDATA[Lock Count]]></lockcnt>
<memtotal><![CDATA[Total Memory]]></memtotal>
<user_cnt><![CDATA[User Session Count]]></user_cnt>
<time_ins><![CDATA[Date/Time]]></time_ins>
<time><![CDATA[Date/Time]]></time>
<num_of_sessions><![CDATA[Number of Sessions]]></num_of_sessions><
id><![CDATA[ID]]></id><
reads><![CDATA[# Reads]]></reads>
<writes><![CDATA[# Writes]]></writes>
<deletes><![CDATA[# Deletes]]></deletes>
<pct_user><![CDATA[% User]]></pct_user>
<pct_sys><![CDATA[% System]]></pct_sys>
<pct_wio><![CDATA[% Wait I/O]]></pct_wio>
<pct_idle><![CDATA[% Idle]]></pct_idle>
</lang>
```

lang/en_us/lang_menu

Here you put in the name of the reports that will show up in OAT menu.

```
<?xml version="1.0" encoding="UTF-8"?>
<lang module="lang_menu">
<myReports><![CDATA[Custom Reports]]></myReports>
<lockReport><![CDATA[Lock Report]]></lockReport>
<sessionMemReport><![CDATA[Session Memory Report]]></sessionMemReport>
<usersescntReport><![CDATA[User Session Count Report]]></usersescntReport>
<sessioncntReport><![CDATA[Session Count Report]]></sessioncntReport>
<sarReport><![CDATA[SAR Report]]></sarReport>
<dbspacegrowthReport><![CDATA[Dbspace Growth Report]]></dbspacegrowthReport>
<tablegrowthReport><![CDATA[Table Growth Report]]></tablegrowthReport>
<checkpointReport><![CDATA[Checkpoint Report]]></checkpointReport>
<sqloperReport><![CDATA[SQL Operations Per Second Report]]></sqloperReport>
</lang>
```

plugin

This script displays this on the Menu of OAT.

<?xml version="1.0"?>

……

</plugin_info><plugin_menu><menu_pos>SQLToolBox</menu_pos><menu lang="**myReports**" name="**Custom Reports**" id="**myReports**">

<item title="**My Lock Report**" lang="**lockReport**" name="**LockReport**" link="**index.php?act=myPlugin/myReports&do=lockReport**"/>

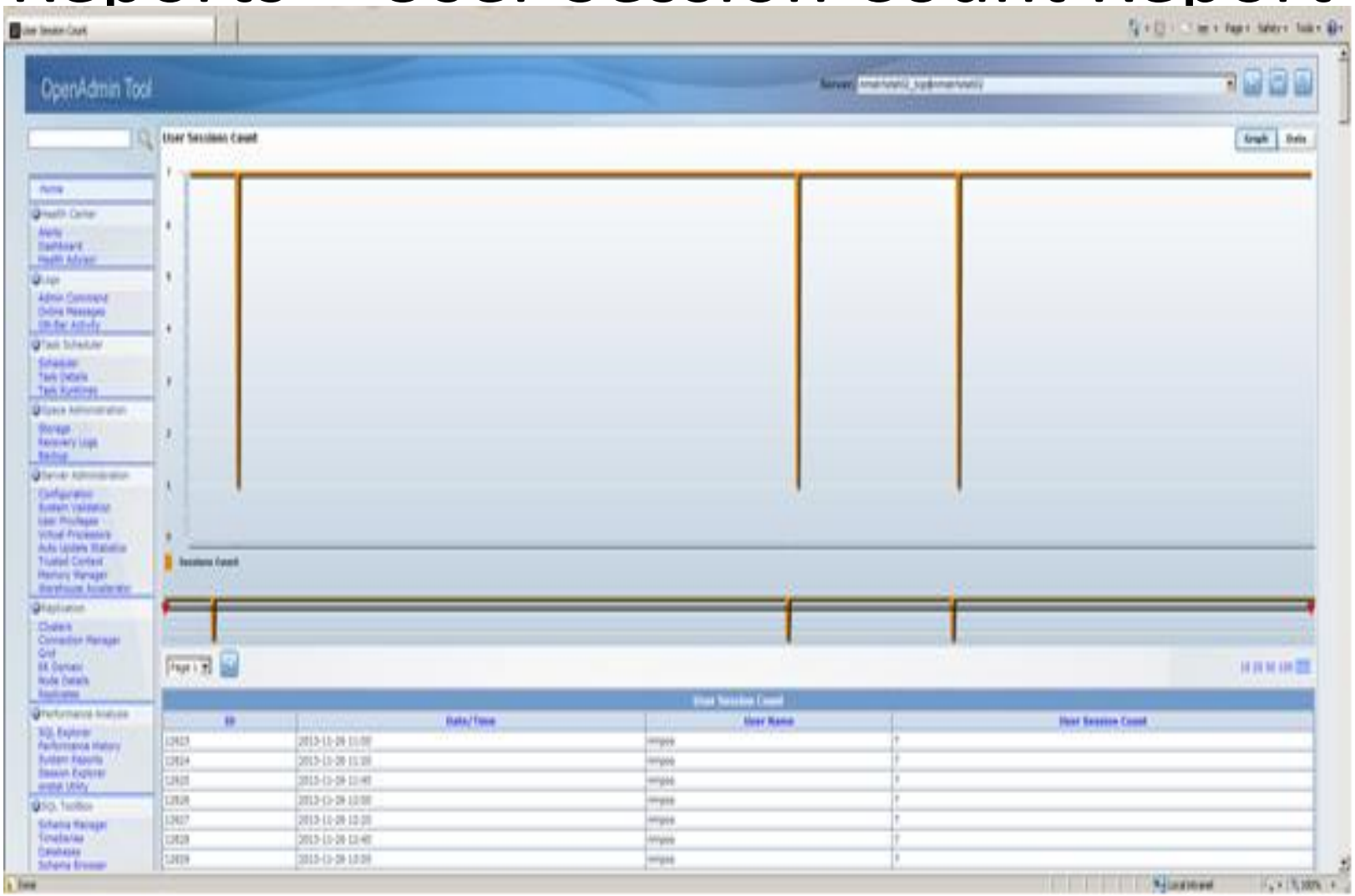
<item title="**My Session Memory Report**" lang="**sessionMemReport**" name="**SessionMemReport**" link="**index.php?act=myPlugin/myReports&do=sessionMemReport**"/>

<item title="**User Session Count Report**" lang="**usersescntReport**" name="**UsersescntReport**" link="**index.php?act=myPlugin/myReports&do=usersescntReport**"/>

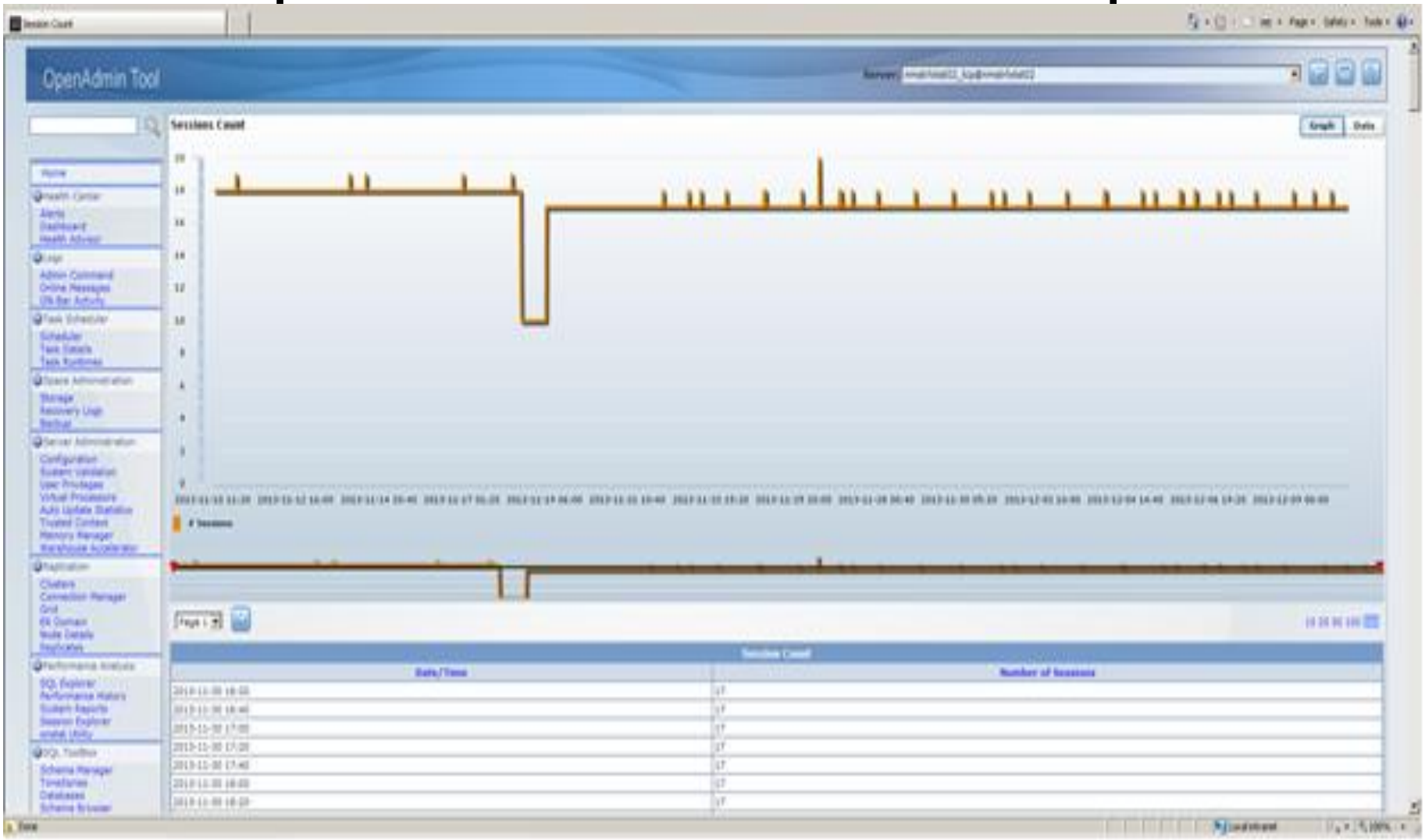
Reports Layout

So now that we have created all the tasks, procedures/functions, tables and reports we will now show how those reports look like.

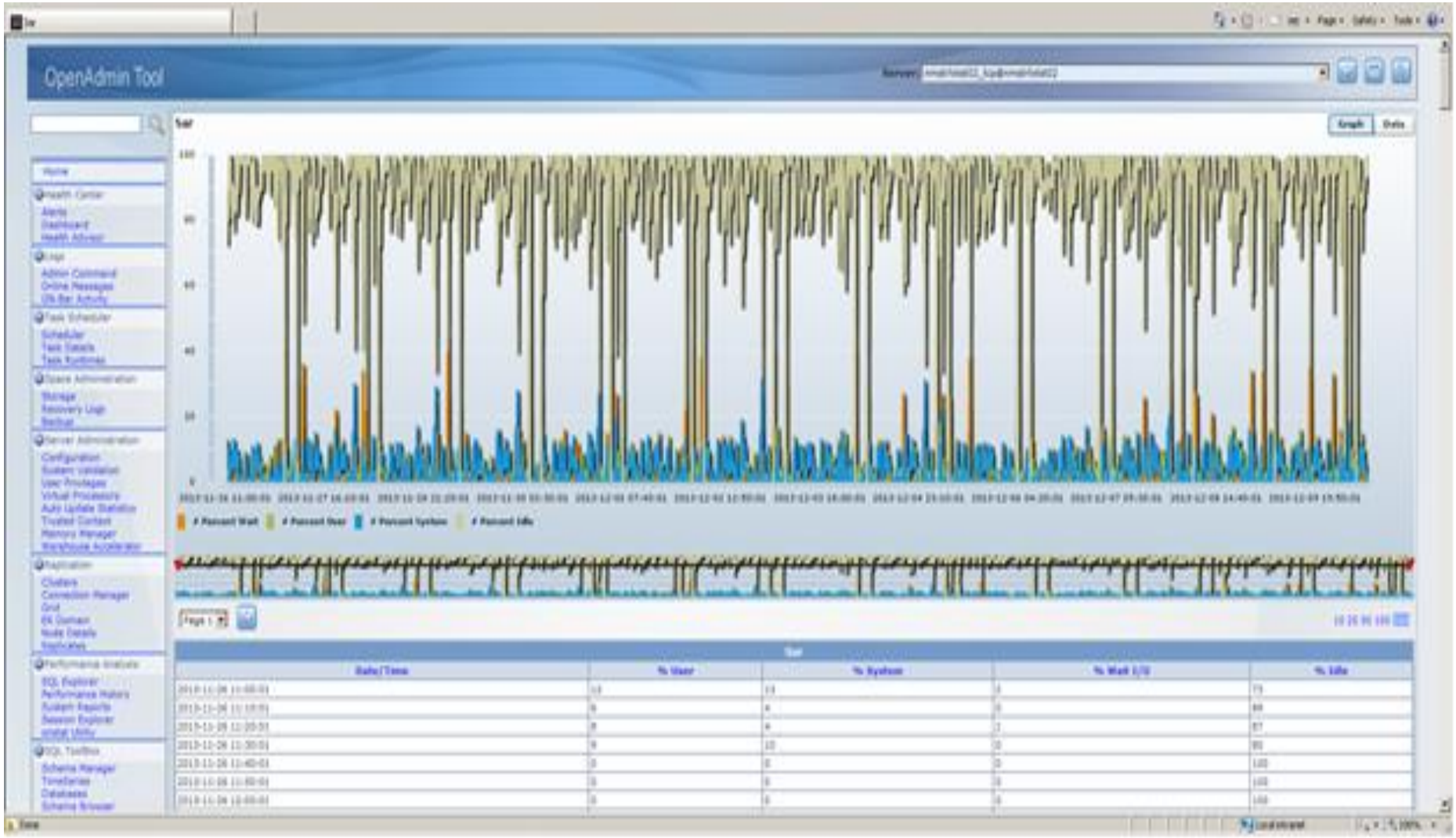
Reports – User Session Count Report



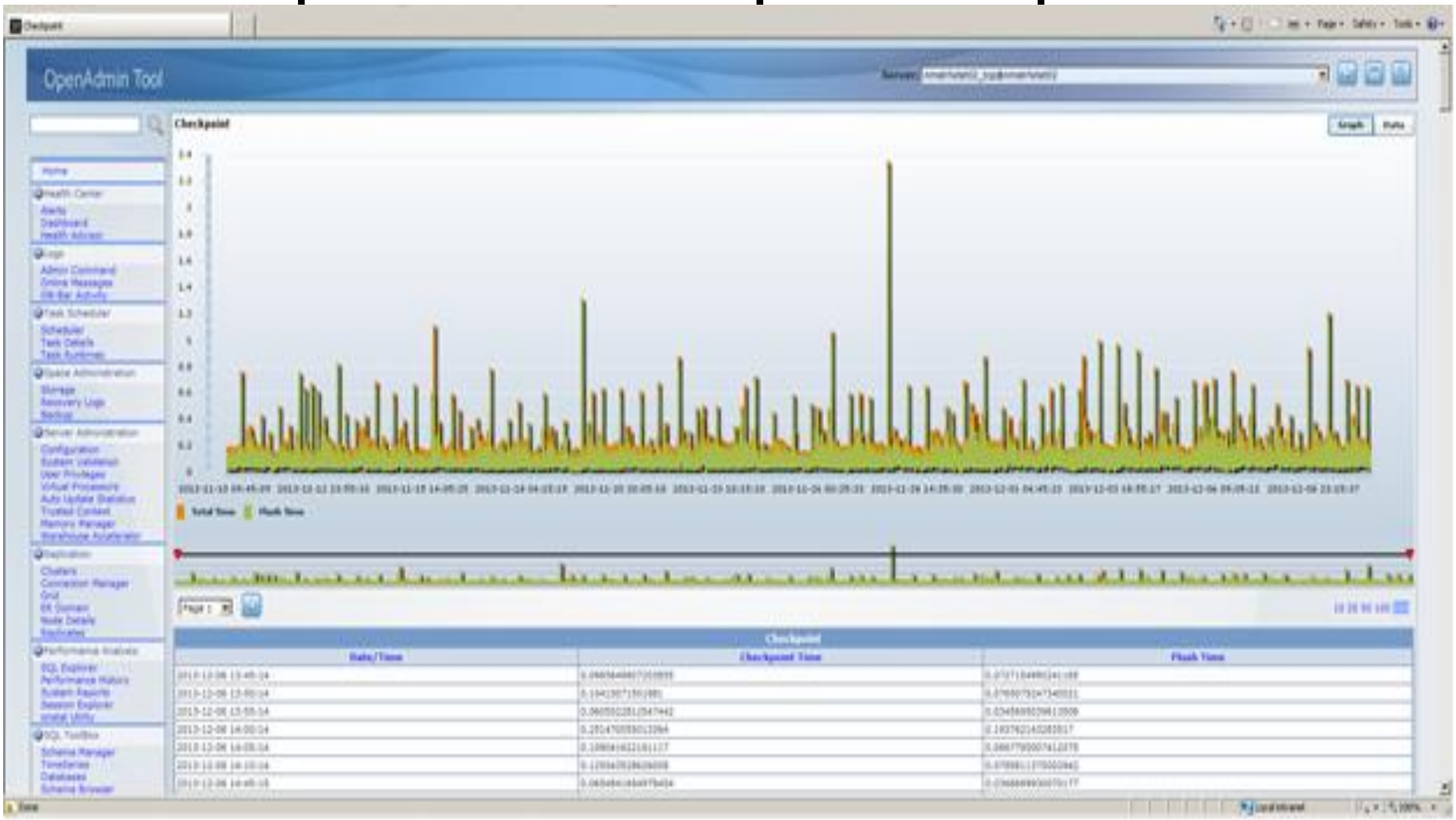
Reports – Session Count Report



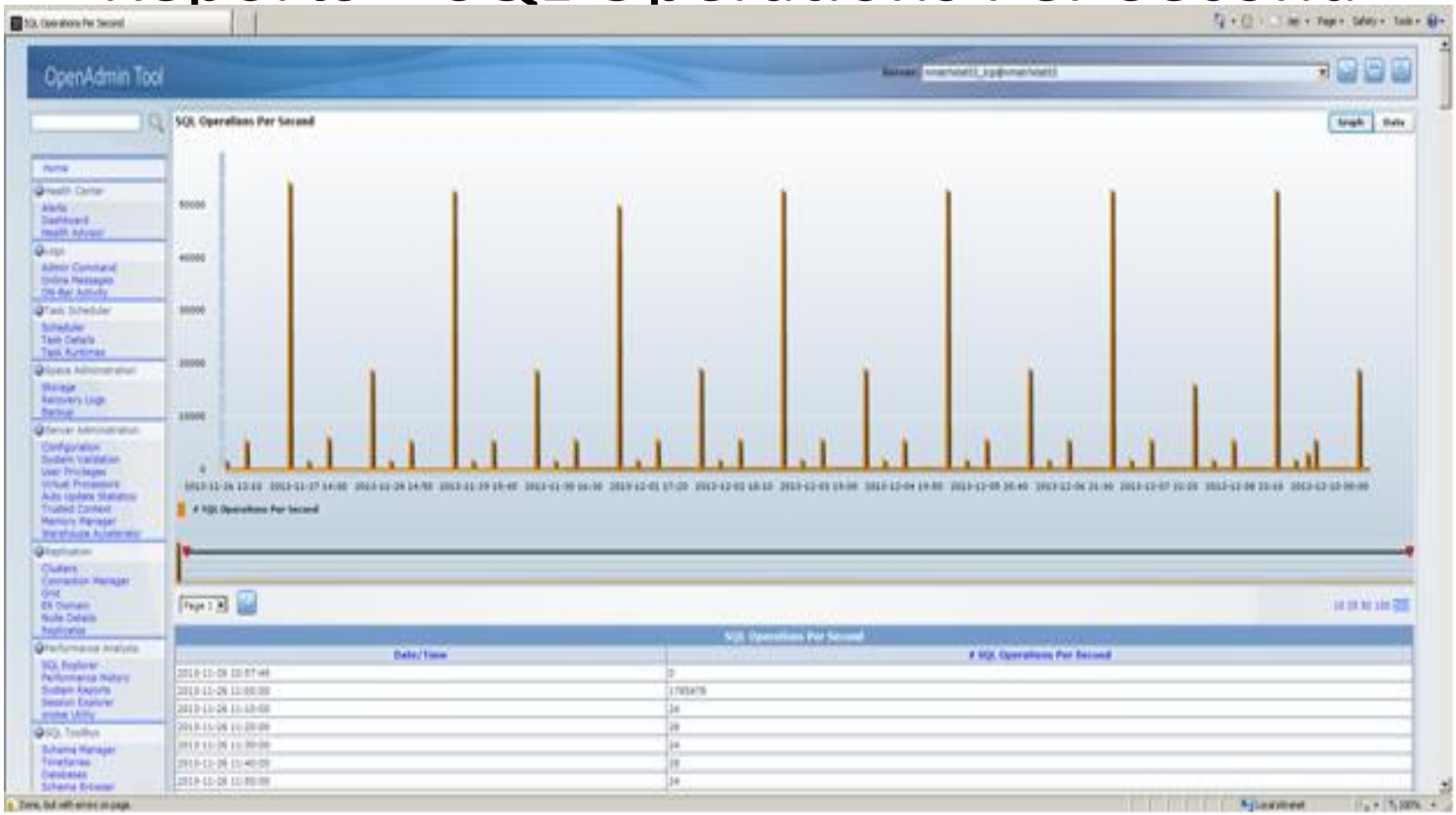
Reports – SAR Report



Reports – Checkpoint Report



Reports – SQL Operations Per Second



Customize Health Advisor

The Health Advisor comes with a set of alerts that you can set.

What if there is not one that you need?

You can create your own alerts to be monitored along with the others.

Customize Health Advisor

- One thing I found was that I first changed the name of the profile from “default” to “Store”.
- But then I found that there were certain things that required there to be a “default” profile.
- So I had my custom profile with all the “default” profile alerts and my new alerts. I then also had the alerts for the default profile which I did not use.
- I made the “Store” profile the active profile.

Health Advisor

- In this case the customer had a lot of instances at remote locations and wanted to use the health advisor to alert them to any issues.

- Here are the custom alerts that I had create for the client.

Health Advisor

- **Free Space File System** (Alert if /infx_data file system is above threshold)
- **Chunks Expanded** (Alert if a chunk has been expanded due to low space)
- **Informix Backups** (Alert if an Informix backup has not been taken in X days)

Health Advisor

Steps to enable a new alert.

- Depending on the alert you are going to have you may need to have a task run a procedure/function first before the Health Advisor runs.
  - So if needed we add a task to the ph_tasks table.
- Next we add a row to the “hadv_gen_prof” table for the new alert.

Free Space File System

- This alert is to let the DBA know that the file system where the Informix chunks reside is past a defined threshold. This way the DBA can proactively expand the file system.

- Add a task to the ph_task table, then add a row to the “hadv_gen_prof” table for the new alerts.

Free Space File System

Here is the stored procedure which writes to a file the space used and the external table to read the record.

create procedure mon_used_file_system()

system "df -k /infx_data |grep '\/infx_data' |awk '{print $4}' |sed 's/%/|/' > /prod/bin/dba/used_fs.unl";

end procedure;

create external table used_file_system
(
used_fs smallint
)
USING (DATAFILES ("DISK:/prod/bin/dba/used_fs.unl"),
FORMAT "DELIMITED"
);

Free Space File System

Here is the task to be inserted into the ph_task table.

0|mon_used_file_system|Check percent free for /infx_data file system|TASK|3|||sysadmin|execute procedure mon_used_file_system()| 0 01:00:00|23:50:00|| 1 00:00:00||1|0.0|t|t|t|t|t|t|t|400|SERVER|t|0 |

Free Space File System

Category: Storage

Alarm Name: Free Space File System

Description: RED: Checks that the percentage used for /infx_data file system is below the read alarm threshold.

Yellow: Checks that the percentage used for /infx_data file system is below the yellow alarm threshold.

Red: 90

Yellow: 80

Free Space File System

Need to insert a new record into the "**hadv_gen_prof**" table:

1|49|Storage|Free Space File System|Free Space File System|Red alarm: Checks that the percentage used for /infx_data file system is below the red alarm threshold. Yellow alarm: Checks that the percentage used for /infx_data file system is below the yellow alarm threshold.|Y| |red_rvalue|SQL|select used_fs from sysadmin:used_file_system|90|>|VALUE|90| |File system /infx_data is %param1%% used, RED WARNING is %param3%% used space. Expand the file system.|yel_rvalue|SQL|select used_fs from sysadmin:used_file_system |80|>|VALUE|80| |File system /infx_data is %param1%% used, YELLOW WARNING is %param3%% used space. Expand the file system.|||||

Chunks Expanded

- This alert is to let the DBA know that a chunk has been expanded. It is more informational than anything, but just lets the DBA be aware of the growth.

- So first we add a task to the ph_tasks table, then we add a row to the “hadv_gen_prof” table for the new alerts.

Chunk Expanded

Category: Storage

Alarm Name: Chunks Expanded

Description: YELLOW: INFORMATIONAL: Chunk was expanded.

Red:

Yellow:

Chunk Expanded

Need to insert a new record into the “hadv_gen_prof” table:

1|50|Storage|Chunks Expanded|Chunks Expanded|Red alarm: None. Yellow alarm: Chunk was extended.|Y|expand_chunk|
|VALUE|1|||VALUE|0||||SQL|execute function hadv_stage_data('','select alert_message from ph_alerts,ph_task where task_id = tk_id and tk_name = "alert_chunk_extend" and alert_time > CURRENT - 1 units day','t_yel_expand_chunk');||>|VALUE|0||INFORMATIONAL: Chunks were Expanded|||||

Informix Backups

- There is an OAT task which displays message in OAT if and Informix backup has not been taken for a specific number of days, but it does not email if that occurs.
- I added an alert in the Health Advisor to email if an Informix backup has not occurred in a specific number of days.

Informix Backups

- Here is the task to be inserted into the ph_task table.

0|check_backup_days|Checks how many days since last backup.|TASK|139|||sysadmin|check_backup_days| 0 01:00:00|23:45:00|| 1 00:00:00|2014-02-12 23:45:00|169|6.9934421121856829|t|t|t|t|t|t|t|404|BACKUP|t|0|

Informix Backups

Category: Storage

Alarm Name: Informix Backups

Description: RED: Informix Backup not Taken.

Red: 2

Yellow: 1

Questions?

Jeff Filippi
Integrated Data Consulting, LLC
jeff.filippi@itdataconsulting.com

Download OAT plugin
www.itdataconsulting.com