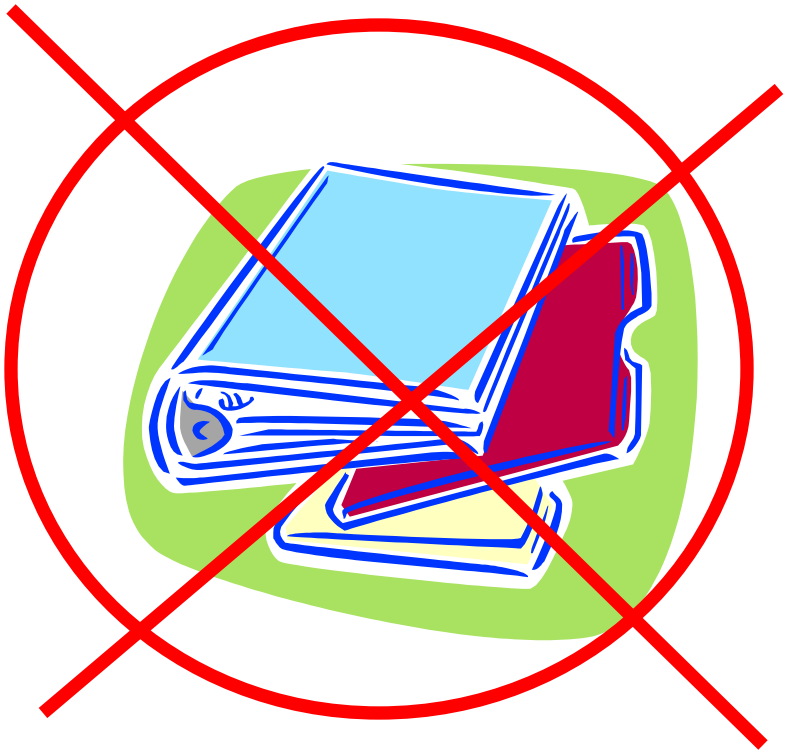
IBM Informix

Improving IDS Administration Zone

Administration Free Zone

- **Low Total Cost of Ownership and Deployment**
- **Easy Scalable Administration**
- **Open Admin API for customized administration**
- **Autonomic architecture**
- **Industrial strength, highly reliable - Install it, Set it up, and Forget about it.**



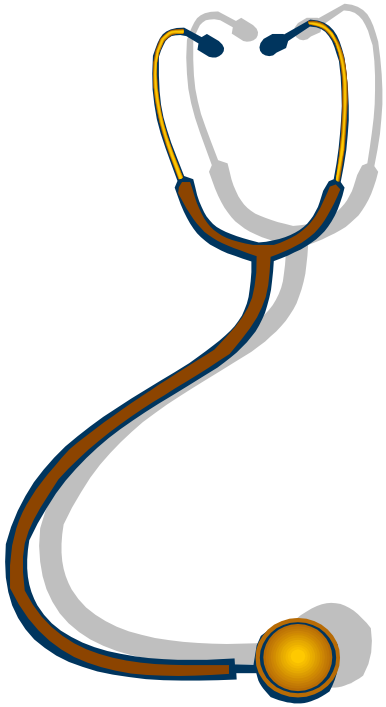
Overview

- **SQL Based Administration Commands**
- **Improved Sysmaster Database**
- **SQL History Tracing**
- **Built in Database Scheduler**
- **Quick overview of IDSAdmin**

Laying the Foundation for Improvements

- **A new database (sysadmin) for**
  - Administrative functions
  - Alert System
  - Data collection
- **Different methods for collecting information**
- **Additional information required to present in a way DBAs can understand**

Improve Information for sysmaster and onstat



- **Improve the display of information in the following areas:**
  - Improved Thread Wait Information (onstat –g ath)
  - Checkpoint Information (onstat –g ckp)
  - User/Server Environment Information (onstat –g env)
  - Online & onbar log files (onstat –m)
  - Storage system IO counts and times (onstat –g iof)
  - MGM Information (onstat –g mgm)
  - SQL Generic Cache Profiles (onstat –g ssc)
  - SQL Statement Performance History (onstat –g his)

Make the information clear for the DBA to understand

New Sysmaster Tables

TABLE NAME	DESCRIPTION
syscheckpoint	**The information about the checkpoint and associated statistics**
systcblst	**Modified the existing table to add wait stats.**
sysenvses	**View Informix's session environment variables**
sysenv	**View the servers environment variables**
sysonlinelog	**View the online.log for the server**
sysscblst	**Improvement to view the memory used by session**
sysnetworkio	**View the network I/O generated by database session**
sysdual	**Oracle compatibility feature**
syssqlcacheprof	**Displays the profile information about each SQL cache**
syssqltrace	**The sql statements which have been recently executed on the system**
syssqltrace_itr	**The list of iterators for the SQL statement.**
syssqltrace_info	**General information about the SQL tracing**
sysnetglobal	**Global Network Information**
sysnetclienttype	**Network information based on client type**
sysbaract_log	**The OnBar Activity Log file**
sysrstcb	**Improvement to view I/O and lock wait information**

Onstat –g ath

- **Remove/Reduce the number of thread status of “sleeping forever”**
- **Give the DBA a clear picture of what is happening**

```
Threads:
tid      tcb              rstcb          prty status                 vp-class    name
2        10bbf36a8         0              2    sleeping forever         3lio    lio vp 0
3        10bc12218         0              2    sleeping forever         4pio    pio vp 0
4        10bc31218         0              2    sleeping forever         5aio    aio vp 0
5        10bc50218         0              2    sleeping forever         6msc    msc vp 0
6        10bc7f218         0              2    sleeping forever         7aio    aio vp 1
7        10bc9e540         10b231028      4    sleeping secs: 1         1cpu    main_loop()
8        10bc12548         0              2    running                  1cpu    tlitcppoll
9        10bc317f0         0              3    sleeping forever         1cpu    tlitcplst
10       10bc50438         10b231780      2    sleeping forever         1cpu    flush_sub(0)
11       10bc7f740         0              2    sleeping forever         8aio    aio vp 2
12       10bc7fa00         0              2    sleeping forever         9aio    aio vp 3
13       10bd56218         0              2    sleeping forever        10aio    aio vp 4
14       10bd75218         0              2    sleeping forever        11aio    aio vp 5
15       10bd94548         10b231ed8      3    sleeping forever         1cpu    aslogflush
16       10bc7fd00         10b232630      1    sleeping secs: 26        1cpu    btscanner 0
32       10c738ad8         10b233c38      4    sleeping secs: 1         1cpu    onmode_mon
50       10c0db710         10b232d88      2    cond wait  netnorm       1cpu    sqlexec
```

Improved onstat –g ath

Threads:

tid	tcb	rstcb	prty	status	vp-class	name
2	10bbf36a8	0	2	IO Idle	3lio	lio vp 0
3	10bc12218	0	2	IO Idle	4pio	pio vp 0
4	10bc31218	0	2	running	5aio	aio vp 0
5	10bc50218	0	2	IO Idle	6msc	msc vp 0
6	10bc7f218	0	2	running	7aio	aio vp 1
7	10bc9e540	10b231028	4	sleeping secs: 1	1cpu	main_loop()
8	10bc12548	0	2	running	1cpu	tlitcppoll
9	10bc317f0	0	3	sleeping forever	1cpu	tlitcplst
10	10bc50438	10b231780	2	IO Wait	1cpu	flush_sub(0)
11	10bc7f740	0	2	IO Idle	8aio	aio vp 2
12	10bc7fa00	0	2	IO Idle	9aio	aio vp 3
13	10bd56218	0	2	IO Idle	10aio	aio vp 4
14	10bd75218	0	2	IO Idle	11aio	aio vp 5
15	10bd94548	10b231ed8	3	sleeping forever	1cpu	aslogflush
16	10bc7fd00	10b232630	1	sleeping secs: 34	1cpu	btscanner 0
32	10c738ad8	10b233c38	4	sleeping secs: 1	1cpu	onmode_mon
50	10c0db710	10b232d88	2	IO Wait	1cpu	sqlexec

SQL Admin Commands

- **A set of User Defined Routines (UDRs) to administer the Informix database server.**
- **The major categories of administration include:**
  - Space Management
  - Configuration Management
  - Routine task maintenance
  - System Validation (oncheck functionality)
- **Feature Benefits**
  - SQL Based Administration
  - Remote Administration
  - Tracking of command execution and results in a system table

Admin Commands – Two New UDRs

- **Two UDRs called *task* & *admin* are part of the sysadmin database**
- **They perform exactly the same, only the return code is different**
  - *task()* UDR returns a character string describing the return status

```
EXECUTE FUNCTION task('create dbspace', 'dbspace2', '/CHUNKS/dbspace2');
(expression)  created dbspace number 2 named dbspace2
```

  - admin() UDR returns an integer which is a link to the *command_history* table

```
EXECUTE FUNCTION admin('create dbspace', 'dbspace2', '/CHUNKS/dbspace2');
(expression)  107
```

- **Both UDRS log all executions into a table called *command_history in the sysadmin database***

SQL Admin Command - Tracking

- **SQL Admin API logs all executions into a table called *command_history in the sysadmin database***

Column	Type	Description
cmd_number	serial	
cmd_exec_time	datetime year to second	Time the command was started
cmd_user	varchar	User executing the command
cmd_hostname	varchar	Host the command was executed from
cmd_executed	varchar	The command executed
cmd_ret_status	Integer	Return code
cmd_ret_msg	lvarchar	Return message

SQL Admin Commands - Parameters

- **Environment Variable Expansion**
  - A pathname may start with an environment variable.
  - The environment variable may only exist in the server's environment

- **Unit Extensions**
  - All offsets and sizes can be provided with unit extensions
  - The extensions are case insensitive
  - Default is KB
  - PB, TB, GB, MB, KB, B

SQL Admin Commands – EXAMPLE

```
EXECUTE FUNCTION admin('create dbspace','dbspace2',`$INFORMIXDIR/SPACE/dbspace2',`20MB`);
(expression)          108

SELECT * FROM command_history WHERE cmd_number IN (108)
cmd_number       108
cmd_exec_time    2005-11-17 16:26:15
cmd_user         informix
cmd_hostname     olympia.beaverton.ibm.com
cmd_executed     create dbspace
cmd_ret_status   0
cmd_ret_msg      created dbspace number 2 named dbspace2
```

Admin API Commands

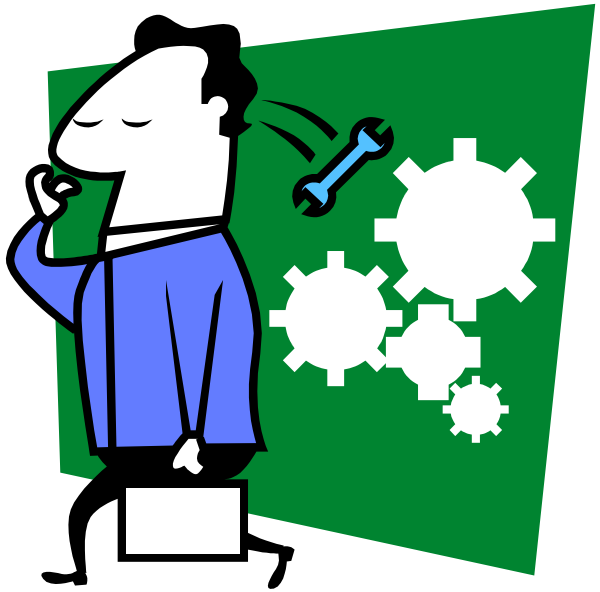
- ADD BUFFERPOOL
- ADD CHUNK
- ADD LOG
- ADD MEMORY
- ADD MIRROR
- ALTER CHUNK OFFLINE
- ALTER CHUNK ONLINE
- ALTER LOGMODE
- ALTER PLOG
- ARCHIVE FAKE
- CHECK DATA
- CHECK EXTENTS
- CHECK PARTITION
- CHECKPOINT
- CLEAN SBSPACE
- CREATE BLOBSPACE
- CREATE CHUNK
- CREATE DBSPACE
- CREATE SBSPACE
- CREATE TEMPDBSPACE
- DROP BLOBSPACE
- DROP CHUNK
- DROP DBSPACE
- DROP LOG
- DROP SBSPACE
- DROP TEMPDBSPACE
- ONMODE
- PRINT ERROR
- PRINT PARTITION
- QUIESCENT
- RENAME SPACE
- SET CHUNK OFFLINE
- SET CHUNK ONLINE
- SET DATASKIP ON
- SET DATASKIP OFF
- SET SBSPACE ACCESSTIME ON
- SET SBSPACE ACCESSTIME OFF
- SET SBSPACE AVG_LO_SIZE
- SET SBSPACE LOGGING ON
- SET SBSPACE LOGGING OFF
- SET SQL TRACING
- SET SQL TRACING OFF
- SET SQL TRACING ON
- SET SQL TRACING RESIZE
- SET SQL USER TRACING
- SET SQL USER TRACING CLEAR
- SET SQL USER TRACING OFF
- SHUTDOWN
- START MIRRORING space
- STOP MIRRORING

See IBM Informix Guide to SQL: Syntax, Chapter 6

DBA Constant Struggle

Identifying Performance Bottlenecks in SQL Statements

- **Current methods for examination**
  - Set explain
  - Command line utilities
  - sysmaster
- **Lack of simplicity in the process**
- **Hard to build a repeatable process**



Questions DBAs like to ask?

- **How long did a SQL statements take?**
- **How many resources of each category did a statement take?**
  - Disk I/O
  - Memory
  - CPU
- **How long and how many times did we wait on each resource?**
  - Locks
  - Disk I/O

Solutions – SQL Query Drill Down Feature

- **Collect detailed sql performance information.**
- **Display using onstat or sysmaster.**
- **Dynamically configurable**
- **By default disabled**
- **Global and User Tracing modes**

Name	Description
User ID	**User id**
Session ID	**Database Session ID**
Database Name	**Current Database Name**
Statement Type	**Type of SQL statement being executed**
Statement Execution time	**Duration of the SQL statement**
Time of Execution	**Date & Time this statement completed**
Statement Text	**SQL statement text or the procedure stack trace with statement type**
RSAM statistics	• **Buffer Reads & Writes** • **Page reads & Writes** • **Memory Sorts, disk Sorts,** • **Lock requests, waits** • **Logical Log Records**
SQL statistics	• **Estimated # of rows** • **Estimated Cost** • **# of rows returned** • **Statement Type** • **Database Isolation Level**

Controlling SQL Query Drill Down

- **ONCONFIG variable SQLTRACE**
  - Level =[off,low,med,high]
  - Ntraces=[number of traces]
  - Size=[size of each trace buffer in KB]

```
SQLTRACE level=low,ntraces=2000,size=1,mode=global
```

- **Turn off SQL Tracing for session id 147**

```
execute function task("SET SQL USER TRACING OFF",147);
```

Controlling SQL Query Drill Down

- **Dynamically enable or modify SQL Tracing**
  - Trace 2000 SQL statements
  - Trace 1024 bytes of data for each SQL statement

```
execute function task("SET SQL TRACING ON",2000,1);
```

- **Turn off SQL Tracing**

```
execute function task("SET SQL TRACING OFF");
```

SQL Query Drill Down - onstat

```
Database:        sysadmin
 Statement text:
  SELECT MAX(run_task_seq) FROM ph_run A, ph_task B WHERE A.run_task_id = ?
    AND A.run_task_id = B.tk_id AND A.run_time + B.tk_delete < CURRENT

 Iterator/Explain
 ================
    ID   Left  Right   Est Cost   Est Rows   Num Rows      Type
     3      0      0          1          1          1     Index Scan
     4      0      0         19        545          1     Index Scan
     2      3      4         20          5          1     Nested Join
     1      2      0          1          1          1     Group

 Statement information:
  Sess_id  User_id  Stmt Type           Finish Time    Run Time
  21       0        SELECT             10:51:11       0.0023

 Statement Statistics:
  Page       Buffer     Read       Buffer     Page       Buffer     Write
  Read       Read       % Cache    IDX Read   Write      Write      % Cache
  0          77         100.00     0          0          0          0.00

  Lock       Lock       LK Wait    Log        Num        Disk       Memory
  Requests   Waits      Time (S)   Space      Sorts      Sorts      Sorts
  0          0          0.0000     0.000 B    0          0          0

  Total      Total      Avg        Max        Avg        I/O Wait   Avg Rows
  Executions Time (S)   Time (S)   Time (S)   IO Wait    Time (S)   Per Sec
  220        78.8463    0.3584     1.9557     0.000000   0.000000   439.9908

  Estimated  Estimated  Actual     SQL        ISAM       Isolation  SQL
  Cost       Rows       Rows       Error      Error      Level      Memory
  20         1          1          0          0          DR         41552
```

Built-in Database Scheduler

- **Ability to schedule SQL, Stored procedures or a UDR**
  - Timed backup, profile information.
- **Scheduler managed entities:**
  - Tasks
  - Sensor
  - Startup Task
  - Startup Sensor
- **These entities are driven by the data inside a table called** *ph_task.* **Each row is a task.**

You can schedule your Tasks on the following base:

- **Schedule to only run once at server startup time**
- **Schedule at a specific start time**
- **Schedule in a specified frequency**
- **Schedule a combination of start time and frequency**

Scheduler Specialized Task

- **Task**
  - A means to execute a specific job at a specific time or interval
  - No intention to evaluate the returned data
  - The *task* executes by invoking
    - A single or compound SQL statement
    - Stored procedure,
    - C User Defined Routine
    - Java User Defined Routine

Sensor - Specialized Task

- **A specialized task geared at collecting and saving data without the DBA doing much work.**
- **A means to get information about a managed element**

Why use Tasks and Sensors?

- **Build a specialized task**
  - A way of ensuring routine jobs get completed
  - Periodically check,analyze data server is operating efficiently

- **Sensors provide a simple way of collecting information**
  - Easy to add a new sensor
  - Provide a portable way of collecting information without using the operating system.

Examples of Creating a Sensor

- **Create a sensor to track the number of sessions on the database server every 5 minutes.**
- **Only keep the data for 1 day**

```
INSERT INTO ph_task (tk_name,tk_type,tk_group,tk_description,tk_result_table,
tk_create,tk_execute,tk_start_time,tk_stop_time,tk_frequency,tk_delete)
VALUES
("mon_user_count","SENSOR","SERVER","Count the number of user count ",
"mon_user_count","create table mon_user_count (ID integer, session_count integer)",
"insert into mon_user_count select $DATA_SEQ_ID, count(*) from
    sysmaster:syssessions",
NULL,NULL,
INTERVAL ( 5 ) MINUTE TO MINUTE,
INTERVAL ( 1 ) DAY TO DAY);
```

The PH_TASK table

select * from PH_TASK where tk_name="mon_user_count"

```
tk_id                19
tk_name              mon_user_count
tk_description       Count the number of
   user count
tk_type              SENSOR
tk_sequence          243
tk_result_table      mon_user_count
tk_create            create table
   mon_user_count (ID integer,
   session_count integer)
tk_dbs                sysadmin
tk_execute            insert into
   mon_user_count select $DATA_SEQ_ID,
   count(*) from sysmaster:syssessions
```

```
tk_delete            30 00:00:00
tk_start_time
tk_stop_time
tk_frequency          0 00:05:00
tk_next_execution    2007-07-15 07:19:21
tk_total_execution   2
tk_total_time        2.3839
tk_monday            t
tk_tuesday           t
tk_wednesday         t
tk_thursday          t
tk_friday            t
tk_saturday          t
tk_sunday            t
tk_attributes        1
tk_group             SERVER
tk_enable            t
tk_priority          0
```

Built in Sensor & Tasks

Sensor Name	Description
mon_command_history	**Purges the command history table**
mon_config	**Saves any difference in the onconfig file**
mon_config_startup	**Save the onconfig file on every server startup**
mon_profile	**Save the server profile information**
mon_vps	**Collects the virtual processor timings**
mon_checkpoint	**Save information about checkpoints**
mon_table_profile	**Save table profile information including UDI counters**
mon_table_names	**Save the table names along with their create time**
mon_users	**Save profile information about each user**
check_backup	**Check to ensure backups have been done**

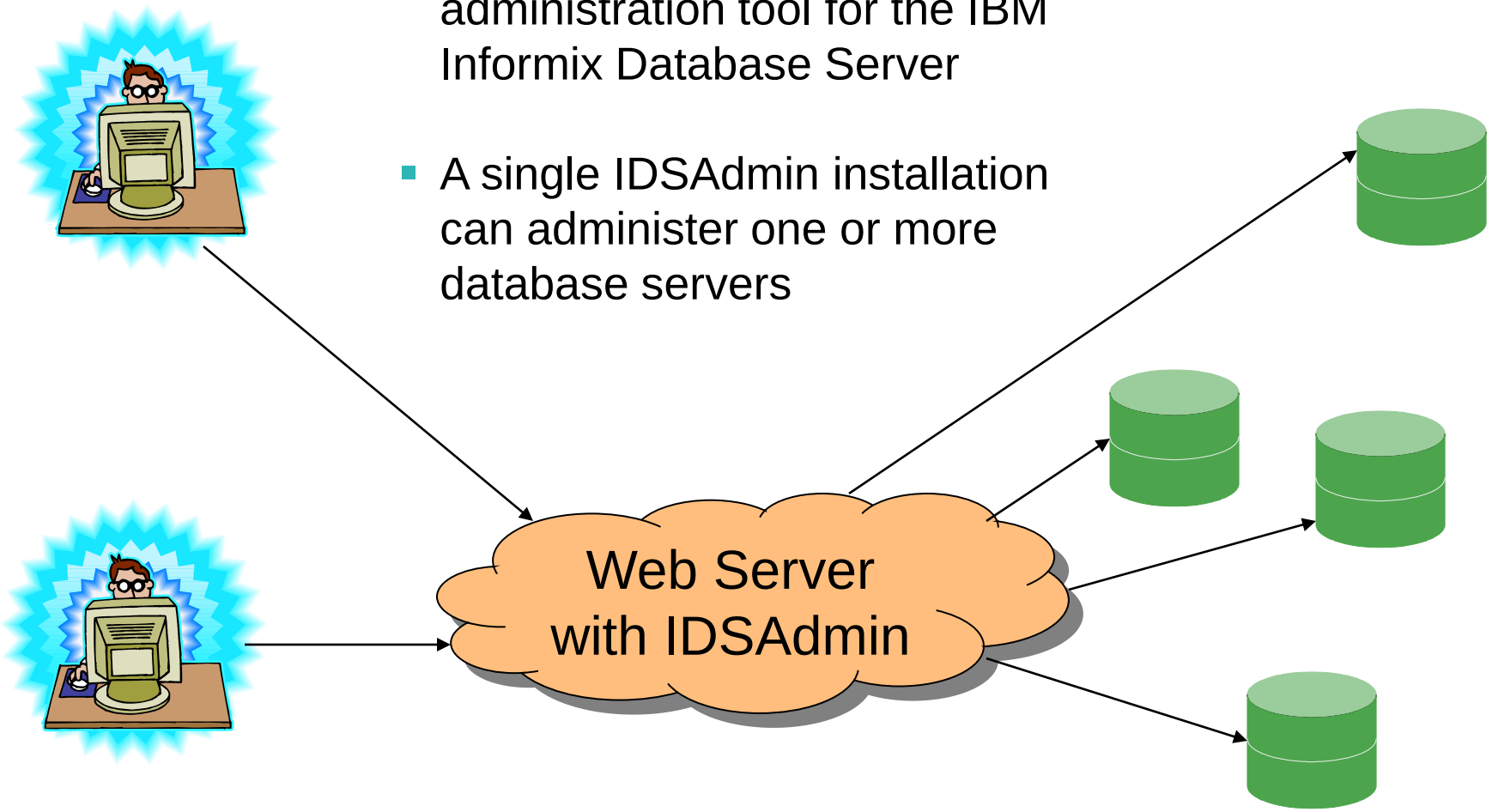
Admin Console

Configuring Scheduler

- **Scheduler is switched on by default**
- **Execute function task (“scheduler stop”)**
- **Execute function task (“scheduler start”)**
- **Execute function task(“reset sysadmin”, “dbspace1”);**
- **$INFORMIXDIR/etc/sysadmin/stop – If present, scheduler will not start when the database server starts**

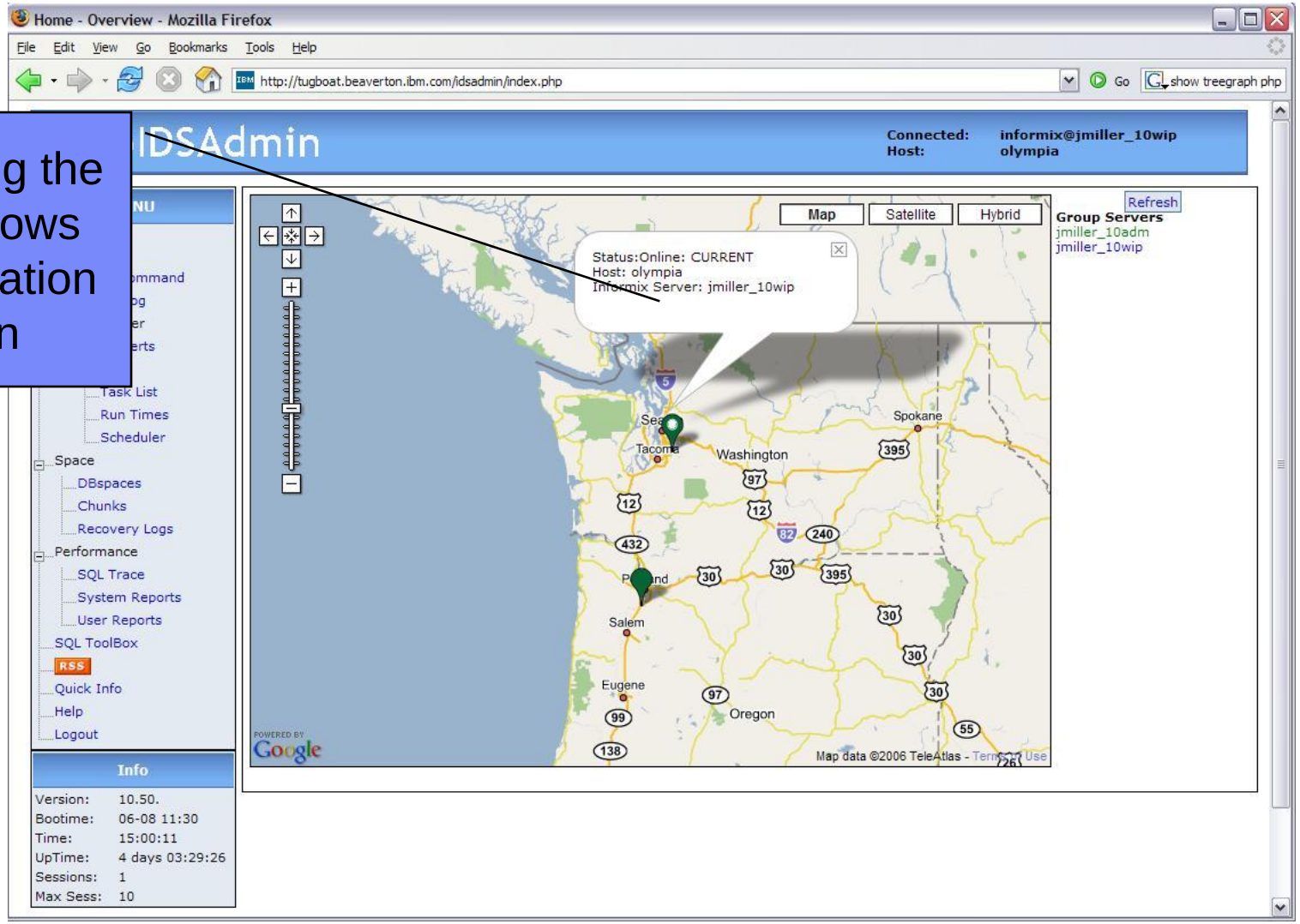
What is IDSAdmin?

- IDSAdmin is a web-based administration tool for the IBM Informix Database Server
- A single IDSAdmin installation can administer one or more database servers



Map View

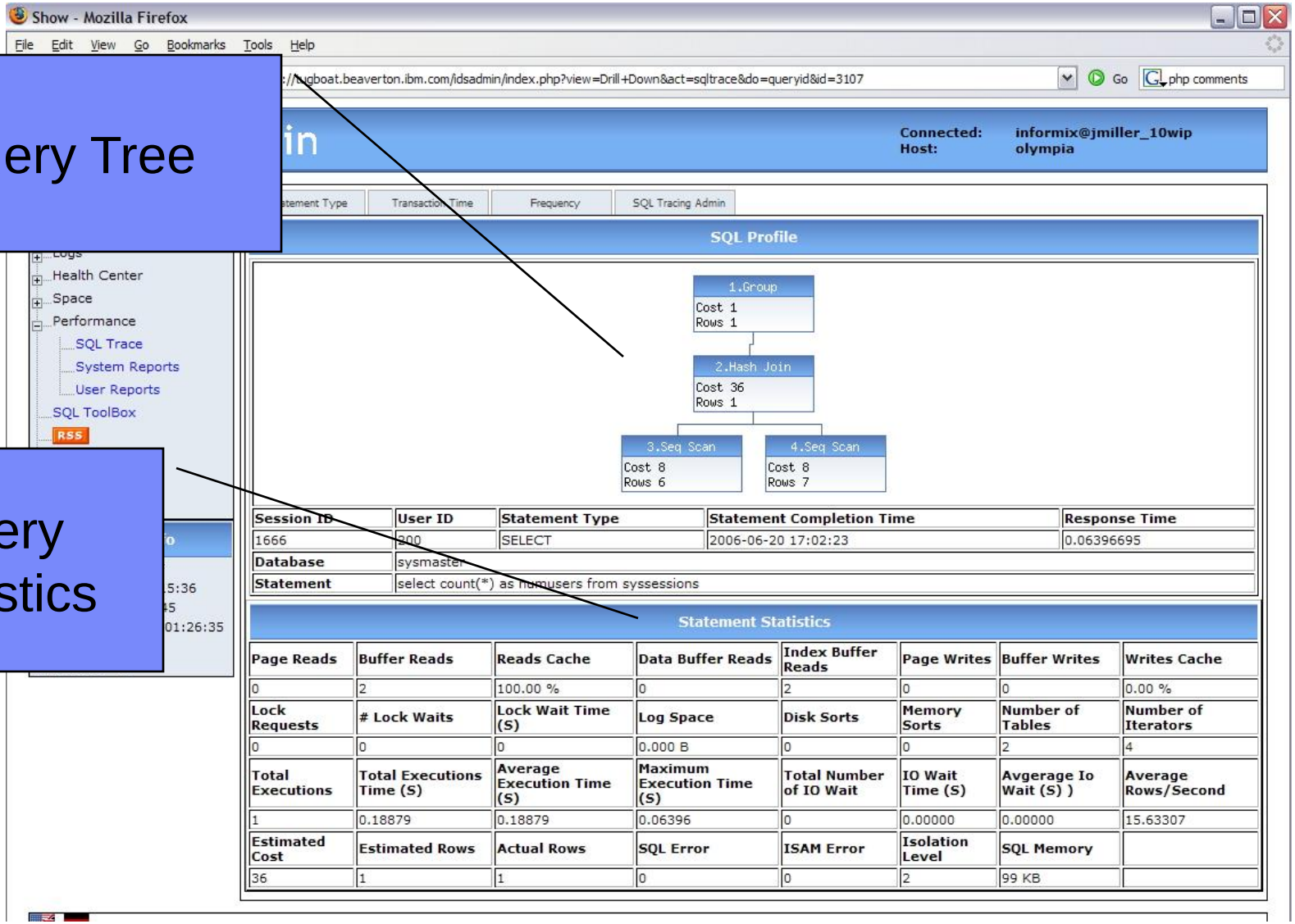
Highlighting the server shows the information balloon



A Specific Instance of an SQL Statement

Query Tree

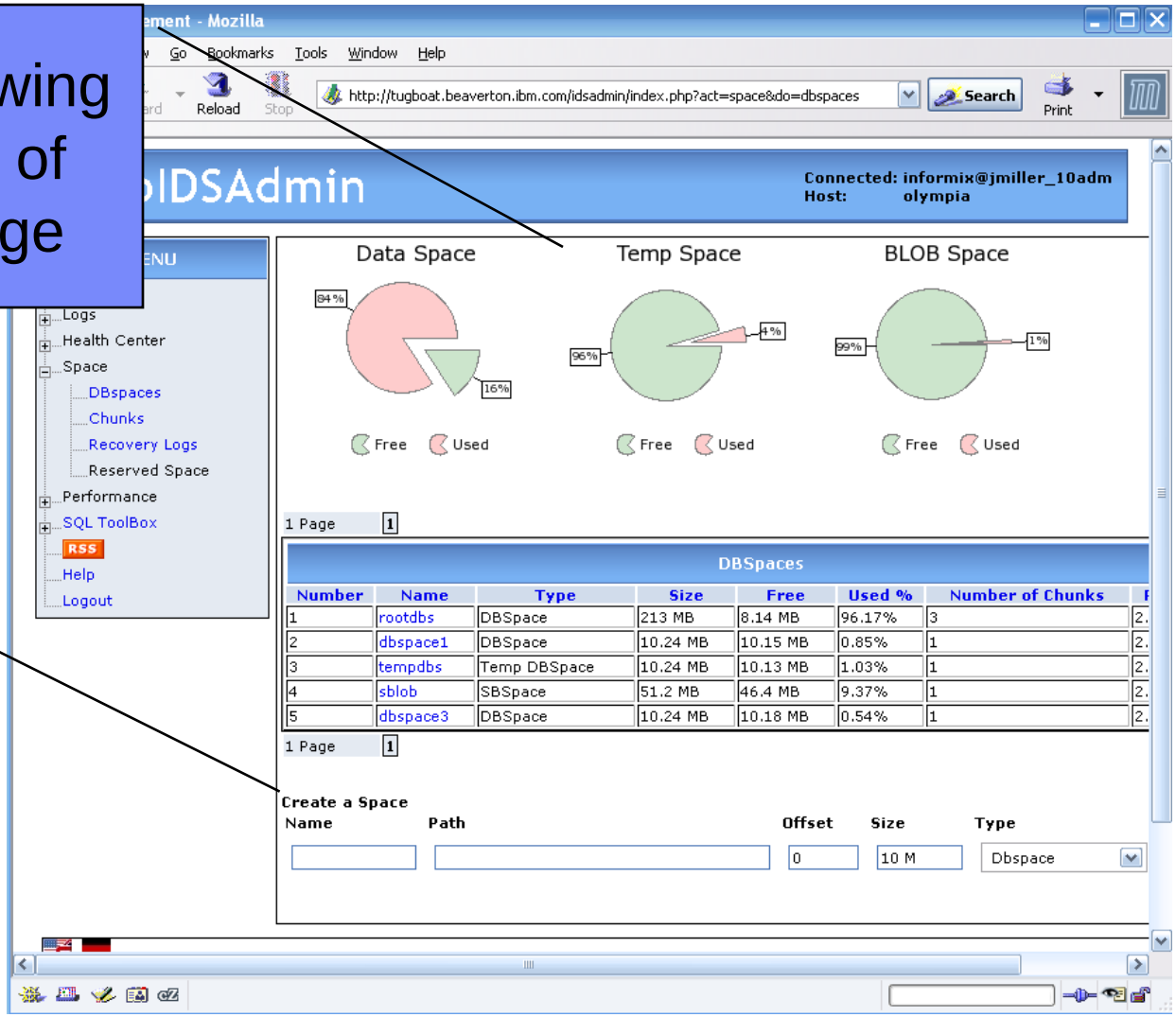
Query Statistics



Overview of Disk Space

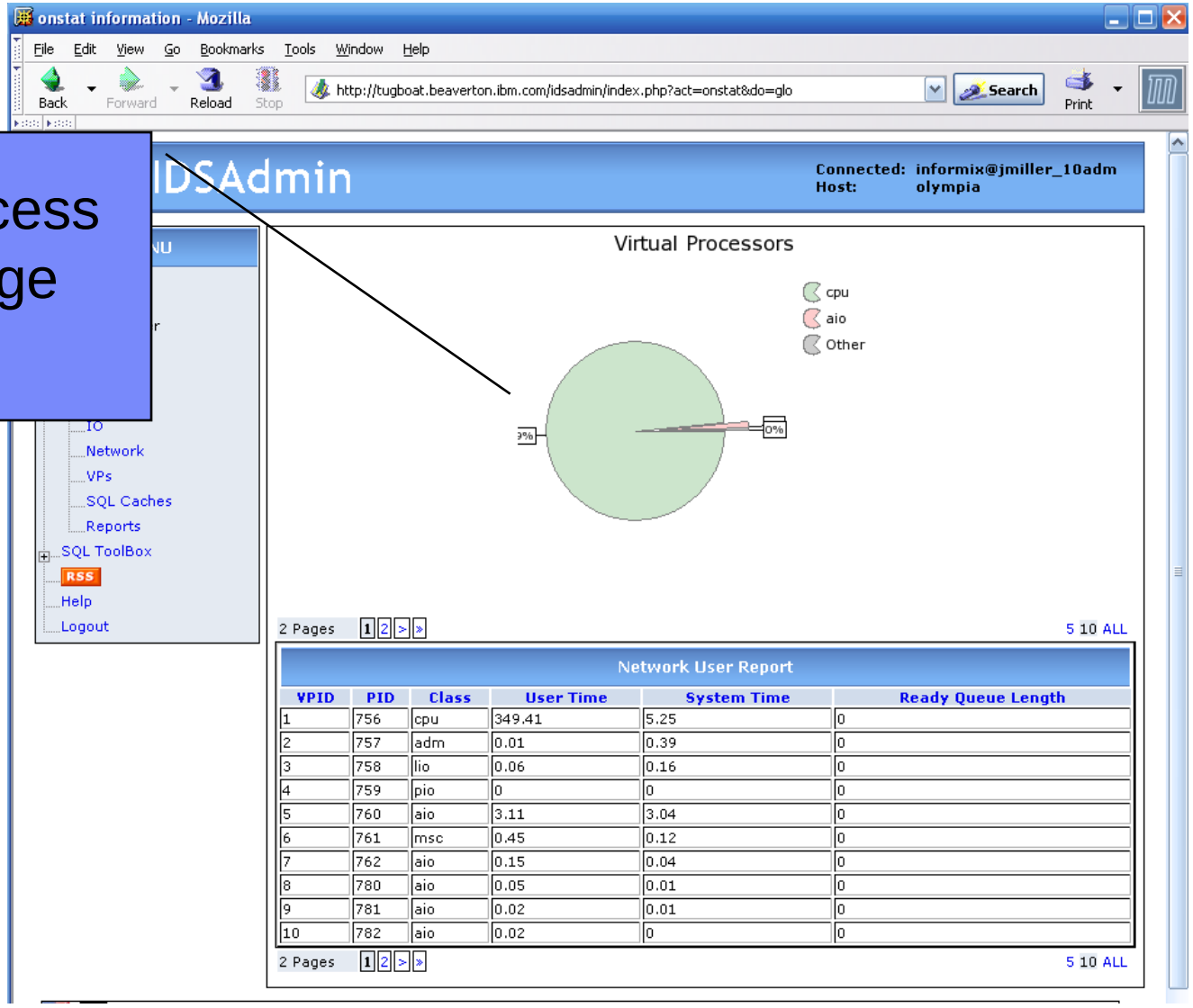
Graphs showing quick view of space usage

Ability to create additional spaces



Process Usage

Virtual Process CPU Usage Graph



Questions

Terminate Idle Users with the Database Admin System

- **The components we are going to utilize are:**
  - Database Scheduler
  - Alert System
  - User Configurable Thresholds
  - SQL Admin API
- **We are going to break the above problem into three separate parts.**
  - Creating a tunable threshold for the idle time out
  - Develop a stored procedure to terminate the idle users
  - Schedule this procedure to run at regular intervals

```
INSERT INTO
    ph_threshold(name,task_name,value,value_type,description)VALUES("IDLE
    TIMEOUT", "Idle Timeout","60","NUMERIC","Maximum amount of time in
    minutes for non-informix users to be idle.");
```

```
CREATE FUNCTION idle_timeout( task_id INT, task_seq INT)
RETURNING INTEGER
DEFINE time_allowed INTEGER;
DEFINE sys_hostname CHAR(16);
DEFINE sys_username CHAR(257);
DEFINE sys_sid      INTEGER;
DEFINE rc           INTEGER;
{*** Get the maximum amount of time to be idle ***}
SELECT value::integer        INTO time_allowed
FROM ph_threshold
 WHERE name = "IDLE TIMEOUT";
{*** Find all users who are idle longer than the threshold ***}
FOREACH
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 CURRENT - DBINFO("utc_to_datetime",last_run_time) > time_allowed UNITS MINUTE
AND lower(A.username) NOT IN( "informix", "root")
{*** If we sucessfully terminated a user log ***}
 {*** the information into the alert table    ***}
```

```
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,"INFO", "GREEN",  "ADDRESSED","USER","TIMEOUT",
                  "User "||TRIM(sys_username)||"@"||TRIM(sys_hostname)||
                   " sid("||sys_sid||")"|| " terminated due to idle timeout.",NULL );
END IF
END FOREACH;
RETURN 0;
END FUNCTION;
/* ************************************************************ *
Create a task which will schedule the idle_timeout
SPL *  to be run between 6 AM and 6 PM.
************************************************************ */
INSERT INTO
    ph_task(tk_name,tk_type,tk_group,tk_description,tk_execute,tk_start_time,tk_stop_time,tk_fr
    equency)
VALUES("Idle Timeout","TASK","USER","Remove all idle users from the
    system.","idle_timeout",DATETIME(06:00:00) HOUR TO SECOND,DATETIME(18:00:00)
    HOUR TO SECOND,INTERVAL ( 10 ) MINUTE TO MINUTE);
```