

TECHNICAL NOTES

**EMC Unisphere for VMAX
Database Storage Analyzer
v8.1
Technical Notes**

P/N H14606
REV 01
October 7, 2015

This document contains information on these topics:

Contents

Introduction 2
Architecture 3
Database collection and retention policy 3
Mapping files 4
Storage statistics collection 5
Support matrix 6
Creating guest user 6

Introduction

Database Storage Analyzer (DSA) is an application that provides a database to storage performance troubleshooting solution for Oracle and SQL Server databases running on EMC Symmetrix and VMAX storage systems.

This document describes how DSA collects data and the prerequisites that you should be aware of prior to installing the application.

DSA is a feature of the Unisphere for VMAX Foundation Suite. It supports database to storage correlation by providing a shared view of how performance issues correlate to database and storage level activities. This view is accessible by database administrators (DBAs) and storage administrators (SAs). The view presents I/O metrics such as input/output operations per second (IOPS), throughput and response time from both the data base and the storage system, which helps to immediately identify gaps between the database I/O performance and the storage I/O performance.

Note the following:

- Unisphere for VMAX is the only software requirement.
- There is no additional cost to use DSA.
- DSA currently supports Symmetrix systems running Enginuity 5671 – 5876 and VMAX systems running HYPERMAX OS 5977 or higher.
- There is no need to allocate any additional resources for the Unisphere server other than what is specified in the Unisphere for VMAX documentation.

KEY FEATURES ADDED IN 8.1:

Hinting

With Version 8.1 DSA introduces the new Hinting feature. This feature provides a way to improve application performance by sending hints to the array for data that is likely to be accessed in a given period of time.

The idea is to allow customers to accelerate mission critical processes by leveraging the hinting feature in the platform.

The user would be able to set hints from the analytics tab in the DSA interface while going through a simple process of selecting the relevant objects and then setting the priority and the time when the hint would be running.

Offering the capability for Hosts to provide hints is critical for enhancing FAST VP to be “Application aware” as well ensure SLO compliance.

Note that Hinting would be supported for VMAX3 with HYPERMAX OS 5977 or higher and for Oracle 10G and above (Hinting is not supported for SQL Server in this release) Also, Hinting is not supported for databases running on virtual environments other than VMware with RDM configuration.

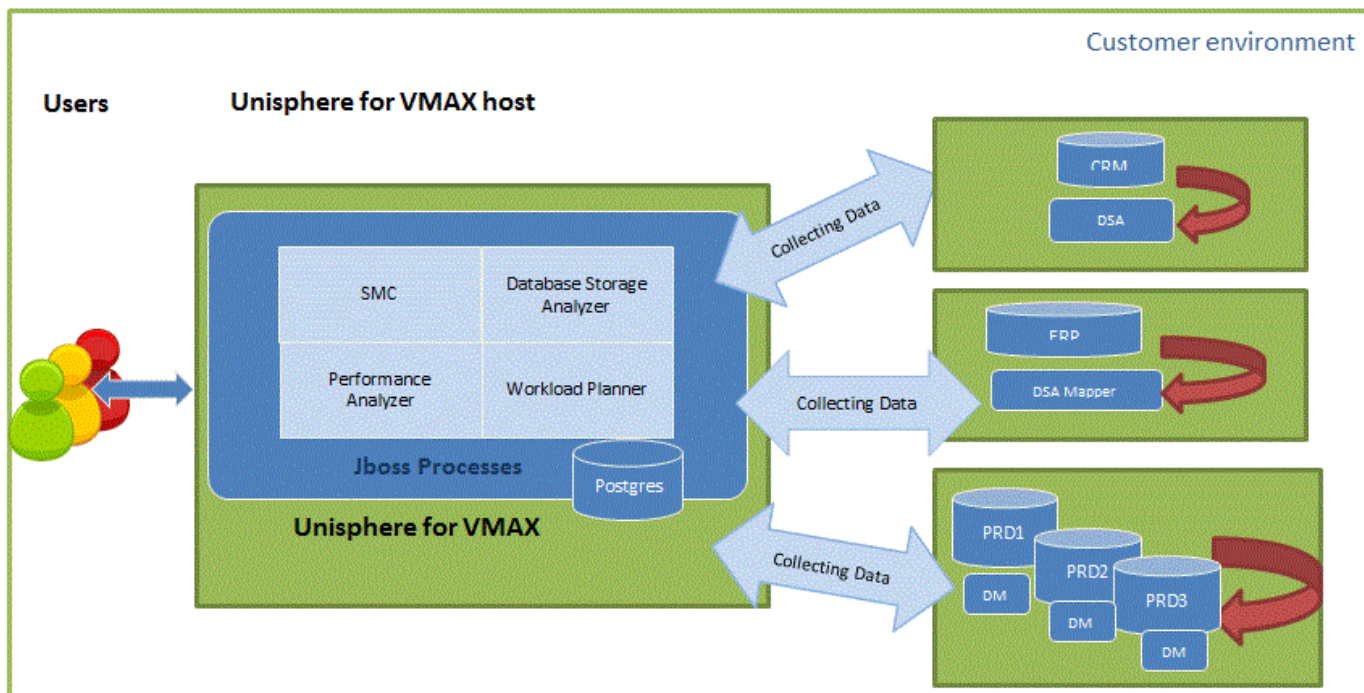
SQL Server support

With Version 8.1 DSA support for SQL server was added in a similar way as for Oracle so customers would have a high level DB/Storage correlation view as well as a deep dive view showing database sessions and objects correlated to their Symmetrix devices.

Note that this functionality is available for both VMAX2 and VMAX3 However, in this release hinting are not part of the SQL Server version.

Architecture

DSA runs in Unisphere under Jboss. It uses the same Postgres DB as a repository for its information as other Unisphere applications, such as Performance Analyzer (PA) and Workload Planner (WLP).



Database collection and retention policy

For Oracle

DSA collects information by connecting directly to the monitored database through a database user. This read-only user only has select permissions on a fixed list of Oracle dictionary tables.

For a detailed list of the tables being access by DSA, refer to the **Error! Reference source not found.** section.

DSA fetches data every 5 minutes and sends it back to the Unisphere repository database (Postgress DB) where it aggregates the data into hourly and daily aggregations. By default, DSA saves the fetched data for 15 days; however, you can extend this period to 30 days. DSA saves the hourly aggregations for 15 months and the daily aggregations for 2 years; however, you can extend both periods up to 3 years.

In addition to the 5min collection there is also a nightly process that accesses the database once a day at around 12:00am in order to update the dictionary information about objects and extents.

To connect to the monitored database, you need to open the database TNS port (usually 1521 or 1525) between the Unisphere repository server and the monitored database server.

For SQL Server

DSA collects information by connecting directly to the monitored database through a database user. DSA fetches data dictionary data and activity data directly from the SQL Server tables.

The user can be created during the installation process or DBAs can create it manually prior to running the “Add database” option.

The DSA user needs to have SYSADMIN privileges. Note that SYSADMIN role is required in order to collect object extent data using a command called DBCC EXTENTINFO. This process runs once a day.

However, DSA login can be created with read only privileges, yet any functionality associated with object collection would be disabled.

DSA can work with both “Windows Authentication Mode” and “SQL Server Authentication Mode” with either SQL Server “local” user or as windows authenticated user. In any case those users need to have SA permission.

Mapping files

The mapping process is responsible for mapping the Oracle or SQL Server files to the storage system devices. By default, the process runs once a week, however, you can configure it to run at different times.

During device mapping, the list of database files is copied using SSH to the monitored database host. A process executing on the monitored database host identifies the host physical devices associated with the Oracle files, and then sends the list back to be loaded into the DSA repository.

An executable called Mapper is copied to the remote server with a list of Oracle datafiles to map. The mapper contains a solution enabler library called SRMLite so

there is no need to install SE on the host. The Mapper uses the SYMAPI infrastructure and calls the same APIs as the following Solutions Enabler (SYMCLI) commands:

- `symrslv identify`—Identifies whether the storage object is a file, ASM file, logical volume or a host physical device (disk).
- `symhostfs`—Maps the oracle datafile to its filesystem device.
- `symlv show`—Maps a logical volume to its disks. In case the storage object is identified as an ASM file, this command inquires `symrslv pd` to get the storage device and array for a given disk.
- `symrslv pd`—Gets the storage device and array for a given disk.

For Oracle running on Linux/Unix - To run the above commands, DSA requires root or sudo user and an open SSH port between the Unisphere server and the database host. The installation program will prompt for the root password or a sudo user. This is an automated process that requires no manual intervention.

For SQL Server - In order to map database files to Symmetrix devices, the user needs to install a DSA Listener on the monitored environment. The listener should be manually installed before adding the new monitored database. In order to install the DSA listener on the monitored environment, go to the administration tab in DSA, click on “Add database” and then click **DSA Listener** to download it, copy it to the monitored environment and then follow the instructions in the readme.txt file. Note that the default port for the DSA listener is 22200 but it is configurable.

The password is kept encrypted in Unisphere using the standard Unisphere encryption method.

Impact on the monitored server: DSA should not impact database activity. In general, DSA may take up to 3% of one CPU.

Mapping on databases running on VMware virtual environment

In order to run the mapping on a database that is running on VMware virtual disks, make sure to add the virtual server to Unisphere. You do this by clicking **UI Hosts** => **Virtual Servers** on the **Unisphere for VMAX** GUI and then clicking **Add VM Server**. In addition, you need to set the parameter `disk EnableUUID = "TRUE"` in the ESX server. You do this by clicking **Edit VM settings** on the **vSphere** client.

Storage statistics collection

DSA gets its storage statistics directly from Performance Analyzer, thereby keeping the DSA and PA storage views consistent.

Support matrix

- Storage platforms: Symmetrix systems running Enginuity 5671 – 5876, or VMAX systems running HYPERMAX OS 5977 or higher
- Oracle version 10g or higher
- Operating systems (monitored database server for Oracle):
 - AIX version 5.2 or higher (64-bit)
 - SUN Solaris 10 or higher (64-bit)
 - Linux Red Hat 5 or higher (64-bit)
 - SUSE Linux Enterprise Server
 - Oracle Linux 5 or higher
 - HP UX Itanium version 11.23 or higher
- The monitored database must have an Oracle diagnostic pack license.
- SQL Server 2008 or higher
- Virtual environment: VMware Virtual disks VMDK & RDM. Other virtual environments (Solaris containers, AIX VIO, VPLEX and others) are not supported.
- 10GB is required on the monitored database host for storing the mapping output.

Creating guest user

The following example illustrates the commands used by DSA to create the DSA database guest user.

| Note: there is no need to manually run this script to create the database user. Instead, it can be run as part of the add environment process. This can be done via Unisphere for VMAX console. For instructions, refer to Unisphere help system.

It is recommend that you use the most up to date script from DSA as opposed to taking it from here.

For Oracle

-- Creation script of EMC DBC's guest user.

```
-- Creation script of EMC DBC's guest user.

-- Before running the script please:
-- Replace 'EMC_Unisphere_USER' with the real name of
the Database guest user
-- Replace 'EMC_Unisphere_PASS' with the user's
password

declare
  sqlStr varchar2(200);
begin
  sqlStr := 'create user EMC_Unisphere_USER identified by
EMC_Unisphere_PASS';

  execute immediate sqlStr;
end;
/

--Create the new user
--create user EMC_Unisphere_USER identified by
EMC_Unisphere_USER_PASS;
grant connect to EMC_Unisphere_USER;
GRANT SELECT ON dba_segments TO EMC_Unisphere_USER;
GRANT SELECT ON dba_tab_partitions TO EMC_Unisphere_USER;
GRANT SELECT ON dba_tab_subpartitions TO
EMC_Unisphere_USER;
GRANT SELECT ON dba_tables TO EMC_Unisphere_USER;
GRANT SELECT ON dba_ind_columns TO EMC_Unisphere_USER;
GRANT SELECT ON dba_part_key_columns TO EMC_Unisphere_USER;
GRANT SELECT ON dba_data_files TO EMC_Unisphere_USER;
GRANT SELECT ON dba_cons_columns TO EMC_Unisphere_USER;
GRANT SELECT ON dba_tab_columns TO EMC_Unisphere_USER;
GRANT SELECT ON dba_indexes TO EMC_Unisphere_USER;
GRANT SELECT ON dba_part_tables TO EMC_Unisphere_USER;
GRANT SELECT ON dba_col_comments TO EMC_Unisphere_USER;
GRANT SELECT ON dba_lobs TO EMC_Unisphere_USER;
GRANT SELECT ON dba_free_space TO EMC_Unisphere_USER;
GRANT SELECT ON dba_temp_files TO EMC_Unisphere_USER;
GRANT SELECT ON dba_tablespaces TO EMC_Unisphere_USER;
GRANT SELECT ON dba_tab_comments TO EMC_Unisphere_USER;
GRANT SELECT ON obj$ TO EMC_Unisphere_USER;
GRANT SELECT ON syn$ TO EMC_Unisphere_USER;
GRANT SELECT ON view$ TO EMC_Unisphere_USER;
GRANT SELECT ON user$ TO EMC_Unisphere_USER;

GRANT SELECT ON v_$tempfile TO EMC_Unisphere_USER;
```

Creating guest user

```
GRANT SELECT ON v_$event_name TO $EMC_Unisphere_USER;
GRANT SELECT ON v_$sqlarea TO $EMC_Unisphere_USER;
GRANT SELECT ON v_$database TO $EMC_Unisphere_USER;
GRANT SELECT ON V_$PARAMETER TO $EMC_Unisphere_USER;
GRANT SELECT ON v_$instance TO $EMC_Unisphere_USER;

GRANT SELECT ON gv_$instance TO $EMC_Unisphere_USER;
GRANT SELECT ON gv_$database TO $EMC_Unisphere_USER;
GRANT SELECT ON gv_$filestat TO $EMC_Unisphere_USER;
GRANT SELECT ON gv_$tempfile TO $EMC_Unisphere_USER;
GRANT SELECT ON gv_$tempstat TO $EMC_Unisphere_USER;
grant select on gv_$system_event to $EMC_Unisphere_USER;
GRANT SELECT ON gv_$active_session_history TO
$EMC_Unisphere_USER;

GRANT SELECT ON gv_$segstat TO $EMC_Unisphere_USER;

GRANT SELECT ON dba_hist_active_sess_history TO
$EMC_Unisphere_USER;

GRANT SELECT ON dba_extents TO $EMC_Unisphere_USER;

-- These permission are required for the script
GRANT SELECT ON DBA_HIST_FILESTATXS TO $EMC_Unisphere_USER;
GRANT SELECT ON DBA_HIST_TEMPSTATXS TO $EMC_Unisphere_USER;
GRANT SELECT ON dba_hist_snapshot TO $EMC_Unisphere_USER;
GRANT SELECT ON ts$ TO $EMC_Unisphere_USER;
GRANT SELECT ON sys_objects TO $EMC_Unisphere_USER;
GRANT SELECT ON seg$ TO $EMC_Unisphere_USER;
GRANT SELECT ON file$ TO $EMC_Unisphere_USER;

--10 and up
GRANT SELECT ON gv_$waitclassmetric_history TO
$EMC_Unisphere_USER;
GRANT SELECT ON gv_$system_wait_class TO
$EMC_Unisphere_USER;
GRANT SELECT ON gv_$sysmetric_history TO
$EMC_Unisphere_USER;

GRANT SELECT ON GV_$LOGFILE TO $EMC_Unisphere_USER;
GRANT SELECT ON GV_$LOG TO $EMC_Unisphere_USER;

GRANT SELECT ON GV_$PARAMETER TO $EMC_Unisphere_USER;
GRANT SELECT ON gv_$services TO $EMC_Unisphere_USER;

GRANT SELECT ON dba_part_indexes TO $EMC_Unisphere_USER;
GRANT SELECT ON dba_ind_partitions TO $EMC_Unisphere_USER;
GRANT SELECT ON dba_ind_subpartitions TO
$EMC_Unisphere_USER;

GRANT SELECT ON gv_$filestat TO $EMC_Unisphere_USER;
GRANT SELECT ON gv_$sysstat TO $EMC_Unisphere_USER;
```



```

-- Create synon
create synonym $EMC_Unisphere_USER.dbc_obj$ for obj$;
create synonym $EMC_Unisphere_USER.dbc_syn$ for syn$;
create synonym $EMC_Unisphere_USER.dbc_view$ for view$;
create synonym $EMC_Unisphere_USER.dbc_user$ for user$;

create synonym $EMC_Unisphere_USER.dbc_ts$ for ts$;
create synonym $EMC_Unisphere_USER.dbc_sys_objects for
sys_objects;
create synonym $EMC_Unisphere_USER.dbc_seg$ for seg$;
create synonym $EMC_Unisphere_USER.dbc_file$ for file$;

--10,11 only
declare
  x number;
begin
  select substr(version,0,2) into x from v$instance;
  if (x < 12) then
    execute immediate 'create or replace view dbc_$kccle as
select * from x$kccle';
    execute immediate 'grant select on dbc_$kccle to
$EMC_Unisphere_USER';
    execute immediate 'create or replace synonym
$EMC_Unisphere_USER.dbc_$kccle for sys.dbc_$kccle';
  end if;
end;
/

-- 12 only
declare
  x number;
  v_name varchar (200);
begin
  select substr(version,0,2) into x from v$instance;
  if (x >= 12) then
    execute immediate 'select distinct cdb from
gv$database' into v_name;
    if (v_name = 'YES') then
      execute immediate 'grant select on gv_$pdbs to
$EMC_Unisphere_USER';
      execute immediate 'select distinct name from gv$pdbs
where name = ''PDB$SEED''' into v_name;
      execute immediate 'alter user $EMC_Unisphere_USER
SET CONTAINER_DATA = all CONTAINER = CURRENT';
    end if;
  end if;
  exception when NO_DATA_FOUND then
    null;
end;

```

Creating guest user

```
/
--for 11g and above only
N
declare
  x number;
begin
  select substr(version,0,2) into x from v$instance;
  if (x >= 11 ) then
    --execute immediate 'grant SELECT ON gv_$iostat_function
TO $EMC_Unisphere_USER';
    execute immediate 'GRANT SELECT ON gv_$iostat_file TO
$EMC_Unisphere_USER';
  end if;
end;
/

create or replace view $EMC_Unisphere_USER.dbc_segments as
select o.*,
-- Get table name - If the object is a table or
cluster(including partitions) we use its name, otherwise we
use the table name from the index join or lob join
  case when segment_type like '%TABLE%' or
segment_type='CLUSTER' OR OBJECT_ID<0 then o.SEGMENT_NAME
else nvl(i.table_name, l.table_name) end table_name,
-- Get owner - If the object is a table or cluster(including
partitions) we use its owner, otherwise we use the table
name from the index join or lob join
  case when segment_type like '%TABLE%' or
segment_type='CLUSTER' OR OBJECT_ID<0 then o.owner else
nvl(i.owner, l.owner) end table_owner,
-- Get table id - If the object is a table or cluster we use
its id
-- If the object is a partition we find the id using
analytic function from the dbc_obj$ table
-- If the object is a index of some sort we get it from the
joined index query
-- If the object is a lob of some sort we get it from the
joined lob query
  case when segment_type like '%TABLE' or
segment_type='CLUSTER' OR OBJECT_ID<0 then o.object_id
when segment_type like 'TABLE%PARTITION%' then
  first_value(object_id) over(partition by
o.segment_name, o.owner,
  case when segment_type like '%TABLE%' then 1
when segment_type like '%INDEX%' then 2
when segment_type like '%LOB%' then 3
else 4
end
order by decode(segment_type,'TABLE',1,2))
when segment_type like '%INDEX%' and i.table_id is not
null then i.table_id
when segment_type like 'LOB%' then l.table_id
end table_id,
```

```

-- Get parent object_id
  case when segment_type like '%PARTITION%' then
    first_value(object_id) over(partition by o.segment_name,
o.owner,
  case when segment_type like '%TABLE%' then 1
  when segment_type like '%INDEX%' then 2
  when segment_type like '%LOB%' then 3
  else 4
  end)
  else
    o.object_id
  end parent_object_id,
  case when column_name is not null then column_name
  when segment_type like '%INDEX%' then (select
column_name
                                     from      dba_lobs
1                                     where
1.table_name=i.table_name
                                     and
1.index_name=o.segment_name
                                     and
1.owner=o.owner)
  else null end lob_column_name
from
  (select o.obj# object_id, o.dataobj# data_object_id,
s.OWNER, S.SEGMENT_NAME,s.segment_type, S.PARTITION_NAME,
  S.TABLESPACE_NAME,nvl(S.BYTES/1024/1024,0)
SEGMENT_SIZE      , o.ctime creation_date, O.mtime
LAST_DDL_TIME, s.extents
  from      dba_segments s,
  $EMC_Unisphere_USER.dbc_obj$ o,
  $EMC_Unisphere_USER.dbc_user$ u
  where o.name= s.segment_name
  and o.name not like 'BIN$%'
  and      u.user#=o.owner#
  and      u.name= s.owner
  and
nvl(o.subname,'*****')=nvl(s.partition_name,'*****')
  and      (
  (s.segment_type IN ('TABLE', 'NESTED TABLE') and
o.type#=2)
  or
  (s.segment_type IN ('LOBINDEX', 'INDEX') and
o.type#=1)
  or
  (s.segment_type IN ('CLUSTER') and o.type#=3)
  or
  (s.segment_type IN ('TABLE PARTITION') and
o.type#=19)
  or

```

Creating guest user

```
        (s.segment_type IN ('INDEX PARTITION') and
o.type#=20)
        or
        (s.segment_type IN ('TABLE SUBPARTITION') and
o.type#=34)
        or
        (s.segment_type IN ('INDEX SUBPARTITION') and
o.type#=35)
        or
        (s.segment_type IN ( 'LOBSEGMENT') and o.type#=21)
        or
        (s.segment_type IN ( 'LOB PARTITION') and
o.type#=40)
        or
        (s.segment_type IN ( 'LOB SUBPARTITION') and
o.type#=41)
    )
    UNION ALL
    select o.obj# object_id,o.dataobj# data_object_id,
u.name owner, o.name SEGMENT_NAME,decode(o.type#, 1,
'INDEX', 2,'TABLE', 21, 'LOBSEGMENT') segment_type, null
PARTITION_NAME,

null TABLESPACE_NAME,0 SEGMENT_SIZE, o.ctime creation_date,
O.mtime LAST_DDL_TIME, null extents
    from          $EMC_Unisphere_USER.dbc_obj$ o,
        $EMC_Unisphere_USER.dbc_user$ u
    where
        o.dataobj# is null
        and o.name not like 'BIN$%'
        and u.user#=o.owner#
            and o.type# IN (1,2,21)
    ) o
-- Join to index query (get table_name+table_owner+table_id
from index table)
    left outer join (select i.index_name, i.owner,
i.table_name, o.obj# table_id
        from          $EMC_Unisphere_USER.dbc_obj$
o,
        $EMC_Unisphere_USER.dbc_user$ u,
        dba_indexes i
    where      i.table_name=o.name
        and      i.owner=u.name
        and      linkname is null
        and      u.user#=o.owner#
            and
o.type# in(2,3)) i on (i.index_name=o.SEGMENT_NAME and
i.owner=o.owner)
-- Join to lob query (get table_name+table_owner+table_id
from lob table)
    left outer join (select l.segment_name, l.owner,
l.table_name, o.obj# table_id, column_name
        from          $EMC_Unisphere_USER.dbc_obj$
o,
```

```

        $EMC_Unisphere_USER.dbc_user$ u,
        dba_lobs l
    where    l.table_name=o.name
            and    l.owner=u.name
            and    linkname is null
            and    u.user#=o.owner#
            AND
O.TYPE# IN(2,3)) L ON (L.SEGMENT_NAME=O.SEGMENT_NAME AND
L.OWNER=O.OWNER);
/

```

For SQL Server

```

-- Creation script of EMC EMC_Unisphere_USER guest user.
create login $EMC_Unisphere_USER WITH PASSWORD =
'$EMC_Unisphere_PASS';

Grant view server state to $EMC_Unisphere_USER;
/

Grant view any definition to $EMC_Unisphere_USER;
/

--To enable monitoring of object level data, EMC Unisphere
DSA uses the 'dbcc extentinfo' function.
--The 'dbcc extentinfo' requires SA privileges.
EXEC sp_addsrvrolemember '$EMC_Unisphere_USER', 'sysadmin';

/*

--IF not using sa then we should use this:
DECLARE
    @LoginName VARCHAR(128),
    @UserName VARCHAR(128),
    @DatabaseName VARCHAR(100),
    @SQLScript VARCHAR(6000)

SET @LoginName = '$EMC_Unisphere_USER' -- Can be Windows
login such as 'DOMAINLoginName'
SET @UserName = '$EMC_Unisphere_USER' -- Many
times the same as the LoginName

BEGIN
    DECLARE DatabaseCursor CURSOR FOR
        SELECT [name] from sys.databases where state
= 0
        ORDER BY [name]

```

Creating guest user

```
OPEN DatabaseCursor
FETCH NEXT FROM DatabaseCursor INTO @DatabaseName

WHILE @@FETCH_STATUS = 0
    BEGIN

        PRINT 'Updating Database: ' +
@DatabaseName + ' '

        SET @SQLScript = '
            USE [' + @DatabaseName + ' ]
            IF NOT EXISTS (SELECT * FROM
[' + @DatabaseName + ' ] ..sysusers WHERE [name] = ''' +
@UserName + ''')
            EXEC SP_ADDUSER '''
+ @LoginName + ''',''' + @UserName + ''''

        -- PRINT @SQLScript
        EXEC (@SQLScript)

        FETCH NEXT FROM DatabaseCursor INTO
@DatabaseName

    END

CLOSE DatabaseCursor
DEALLOCATE DatabaseCursor
END;

*/
```