Logging at JHU

This document describes usage log generation and collection (harvesting) activities at JHU. Usage logging has been used extensively for the Sloan Digital Sky Survey (SDSS) SkyServer since its inception in 2001. A thorough analysis of the first 5 years of SkyServer usage logs is documented in the "SkyServer Traffic Report: The First Five Years" by Singh et al. 2006 (http://research.microsoft.com/apps/pubs/default.aspx?id=64520). The logging philosophy and basic framework developed for SDSS was then used to create a Web hits logging system for the National Virtual Observatory (NVO), which has now been succeeded by the Virtual Astronomical Observatory (VAO). The following sections describe both these logging systems. There are two kinds of usage logging that are of interest for most applications and purposes: Web hits and service requests, so each of these types of logging will be described for the SDSS and NVO/VAO projects.

Logging Web Hits

The Web hits log, or Weblog for short, records each time a Web page is visited by a user. This is basic information that can be used to assess who is accessing the site, which pages (URLs) they are accessing and how often, and whether Web pages are being viewed successfully or not (errors are logged). The actual logs of Web hits are recorded by the Webserver software itself, e.g., IIS in Windows or Apache in LINUX.

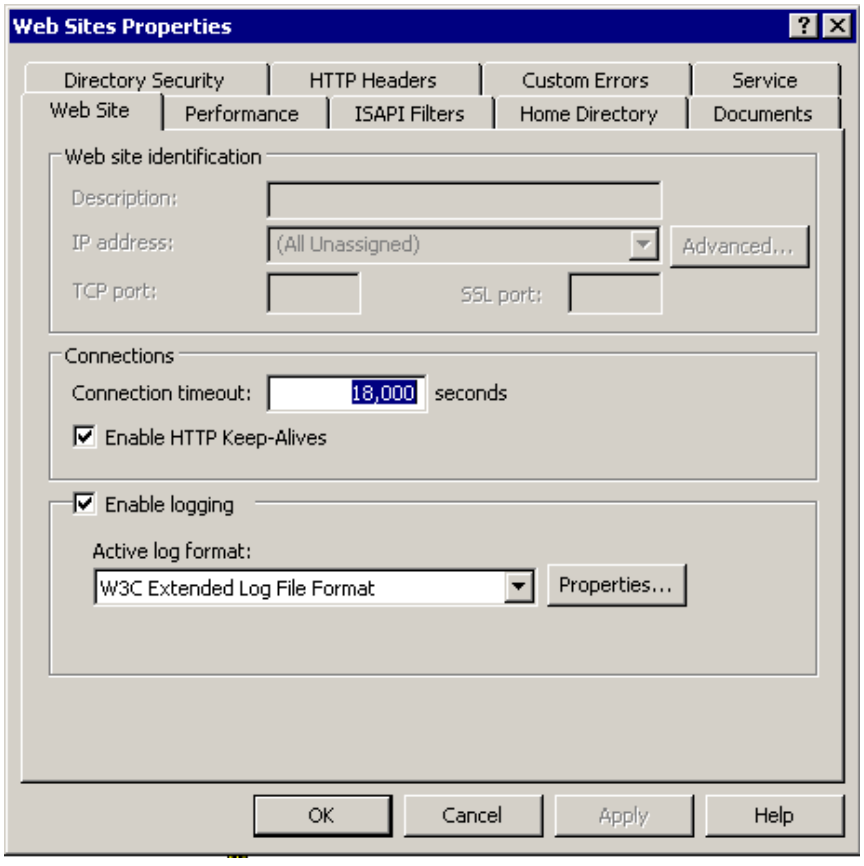

Figure 1. IIS Web Sites Properties dialog for setting up the Web hits logging. The "Enable Logging" option must be checked, and the actual fields to be recorded in the log are configured by clicking on the Properties for the W3C Extended Log File Format.

Configuring Web Hit Logs on Windows

For Windows Webservers, the

information that you wish IIS to record for Website visits is configured by right-clicking on the "Web Sites" tab under the main IIS tab in the Computer or IIS Manager and selecting Properties. In the Web Sites Properties dialog (Figure 1), make sure the "Enable logging" option is checked and the Active log format is set to "W3C Extended Log File Format", then click on the Properties button for the Active log format. The new log file generation schedule is set up in the General tab. A new log file should be generated every day at midnight, so the option selected for the New log schedule should be Daily.

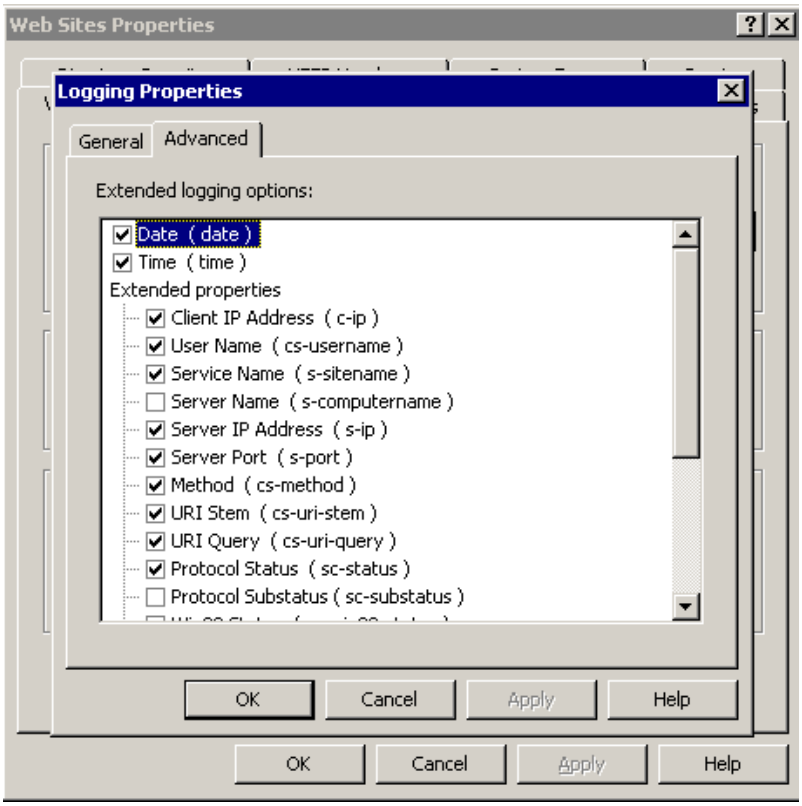

Figure 2. Setting up the information to be logged for each Web hit in the Log gingProperties Advanced tab. The new log file schedule is set up in the General tab. It should be set to "Daily" at midnight.

In order to set up the actual information items to be logged for each Web hit, you need to select the Advanced tab in the Logging Properties window and then check the properties to be logged. Figures 2 and 3 show the various properties that should be checked so that they are logged for each Web hit. In general, we want to record the date and time of the hit, all the information about the user (client IP, name etc.), the server that serves the Web page (server IP, port), the URL corresponding to the page, the access method (GET/POST), any errors encountered, the bytes sent/received, how long it took to load the page and the browser (user agent) information.

The rest of the information should remain unchanged (set to default values), e.g. the name and location of the daily log file. IIS will deposit the daily log files in the given location from which the log harvester will copy them and parse them for ingest into the harvester database (see below). The network paths of the log files will need to be entered into the LogSource table described below so that the files can be copied and harvested by the harvester scripts.

Service Logging

Service logging is the process of recording the usage of individual service requests, e.g., SQL queries, image or binary file download requests, Web service requests.

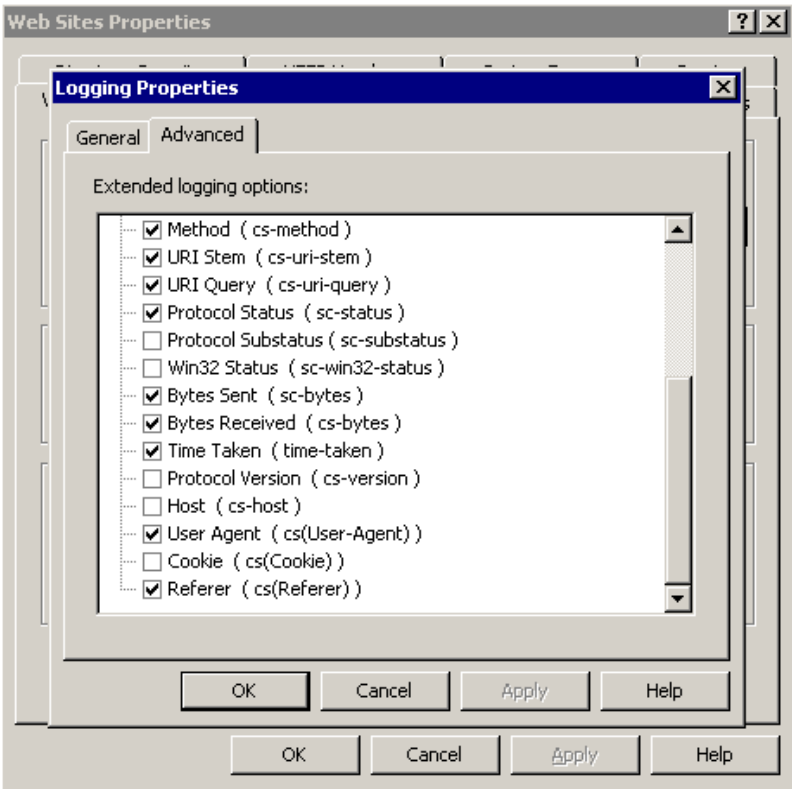

Figure 3. The Logging Properties showing the other information items that should be logged for each Web hit.

This is much more detailed and useful information than just logging the Web hits. In general, several Web hits may correspond to a single service request, but a single service request may actually propagate through multiple services and sites. Information about how a service request is handled is potentially much more valuable and useful than just Web hits.

Unlike Weblogging, service logging cannot be done using existing OS tools or services, but rather must be explicitly included in the software that provides the service. For example, each time a user submits a query to an online query portal like the SDSS SkyServer, a special SQL stored procedure is invoked that writes a record to a SQL log table in the log database. As such, it is not easy to retroactively include service logging in an existing service. It is much better to include service logging in the design of the service itself.

The information attributes that should be logged for each service request include the following: unique id of request, data, time,

SDSS Logging

The logging framework for SDSS was originally designed and developed by Jim Gray (Microsoft Research) and Alex Szalay (JHU), and it includes both Web hits and SQL query logging. The logging schema and scripts are part of the sqlLoader product and are contained in the schema/log/ directory of sqlLoader. There are two DDL (data definition language) script files here: **`weblogDBCreate.sql`** (see Appendix A) and **`weblogHarvesterCreate.sql`** (Appendix B). The first is used to create a **weblog** database on each server that hosts an SDSS database, and the second is used

to create a special **weblog** database that is used to harvest the contents of all the other weblog databases. The regular weblog database on each SDSS server is where log entries are written to in real time by the query tools that access that server.

Log Generation

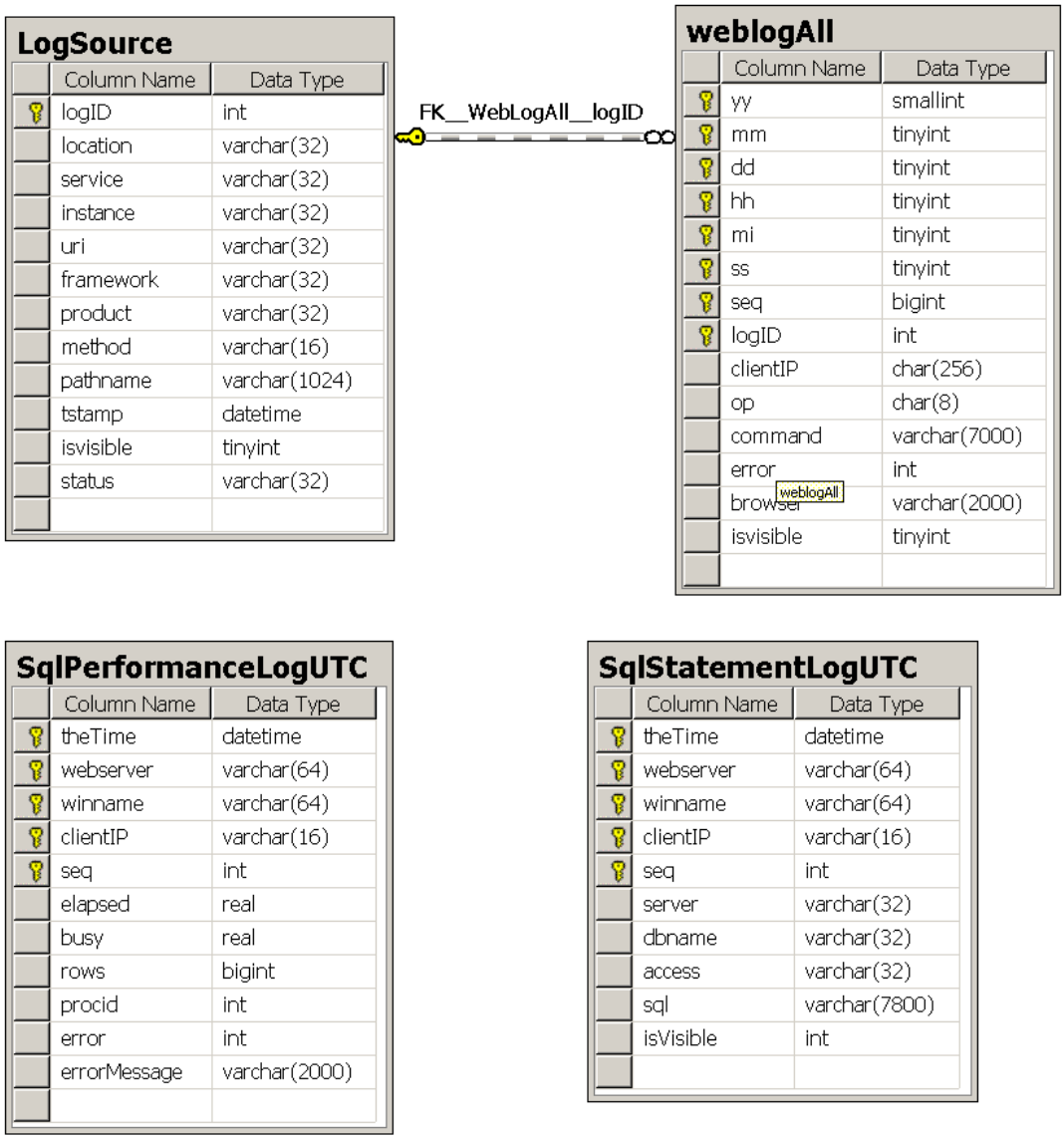

Figure 4. Schema diagram for WebLog DB on an SDSS server. These logs are harvested by the Log Harvester.

This is the process of generating the records that log activity of a particular kind. Figure 4 shows the schema for a WebLog database on an individual SDSS database server at an SDSS site. Although there are LogSource and WebLogAll tables in the schema for each instance of the WebLog database,, they are not really used unless that server also harvests the local logs at that site. Typically, all the logs for a given

remote site are locally collected on one designated server and then harvested by the harvester at JHU.

For the log generation on each SDSS server, only the log entries for the SQL queries handled by each server are deposited into the SqlStatementLogUTC and SqlPerformanceLogUTC tables of the WebLog DB on that server (see below). The Web hits for the site are ingested into the WebLogAll table on the local harvester

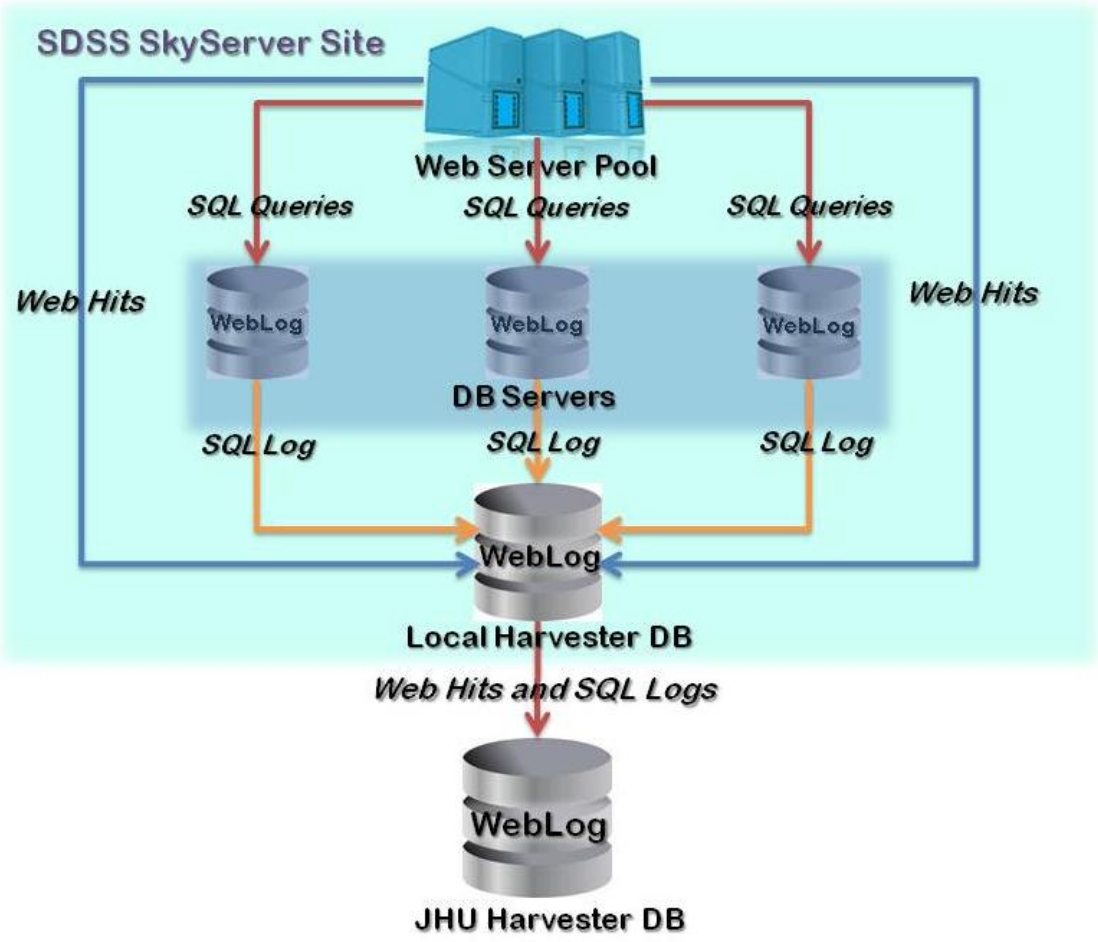

Figure 5. Web hits and SQL query logging at an SDSS site, and harvesting to JHU.

directly from the Webserver(s). This is shown in Figure 5.

The SQL queries are inserted into the appropriate WebLog tables by the stored procedure that executes each user query – **spExecuteSQL** (see Appendix C). This is a two-step process: an entry is inserted into the SqlStatementLogUTC before the query is launched, and once the query execution completes (with or without an error) an entry is inserted into the SqlPerformanceLogUTC with the same timestamp as the first entry. This is done to ensure that the two entries can be matched to the same query submitted by the user and combined into a single entry for each query in the SqlLog and SqlLogAll views. These are the views that you

would normally use to analyze the log data. The second entry in the performance log table includes query performance information such as the elapsed time, CPU (busy) time, the number of rows returned by the query and any error codes and messages.

Two additional stored procedures – **spLogSqlStatement** and **spLogSqlPerformance** – are also provided for convenience in the listing in Appendix C. These may be used to directly write the SQL query entries to the WebLog tables if for any reason the spExecuteSQL procedure is not used to handle user queries.

Log Harvesting

This is the process of collecting the logs from each Webserver at each SDSS site that is harvested, and ingesting that information into the harvester database. Logs may in principle be collected from anywhere in the world (including SDSS mirror sites), but in practice to date the only non-JHU logs harvested have been the FNAL logs, since FNAL was the primary archive center for SDSS-I and SDSS-II.

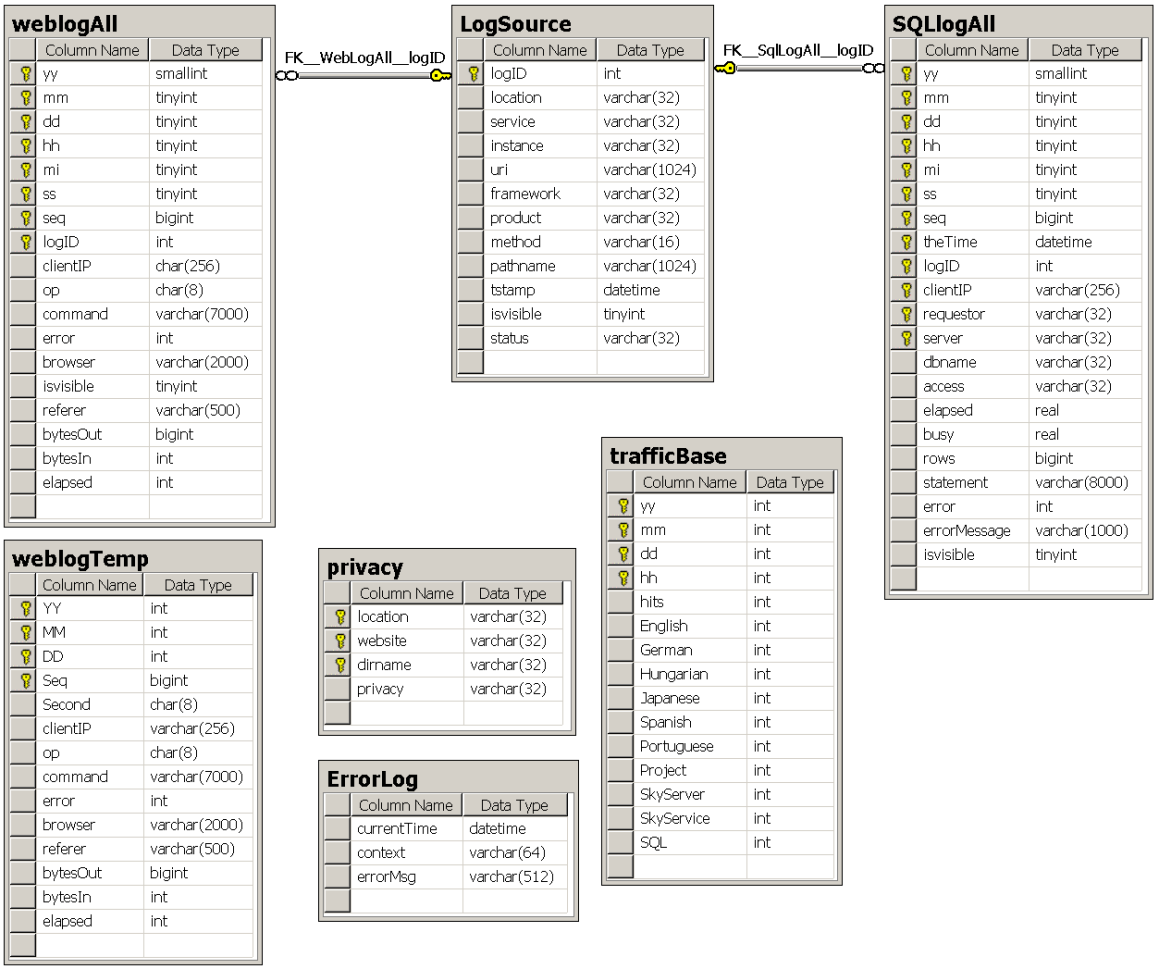

Figure 6. Schema diagram for the Log Harvester WebLog database.

The harvester database maintains a list of all the log sources in the LogSource table that includes all the information necessary to connect to each log's location and harvest the log data, such as the retrieval `method` and the physical path (`pathname`) to the log data. The method can be either TSQL or XCOPY currently depending on whether the log data is in a SQL Server database (as in the case of SQL query logs or Web logs that have already been locally harvested at remote sites) or in files that need to be copied (as in the case of local Web logs). A third method, WGET, is not yet implemented but is meant for retrieving remote logs over HTTP.

For logs retrieved by the XCOPY method, the value of the `pathname` column is the actual network path of the directory containing the log files. For logs retrieved by the TSQL method, `pathname` is set to the fully qualified database name of the WebLog database containing that log.

Appendix A – sqlLoader/schema/log/webLogDBCreate.sql
This file contains the source for scripts and DDL for the WebLog database that is created on each SDSS server that handles user queries.

```
-------------------------------------------------
-- webLogDBCreate.sql
-- 2003-10-03 Jim Gray, Alex Szalay
-------------------------------------------------
-- Modificatdions:
-- 2003-10-04 Alex: Added extra columns to weblog
-- 2003-10-04 Alex: Added extra columns to Sql***Log
-- 2003-10-26 Jim:  went from null to default
--                    change "null" to "not null default 0"
--                                  Weblog.error
--                                  weblog.isVisible
--                    change "null" to "not null default ''"       for
--                                  Weblog.second
--                                  Weblog.framework
--                                  Weblog.product
--                                  Weblog.op
--                                  Weblog.command
--                                  Weblog.browser
--                    Weblog.clientIP  not null default value 0.0.0.0
--                    let the command length be 7KB to catch odd things.
--                    permuted weblog columns (book-keeping last)
--                           added hh, mi, ss to weblog and dropped sec.
--                    Added WeblogAll base table (to replace weblog)
--                    Made WebLog a view.
-- 2003-10-26 Alex: Added LogSource and privacy table
-- 2003-10-26 Jim:  Added weblogTemp table
-- 2003-10-29 Jim:  changed weblog.op from char(6) to char(8)
--                 changed weblog.command from char(256) to char(7000)
--                 changed weblog.browser from char(256) to char(2000)
--                 added weblog.hh, mi, ss and dropped second
--                 added LogSource.weblog -- the name of the log file == name of web
server
--                 added TrafficBase.SkyServer
--                 added TrafficBase.SkyService
--                 changed WebLog to WebLogAll and added WebLog view
--                  added SqlLogAll and converted all sql logging to UTC
--                 added spCopySqlLog and spCopyWebLog
--                 fixed spUpdateTrafficBase to be simpler and to track SQL counters.
--                 appended populating logSource table.
--                 extensive rewrite of code to be driven by LogSource table.
-- 2004-04-01 Ani:  Updated schema of WebLogAll, WebLogTemp and SqlLogAll tables to bump
up
--                  seq to bigint (from int) and clientIP to varchar(256) (from
char(16)).
--
-- 2004-05-19 Ani:  Added creation of Weblog DB.
-- 2004-11-17 Ani:  Replaced bcpWeblog.js call with bcpWeblog.exe (C#
--                  version) in spCopyWeblogs.
-- 2004-12-02 Ani:  Made small change to DailyTraffic view so the month
--                  is zero-padded on the left and hence the descending
--                  order in the daily traffic display works correctly.
-- 2005-02-22 Ani:  Added totalIO to SqlLogAll table.
-- 2005-04-13 Ani:  Added new weblog columns to InsertFNALWeblog and
--                  cast the referer to varchar(500) pending a change
--                  to WebLogAll to expand it to bigger size.
-- 2005-09-07 Ani:  Copied original version of this file to
--                  webLogHarvesterCreate.sql, and took the harvester
--                  schema out of here so this script can be run to
--                  create a new weblog DB on a production SDSS server.
-- 2005-09-07 Ani:  Fixed bug (syntax error) in create db call.
-- 2005-11-09 Ani:  Removed data file creation on C: drive since this was
--                  causing problems at FNAL.  Entire DB is on D: now.
-- 2006-01-18 Ani:  Added seq identity column to sql logs to guarantee
--                  PK uniqueness and avoid PK violations for queries
--                  submitted in quick succession (PR # 6809).
```

```
--* 2006-06-19 Ani: Moved spLogSqlStatement and spLogSqlPerformance here
--*                 from ../sql/spWebSupport.sql.
--* 2011-01-11 Ani: Updates for SDSS-III servers, changed test to skyuser.
-----------------------------------------------------
-- Create WebLog DB
CREATE DATABASE [weblog]  ON
  (NAME = N'weblog_Data', FILENAME = N'C:\data\data1\sql_db\weblog_Data.MDF' , SIZE =
2000, FILEGROWTH = 10%)
  LOG ON (NAME = N'weblog_Log', FILENAME = N'C:\data\data1\sql_db\weblog_Log.LDF' , SIZE
= 1000, FILEGROWTH = 10%)
  COLLATE SQL_Latin1_General_CP1_CI_AS
GO

exec sp_dboption N'weblog', N'autoclose', N'false'
GO
exec sp_dboption N'weblog', N'bulkcopy', N'false'
GO
exec sp_dboption N'weblog', N'trunc. log', N'true'
GO
exec sp_dboption N'weblog', N'torn page detection', N'true'
GO
exec sp_dboption N'weblog', N'read only', N'false'
GO
exec sp_dboption N'weblog', N'dbo use', N'false'
GO
exec sp_dboption N'weblog', N'single', N'false'
GO
exec sp_dboption N'weblog', N'autoshrink', N'false'
GO
exec sp_dboption N'weblog', N'ANSI null default', N'false'
GO
exec sp_dboption N'weblog', N'recursive triggers', N'false'
GO
exec sp_dboption N'weblog', N'ANSI nulls', N'false'
GO
exec sp_dboption N'weblog', N'concat null yields null', N'false'
GO
exec sp_dboption N'weblog', N'cursor close on commit', N'false'
GO
exec sp_dboption N'weblog', N'default to local cursor', N'false'
GO
exec sp_dboption N'weblog', N'quoted identifier', N'false'
GO
exec sp_dboption N'weblog', N'ANSI warnings', N'false'
GO
exec sp_dboption N'weblog', N'auto create statistics', N'true'
GO
exec sp_dboption N'weblog', N'auto update statistics', N'true'
GO
GO

------------------------
-- Create Web Log Tables
------------------------
USE WebLog

-------------------------------------------------------------
if exists (select * from dbo.sysobjects where name =(N'LogSource'))
       drop table LogSource
GO
--
CREATE TABLE LogSource (
-------------------------------------------------------------
--/H The basic information about sites to be harvested
--
--/T The entries here can correspond to websites and database servers
--/T The weblogs of active servers are copied to the WebLog database every hour
--/T and summary statistics are computed.
--/T The tstamp field gives the time of the most recent update.
-------------------------------------------------------------
```

```
       logID          int           NOT NULL DEFAULT(0),    --/D unique ID of log used as
foreign key
       location       varchar(32)   NOT NULL DEFAULT (''), --/D the location of the site
(FNAL, JHU,..
       service        varchar(32)   NOT NULL DEFAULT (''), --/D type of service
(SKYSERVER, SKYSERVICE, SKYQUERY,...)
       instance       varchar(32)   NOT NULL default (''), --/D The log underneath the
service (V1, V2,.. )
       uri            varchar(32)   NOT NULL default (''), --/D The url or other ID for
this service.
       framework      varchar(32)   NOT NULL DEFAULT (''), --/D the calling framework
(ASP,ASPX,HTML,QA,SOAP,...)
       product        varchar(32)   NOT NULL DEFAULT (''), --/D the type of product
acessed (EDR, DR1, DR2,...
       method         varchar(16)   NOT NULL default(''),  --/D Harvesting method
(XCOPY|WGET|SQL)
       pathname       varchar(1024) NOT NULL default(''),  --/D The path of the log
LogSource (UNC|URL)
       tstamp         datetime      NOT NULL default Current_timestamp,    --/D The time
of the last harvesting
       isvisible      tinyint       NOT NULL default(1),    --/D should this log be
visible in the event log (0:no, 1:yes)
       status         varchar(32)   NOT NULL default(''),  --/D The current state
(ACTIVE|DISABLED)
       PRIMARY KEY(logID)
)
GO
ALTER TABLE LogSource ADD CONSTRAINT [ak_LogSource] UNIQUE
       (location, service, instance)
GO

--============================================================
if exists (select * from dbo.sysobjects where name = N'weblogAll')
       drop table weblogAll
GO
--
CREATE TABLE weblogAll (
-----------------------------------------------------------------
--/H The weblog information -- contains both visible and invisible log records.
--
--/T The information is parsed from the W3C format weblog
--/T files, generated on each web server by IIS. A record is
--/T considered to be invisible until its flag is set to 1.
-----------------------------------------------------------------
       yy             smallint       NOT NULL,      --/D the year of the event
       mm             tinyint        NOT NULL,      --/D the month of the event
       dd             tinyint        NOT NULL,      --/D the day of the event
       hh              tinyint       NOT NULL,      --/D the hour of the event
       mi             tinyint        NOT NULL,      --/D the minute of the event
       ss             tinyint        NOT NULL,      --/D the second of the event
       seq            bigint         IDENTITY(1,1), --/D sequence number to uniquify the
event
       logID          int            NOT NULL DEFAULT(0) foreign key references
LogSource(logID),             --/D unique ID of log foreign key LogSource.logID
       clientIP       char(256)              NOT NULL DEFAULT ('0.0.0.0') ,       --/D
the IP address of the client
       op             char(8)        NOT NULL DEFAULT (''),        --/D the operation
(GET,POST,...)
       command        varchar(7000)  NOT NULL DEFAULT (''),        --/D the command
executed
       error          int            NOT NULL DEFAULT (0) ,        --/D the error code if
any
       browser        varchar(2000)  NOT NULL DEFAULT (''),               --/D the
browser type
       isvisible      tinyint        NOT NULL DEFAULT (0),         --/D should this event
be visible (0:no, 1:yes)
       PRIMARY KEY (yy desc ,mm desc,dd desc,hh desc,mi desc,ss  desc,seq desc,logID)
)

GO
ALTER TABLE  weblogAll NOCHECK CONSTRAINT ALL
```

```
--


------------------------------------------------------------
-- Web Log Views
------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'weblog' )
        drop view weblog
GO

CREATE VIEW weblog  (
------------------------------------------------------------
--/H The weblog information -- contains visible log records.
--
--/T The information is parsed from the W3C format weblog
--/T files, generated on each web server by IIS.
------------------------------------------------------------
       yy             ,             --/D the year of the event
       mm             ,             --/D the month of the event
       dd             ,             --/D the day of the event
       hh              ,            --/D the hour of the event
       mi             ,             --/D the minute of the event
       ss             ,             --/D the second of the event
       logID          ,             --/D the log that this came from, foreign key:
LogSource.logID
       seq            ,             --/D sequence number
       clientIP       ,             --/D the IP address of the client
       op             ,             --/D the operation (GET,POST,...)
       command        ,             --/D the command executed
       error          ,             --/D the error code if any
       browser        ,             --/D the browser type
       location       ,             --/D the location of the site (FNAL, JHU,..
       service        ,             --/D type of service (SKYSERVER, SKYSERVICE,
SKYQUERY,...)
       instance       ,             --/D The log underneath the service (V1, V2,.. )
       uri            ,             --/D The url or other ID for this service.
       framework      ,             --/D the calling framework
(ASP,ASPX,HTML,QA,SOAP,...)
       product                      --/D the type of product acessed (EDR, DR1, DR2,...
) AS
SELECT  yy,mm,dd,hh,mi,ss, w.logID, seq, clientIP, op, command, error, browser,
       location, service, instance, uri, framework, product
       from WebLogAll w with(nolock) left outer join logSource ls on w.logID = ls.logID
       where w.isVisible = 1

GO


------------------------------------------------------------------------------
-- SQL log tables and views.
------------------------------------------------------------------------------

------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'SqlStatementLogUTC')
       drop table SqlStatementLogUTC
GO
--
CREATE TABLE SqlStatementLogUTC (
-----------------------------------------------
--/H The SQL statements submitted directly by end users (rather than website generated
ones)
--/U
--/T Records are inserted at the start of the query.
--/T At the end of the query, a corresponding SqlPerfomrnceLog record is generated.
-----------------------------------------------
       theTime        datetime       NOT NULL DEFAULT (getUTCdate()),--/D the timestamp
        webserver       varchar(64)   NOT NULL DEFAULT(''),         -- the url
       winname        varchar(64)    NOT NULL DEFAULT(''),         -- the windows name of
the server
        clientIP              varchar(16)    NOT NULL DEFAULT(''),         -- client IP
address
```

```
       seq            int            identity(1,1) NOT NULL,              -- sequence
number to guarantee uniqueness of PK
       server         varchar(32)    NOT NULL DEFAULT(''),         --/D the name of the
database server
       dbname         varchar(32)    NOT NULL DEFAULT(''),         --/D the name of the
database
       access         varchar(32)    NOT NULL DEFAULT(''),         --/D The website DR1,
collab,...
       sql            varchar(7800)  NOT NULL DEFAULT(''),         --/D the SQL statement
       isVisible      int            NOT NULL DEFAULT(1),          --/D flag says
statement visible on intenet
                                                                    --/D collab activity
is logged but not public
       PRIMARY KEY CLUSTERED (theTime,webserver,winname,clientIP,seq)
)
GO

--============================================================
if exists (select * from dbo.sysobjects where name = N'SqlPerformanceLogUTC')
       drop table SqlPerformanceLogUTC
GO
--
CREATE TABLE SqlPerformanceLogUTC (
------------------------------------------------
--/H The cost of the SQL statements submitted directly by end users
--/U
--/T When a query compleltes the time, row count,and other attributes of the query are
added to the log.
--/T The corresponding SqlQueryLog tells what the statement is and where it came from.
------------------------------------------------
       theTime        datetime       NOT NULL DEFAULT (getUTCdate()),      --/D the
timestamp
        webserver       varchar(64)   NOT NULL DEFAULT(''),         -- the url
       winname        varchar(64)    NOT NULL DEFAULT(''),         -- the windows name of
the server
        clientIP              varchar(16)    NOT NULL DEFAULT(''),         -- client IP
address
       seq            int            identity(1,1) NOT NULL,              -- sequence
number to guarantee uniqueness of PK
       elapsed        real           NOT NULL DEFAULT (0.0),              --/D the lapse
time of the query
       busy           real           NOT NULL DEFAULT (0.0),              --/D the total
CPU time of the query
       [rows]         bigint         NOT NULL DEFAULT (0),         --/D the number of
rows generated
       procid         int            NOT NULL DEFAULT(0),          --/D the processid of
the query
       error          int            NOT NULL DEFAULT(0),          --/D 0 if ok,
otherwise the sql error #, negative numbers are generated by the procedure
       errorMessage   varchar(2000)  NOT NULL DEFAULT(''),         --/D the error
message.
       PRIMARY KEY CLUSTERED (theTime,webserver,winname,clientIP,seq)
)
GO



--============================================================
if exists (select * from dbo.sysobjects where name = N'SqlLogUTC')
       drop view SqlLogUTC
GO
--
CREATE VIEW SqlLogUTC
------------------------------------------------------------
--/H The view joining the two SQL logs
--/U
--/T
------------------------------------------------------------
AS     SELECT s.theTime, s.webserver, s.winname, s.clientIP, s.server, s.dbname,
              elapsed, busy, rows,
              sql, access, isVisible, procID, error, errorMessage
```

```
        FROM SqlStatementLogUTC s  with(nolock), SqlPerformanceLogUTC p  with(nolock)
        WHERE s.theTime = p.theTime
          and s.webserver=p.webserver
          and s.winName = p.winName
          and s.clientIP = p.clientIP

GO


--============================================================================
if exists (select * from dbo.sysobjects where name = N'SQLlogAll')
        drop view SQLlogAll
GO
CREATE VIEW SQLlogAll AS
        SELECT *
        FROM SqlLogUtc
GO
--============================================================================

--============================================================================
if exists (select * from dbo.sysobjects where name = N'SQLlog')
        drop view SQLlog
GO
CREATE VIEW SQLlog AS
        SELECT *
        FROM SqlLogUtc
        WHERE isVisible = 1
GO
--============================================================================

--============================================================================
if exists (select * from dbo.sysobjects where name = N'SQLStatementLog')
        drop view SQLStatementLog
GO
CREATE VIEW SQLStatementLog AS
        SELECT *
        FROM SqlStatementLogUtc
        WHERE isVisible = 1
GO
--============================================================================

--============================================================================
if exists (select * from dbo.sysobjects where name = N'SQLPerformanceLog')
        drop view SQLPerformanceLog
GO
CREATE VIEW SQLPerformanceLog AS
        SELECT *
        FROM SqlPerformanceLogUtc
GO
--============================================================================

--=====================================================================
IF EXISTS (SELECT name FROM   sysobjects
           WHERE  name = N'spLogSqlStatement' )
        DROP PROCEDURE spLogSqlStatement
GO
--
CREATE PROCEDURE spLogSqlStatement (
        @cmd VARCHAR(8000) OUTPUT,
         @webserver      VARCHAR(64) = '',   -- the url
        @winname       VARCHAR(64) = '',   -- the windows name of the server
         @clientIP       VARCHAR(16)  = 0,   -- client IP address
        @access        VARCHAR(64) = '',   -- subsite of site,  if 'collab' statement
'hidden'
        @startTime     datetime            -- time the query was started
)
-------------------------------------------------------------------------------
--/H Procedure to log a SQL query to the statement log.
-------------------------------------------------------------------------------
--/T Log the given query and its start time to the SQL statement log.  Note
--/T that we are logging only the start of the query yet, not a completed query.
```

```
--/T All the SQL statements are journaled into WebLog.dbo.SQLStatementlog.
--/T <samp>EXEC dbo.spLogSqlStatement('Select count(*) from
PhotoObj',getutcdate())</samp>
--/T See also spLogSqlPerformance.
------------------------------------------------------------------------------
AS
    BEGIN
        SET NOCOUNT ON
        DECLARE @error      INT;                    -- error number
        DECLARE @serverName varchar(32);            -- name of this databaes server
        DECLARE @dbName     VARCHAR(32);            -- name of this database
        SET     @serverName = @@servername;
        SELECT @dbname = [name] FROM master.dbo.sysdatabases WHERE dbid = db_id()
        DECLARE @isVisible  INT;                    -- flag says sql is visible to
internet queries
        SET @isVisible = 1;
        IF (UPPER(@access) LIKE '%COLLAB%') SET @isVisible = 0;  -- collab is invisible
        ------------------------------------------------------------------------
        --- log the command if there is a weblog DB
        if (0 != (select count(*) from master.dbo.sysdatabases where name = 'weblog'))
            begin
                insert WebLog.dbo.SqlStatementLogUTC
                values(@startTime,@webserver,@winName, @clientIP,
                      @serverName, @dbName, @access, @cmd, @isVisible)
            end
    END
GO


--====================================================================
IF EXISTS (SELECT name FROM   sysobjects
          WHERE  name = N'spLogSqlPerformance' )
       DROP PROCEDURE spLogSqlPerformance
GO
--
CREATE PROCEDURE spLogSqlPerformance (
        @webserver      VARCHAR(64) = '',   -- the url
        @winname        VARCHAR(64) = '',   -- the windows name of the server
        @clientIP       VARCHAR(16)  = 0,   -- client IP address
        @access         VARCHAR(64) = '',   -- subsite of site,  if 'collab' statement
'hidden'
        @startTime      datetime,           -- time the query was started
        @busyTime       bigint      = 0,    -- time the CPU was busy during query execution
        @endTime        datetime    = 0,    -- time the query finished
        @rows           bigint      = 0,    -- number of rows returned by the query
        @errorMsg       VARCHAR(1024) = ''  -- error message if applicable
)
------------------------------------------------------------------------------
--/H Procedure to log success (or failure) of a SQL query to the performance log.
------------------------------------------------------------------------------
--/T The caller needs to specify the time the query was started, the number of <br>
--/T seconds (bigint) that the CPU was busy during the query execution, the    <br>
--/T time the query ended, the number of rows the query returned, and an error <br>
--/T message if applicable.  The time fields can be 0 if there is an error.
--/T <samp>EXEC dbo.spLogSQLPerformance('skyserver.sdss.org','',,'',getutcdate())</samp>
--/T See also spLogSqlStatement.
------------------------------------------------------------------------------
AS
    BEGIN
        SET NOCOUNT ON
        ------------------------------------------------------
        -- record the performance when (if) the command completes.
        IF ( (@startTime IS NOT NULL) AND (@startTime != 0) AND
             (@busyTime != 0) AND (@endTime != 0) AND (LEN(@errorMsg) = 0) )
            BEGIN
                INSERT WebLog.dbo.SqlPerformanceLogUTC
                VALUES (@startTime,@webserver,@winName, @clientIP,
                        DATEDIFF(ms, @startTime, @endTime)/1000.0,      -- elapsed time
                        ((@@CPU_BUSY+@@IO_BUSY)-@busyTime)/1000.0,      -- busy time
                        @rows, @@PROCID, 0,'')                                      -
- rows returned
```

```
            END
        ELSE
            BEGIN
                IF ( (@startTime IS NULL) OR (@startTime = 0) )
                    SET @startTime = GETUTCDATE();
                INSERT WebLog.dbo.SqlPerformanceLogUTC
                VALUES (@startTime,@webserver,@winName, @clientIP,
                       0,0,0, @@PROCID, -1, @errorMsg)
            END
     END
GO


--================
-- user stuff
--================

if not exists (select * from dbo.sysusers
       where name = N'skyuser' and uid < 16382)
       EXEC sp_grantdbaccess N'skyuser', N'skyuser'
exec sp_addrolemember N'db_datareader', N'skyuser'
GO
if not exists (select * from dbo.sysusers
       where name = N'internet' and uid < 16382)
       EXEC sp_grantdbaccess N'internet', N'internet'
exec sp_addrolemember N'db_datareader', N'internet'
--
EXEC sp_adduser N'skyuser', N'SKYUSER'
GO
EXEC sp_change_users_login N'UPDATE_ONE', N'skyuser', N'skyuser'
GO
--

GRANT  SELECT, INSERT  ON SqlPerformanceLogUTC       TO weblog
GRANT  SELECT, INSERT  ON SqlStatementLogUTC         TO weblog
GRANT  SELECT                ON weblog                      TO weblog
GRANT  SELECT                ON SqlLog                      TO weblog
GRANT  SELECT                ON SqlLogAll                   TO weblog
GRANT  SELECT                ON SqlLogUTC                   TO weblog
GO
GRANT  SELECT, INSERT  ON SqlPerformanceLogUTC               TO skyuser
GRANT  SELECT, INSERT  ON SqlStatementLogUTC         TO skyuser
GRANT  SELECT          ON SqlPerformanceLog          TO skyuser
GRANT  SELECT          ON SqlStatementLog            TO skyuser
GRANT  SELECT                ON weblog                      TO skyuser
GRANT  SELECT                ON SqlLog                      TO skyuser
GO
GRANT  SELECT, INSERT  ON SqlPerformanceLogUTC               TO internet
GRANT  SELECT, INSERT  ON SqlStatementLogUTC         TO internet
GRANT  SELECT          ON SqlPerformanceLog          TO internet
GRANT  SELECT          ON SqlStatementLog            TO internet
GRANT  SELECT                ON weblog                      TO internet
GRANT  SELECT                ON SqlLog                      TO internet
GRANT  SELECT                ON SqlLogUTC                   TO internet
GO
```

Appendix B – sqlLoader/schema/log/webLogHarvesterCreate.sql

This is the source for all the scripts and stored procedures to create the harvester database for SDSS logs.

```
--================================================
-- weblogHarvesterCreate.sql
-- 2003-10-03 Jim Gray, Alex Szalay
-- 2005-09-07 Ani Thakar - renamed weblogDBCreate.sql
--                         to weblogHarvesterCreate.sql
------------------------------------------------
-- Modifications:
-- 2003-10-04 Alex: Added extra columns to weblog
-- 2003-10-04 Alex: Added extra columns to Sql***Log
-- 2003-10-26 Jim:  went from null to default
--                     change "null" to "not null default 0"
--                                   Weblog.error
--                                   weblog.isVisible
--                     change "null" to "not null default ''"       for
--                                   Weblog.second
--                                   Weblog.framework
--                                   Weblog.product
--                                   Weblog.op
--                                   Weblog.command
--                                   Weblog.browser
--                     Weblog.clientIP  not null default value 0.0.0.0
--                     let the command length be 7KB to catch odd things.
--                     permuted weblog columns (book-keeping last)
--                            added hh, mi, ss to weblog and dropped sec.
--                     Added WeblogAll base table (to replace weblog)
--                     Made WebLog a view.
-- 2003-10-26 Alex: Added LogSource and privacy table
-- 2003-10-26 Jim:  Added weblogTemp table
-- 2003-10-29 Jim:  changed weblog.op from char(6) to char(8)
--                  changed weblog.command from char(256) to char(7000)
--                  changed weblog.browser from char(256) to char(2000)
--                  added weblog.hh, mi, ss and dropped second
--                  added LogSource.weblog -- the name of the log file == name of web
server
--                  added TrafficBase.SkyServer
--                  added TrafficBase.SkyService
--                  changed WebLog to WebLogAll and added WebLog view
--                   added SqlLogAll and converted all sql logging to UTC
--                  added spCopySqlLog and spCopyWebLog
--                  fixed spUpdateTrafficBase to be simpler and to track SQL counters.
--                  appended populating logSource table.
--                  extensive rewrite of code to be driven by LogSource table.
-- 2004-04-01 Ani:  Updated schema of WebLogAll, WebLogTemp and SqlLogAll tables to bump
up
--                   seq to bigint (from int) and clientIP to varchar(256) (from
char(16)).
--
-- 2004-05-19 Ani:  Added creation of Weblog DB.
-- 2004-11-17 Ani:  Replaced bcpWeblog.js call with bcpWeblog.exe (C#
--                  version) in spCopyWeblogs.
-- 2004-12-02 Ani:  Made small change to DailyTraffic view so the month
--                  is zero-padded on the left and hence the descending
--                  order in the daily traffic display works correctly.
-- 2005-02-22 Ani:  Added totalIO to SqlLogAll table.
-- 2005-04-13 Ani:  Added new weblog columns to InsertFNALWeblog and
--                  cast the referer to varchar(500) pending a change
--                  to WebLogAll to expand it to bigger size.
-- 2005-10-11 Ani:  Added call to InsertFNALWeblog in spCopyWebLogs.
--                  Also added more logsources to InsertFNALWeblogs.
-- 2005-10-17 Ani:  Added Hungarian and Spanish branches.
-- 2006-01-23 Ani:  Added fields required by VO logging data model
--                  (referer, bytesOut, bytesIn, elapsed) to webLogAll
```

```
--                  and webLogTemp.
-- 2006-07-05 Ani:  Edited MonthlyTraffic view to use leading zero for
--                  the month so that order by desc is correct for
--                  2-digit months (10,11,12).
-- 2006-08-18 Ani:  Modified spCopyWeblogs and InsertFNALWeblogs to
--                  truncate command string if it contains password
--                  in clear.
-- 2008-09-15 Ani:  Modified spCopySqlLogs to test linked server connection before
harvesting
--                  a server, so it doesnt stop harvesting if there is a link error.
Also added
--                  new table ErrorLog to record errors encountered with linked servers
etc.
--=================================================
-- Create WebLog DB
CREATE DATABASE [weblog]  ON
  (NAME = N'weblog_Data1', FILENAME = N'C:\sql_db\weblog_Data1.MDF' , SIZE = 1000,
FILEGROWTH = 0%),
  (NAME = N'weblog_Data2', FILENAME = N'D:\sql_db\weblog_Data2.MDF' , SIZE = 20000,
FILEGROWTH = 10%)
  LOG ON (NAME = N'weblog_Log', FILENAME = N'D:\sql_db\weblog_Log.LDF' , SIZE = 10000,
FILEGROWTH = 10%)
  COLLATE SQL_Latin1_General_CP1_CI_AS
GO

exec sp_dboption N'weblog', N'autoclose', N'false'
GO
exec sp_dboption N'weblog', N'bulkcopy', N'false'
GO
exec sp_dboption N'weblog', N'trunc. log', N'true'
GO
exec sp_dboption N'weblog', N'torn page detection', N'true'
GO
exec sp_dboption N'weblog', N'read only', N'false'
GO
exec sp_dboption N'weblog', N'dbo use', N'false'
GO
exec sp_dboption N'weblog', N'single', N'false'
GO
exec sp_dboption N'weblog', N'autoshrink', N'false'
GO
exec sp_dboption N'weblog', N'ANSI null default', N'false'
GO
exec sp_dboption N'weblog', N'recursive triggers', N'false'
GO
exec sp_dboption N'weblog', N'ANSI nulls', N'false'
GO
exec sp_dboption N'weblog', N'concat null yields null', N'false'
GO
exec sp_dboption N'weblog', N'cursor close on commit', N'false'
GO
exec sp_dboption N'weblog', N'default to local cursor', N'false'
GO
exec sp_dboption N'weblog', N'quoted identifier', N'false'
GO
exec sp_dboption N'weblog', N'ANSI warnings', N'false'
GO
exec sp_dboption N'weblog', N'auto create statistics', N'true'
GO
exec sp_dboption N'weblog', N'auto update statistics', N'true'
GO
GO

------------------------
-- Create Web Log Tables
------------------------
USE WebLog

--=========================================================
if exists (select * from dbo.sysobjects where name =(N'LogSource'))
       drop table LogSource
```

```
GO
--
CREATE TABLE LogSource (
------------------------------------------------------------
--/H The basic information about sites to be harvested
--
--/T The entries here can correspond to websites and database servers
--/T The weblogs of active servers are copied to the WebLog database every hour
--/T and summary statistics are computed.
--/T The tstamp field gives the time of the most recent update.
------------------------------------------------------------
       logID          int           NOT NULL DEFAULT (0),   --/D unique ID of log used as
foreign key
       location       varchar(32)   NOT NULL DEFAULT (''),  --/D the location of the site
(FNAL, JHU,..
       service        varchar(32)   NOT NULL DEFAULT (''),  --/D type of service
(SKYSERVER, SKYSERVICE, SKYQUERY,...)
       instance       varchar(32)   NOT NULL default (''),  --/D The log underneath the
service (V1, V2,.. )
       uri            varchar(32)   NOT NULL default (''),  --/D The url or other ID for
this service.
       framework      varchar(32)   NOT NULL DEFAULT (''),  --/D the calling framework
(ASP,ASPX,HTML,QA,SOAP,...)
       product        varchar(32)   NOT NULL DEFAULT (''),  --/D the type of product
acessed (EDR, DR1, DR2,...
       method         varchar(16)   NOT NULL default(''),   --/D Harvesting method
(XCOPY|WGET|SQL)
       pathname       varchar(1024) NOT NULL default(''),   --/D The path of the log
LogSource (UNC|URL)
       tstamp         datetime      NOT NULL default Current_timestamp,    --/D The time
of the last harvesting
       isvisible      tinyint       NOT NULL default(1),    --/D should this log be
visible in the event log (0:no, 1:yes)
       status         varchar(32)   NOT NULL default(''),   --/D The current state
(ACTIVE|DISABLED)
       PRIMARY KEY(logID)
)
GO
ALTER TABLE LogSource ADD CONSTRAINT [ak_LogSource] UNIQUE
       (location, service, instance)
GO

------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'privacy')
       drop table privacy
GO
--
CREATE TABLE privacy (
------------------------------------------------------------
--/H The privacy setting of certain virtual directories
--
--/T The entries here can correspond to virtual directory names
------------------------------------------------------------
       location       varchar(32) NOT NULL default(''),     --/D The location of the
server
       website        varchar(32) NOT NULL default(''),     --/D The name of the website,
empty for database
       dirname        varchar(32) NOT NULL default(''),     --/D The name of the virtual
directory
       privacy        varchar(32) NOT NULL default(''),     --/D The privacy flag for the
LogSource (PUBLIC|COLLAB|...)
       PRIMARY KEY(location,website,dirname)
       )
GO


------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'weblogAll')
       drop table weblogAll
GO
--
```

```
CREATE TABLE weblogAll (
-------------------------------------------------------------
--/H The weblog information -- contains both visible and invisible log records.
--
--/T The information is parsed from the W3C format weblog
--/T files, generated on each web server by IIS. A record is
--/T considered to be invisible until its flag is set to 1.
-------------------------------------------------------------
       yy             smallint       NOT NULL,      --/D the year of the event
       mm             tinyint        NOT NULL,      --/D the month of the event
       dd             tinyint        NOT NULL,      --/D the day of the event
       hh              tinyint       NOT NULL,      --/D the hour of the event
       mi             tinyint        NOT NULL,      --/D the minute of the event
       ss             tinyint        NOT NULL,      --/D the second of the event
       seq            bigint         IDENTITY(1,1), --/D sequence number to uniquify the
event
       logID          int            NOT NULL DEFAULT(0) foreign key references
LogSource(logID),             --/D unique ID of log foreign key LogSource.logID
       clientIP       char(256)              NOT NULL DEFAULT ('0.0.0.0') ,        --/D
the IP address of the client
       op             char(8)        NOT NULL DEFAULT (''),        --/D the operation
(GET,POST,...)
       command        varchar(7000)  NOT NULL DEFAULT (''),        --/D the command
executed
       error          int            NOT NULL DEFAULT (0) ,        --/D the error code if
any
       browser        varchar(1024)  NOT NULL DEFAULT (''),               --/D the
browser type
       referer        varchar(1024)  NOT NULL DEFAULT (''),               --/D who
inboked the command
       bytesOut       bigint         NOT NULL DEFAULT (0),         --/D bytes returned by
request
       bytesIn        int            NOT NULL DEFAULT (0),         --/D bytes in request
       elapsed        int            NOT NULL DEFAULT (0),         --/D the time it took
to execute request --/U sec
       isVisible      tinyint        NOT NULL DEFAULT (0),         --/D should this event
be visible (0:no, 1:yes)
       PRIMARY KEY (yy desc ,mm desc,dd desc,hh desc,mi desc,ss  desc,seq desc,logID)
)

GO
ALTER TABLE  weblogAll NOCHECK CONSTRAINT ALL
--

-------------------------------------------------------
if exists (select * from dbo.sysobjects  where name =  N'weblogTemp')
       drop table weblogTemp
GO
-------------------------------------------------------
--/H Working table to hold newly arrived weblog data
--/T
-------------------------------------------------------------
CREATE TABLE weblogTemp (
       YY             int            NOT NULL,             --/D Year
       MM             int            NOT NULL,             --/D Month
       DD             int            NOT NULL,             --/D The location of the
server
       Seq            bigint         NOT NULL,             --/D sequence number in day's
weblog
       [Second]       char(8)        NOT NULL DEFAULT(''), --/D timestamp to second in
00:00:00 fomat
       clientIP       varchar(256)   NOT NULL DEFAULT('0.0.0.0'),--/D sequence number in
day's weblog
       op             char(8)        NOT NULL DEFAULT(''), --/D HTTP operation like GET
| POST |...
       command        varchar(7000)  NOT NULL DEFAULT('') , --/D HTTP command
       error          int            NOT NULL DEFAULT(0),  --/D HTTP error number
associated with this request
       browser        varchar(1024)  NOT NULL DEFAULT(''), --/D browser or program
making the request
```

```
       referer        varchar(1024)  NOT NULL DEFAULT (''),               --/D who
inboked the command
       bytesOut       bigint         NOT NULL DEFAULT (0),         --/D bytes returned by
request
       bytesIn        int            NOT NULL DEFAULT (0),         --/D bytes in request
       elapsed        int            NOT NULL DEFAULT (0),         --/D the time it took
to execute request --/U sec
       isVisible      tinyint        NOT NULL DEFAULT (0),         --/D should this event
be visible (0:no, 1:yes)
       CONSTRAINT pk_weblogtemp PRIMARY KEY  CLUSTERED( YY, MM, DD, Seq)
)
GO

--============================================================
if exists (select * from dbo.sysobjects where name = N'trafficBase')
       drop table trafficBase
GO
--
CREATE TABLE trafficBase (
-------------------------------------------------
--/H Contains an aggregate of the traffic in an hourly breakdown
--/U
--/T Currently the traffic is broken down into four branches.
--/T Later different granularities will also be considered.
-------------------------------------------------
       yy             int NOT NULL,                 --/D the year of the events
       mm             int NOT NULL,                 --/D the month of the events
       dd             int NOT NULL,                 --/D the day of the events
       hh             int NOT NULL,                 --/D the hour of the events
       hits           int NOT NULL DEFAULT(0),      --/D the total number of hits
       English        int NOT NULL DEFAULT(0),      --/D the no of hits on the English
branch
       German         int NOT NULL DEFAULT(0),      --/D the no of hits on the German
branch
       Hungarian      int NOT NULL DEFAULT(0),      --/D the no of hits on the Hungarian
branch
       Japanese       int NOT NULL DEFAULT(0),      --/D the no of hits on the Japanese
branch
       Spanish        int NOT NULL DEFAULT(0),      --/D the no of hits on the Spanish
branch
       Project        int NOT NULL DEFAULT(0),      --/D the no of hits on the Project
branch
       SkyServer      int NOT NULL DEFAULT(0),      --/D the no of hits on the SkyServer
       SkyService     int NOT NULL DEFAULT(0),      --/D the no of hits on the SkyService
       SQL            int NOT NULL DEFAULT(0)       --/D the no of SQL comands processed
)
GO
--
ALTER TABLE trafficBase WITH NOCHECK ADD CONSTRAINT [pk_trafficBase] PRIMARY KEY
CLUSTERED
       (yy,mm,dd,hh)
GO

--============================================================
-- Web Log Views
--============================================================
if exists (select * from dbo.sysobjects where name = N'weblog' )
        drop view weblog
GO

CREATE VIEW weblog  (
-------------------------------------------------------------
--/H The weblog information -- contains visible log records.
--
--/T The information is parsed from the W3C format weblog
--/T files, generated on each web server by IIS.
-------------------------------------------------------------
       yy             ,             --/D the year of the event
       mm             ,             --/D the month of the event
       dd             ,             --/D the day of the event
       hh              ,            --/D the hour of the event
```

```
       mi             ,             --/D the minute of the event
       ss             ,             --/D the second of the event
       logID          ,             --/D the log that this came from, foreign key:
LogSource.logID
       seq            ,             --/D sequence number
       clientIP       ,             --/D the IP address of the client
       op             ,             --/D the operation (GET,POST,...)
       command        ,             --/D the command executed
       error          ,             --/D the error code if any
       browser        ,             --/D the browser type
       location       ,             --/D the location of the site (FNAL, JHU,..
       service        ,             --/D type of service (SKYSERVER, SKYSERVICE,
SKYQUERY,...)
       instance       ,             --/D The log underneath the service (V1, V2,.. )
       uri            ,             --/D The url or other ID for this service.
       framework      ,             --/D the calling framework
(ASP,ASPX,HTML,QA,SOAP,...)
       product                      --/D the type of product acessed (EDR, DR1, DR2,...
) AS
SELECT  yy,mm,dd,hh,mi,ss, w.logID, seq, clientIP, op, command, error, browser,
       location, service, instance, uri, framework, product
       from WebLogAll w with(nolock) left outer join logSource ls on w.logID = ls.logID
       where w.isVisible = 1

GO

------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'DailyTraffic')
       drop view DailyTraffic
GO
--
CREATE VIEW DailyTraffic
AS
------------------------------------------------------------
--/H A view for the daily aggregate of the traffic
--/U
--/T
------------------------------------------------------------
    SELECT top 10000    (str(yy,4) + '/'
                + replace(str(mm,2),' ','0') +  '/'
--              + ltrim(str(mm,2)) +  '/'
                + ltrim(str(dd,2))) as date,
                sum(hits)      as hits,
                sum(English)   as English,
                sum(German)    as German,
                sum(Hungarian)         as Hungarian,
                sum(Japanese)  as Japanese,
                sum(Spanish)   as Spanish,
                sum(Project)   as Project,
                 sum(SkyServer)        as SkyServer,
                 sum(SkyService) as SkyService,
                 sum(SQL)      as SQL
    FROM  trafficBase with(nolock)
    WHERE hh is not null  -- supress any rollup rows
    GROUP BY yy,mm,dd
    ORDER BY yy desc,mm desc,dd desc
GO

------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'MonthlyTraffic')
       drop view MonthlyTraffic
GO
--
CREATE VIEW MonthlyTraffic
------------------------------------------------------------
--/H A monthly aggregation of the traffic
--/U
--/T
------------------------------------------------------------
AS
    SELECT TOP 30  (str(yy,4) + '/' + replace(str(mm,2),' ','0')) as month,
```

```
               sum(hits)      as hits,
               sum(English)   as English,
               sum(German)    as German,
               sum(Japanese)  as Japanese,
               sum(Project)   as Project,
                sum(SkyServer)        as SkyServer,
                sum(SkyService) as SkyService,
                sum(SQL)       as SQL
     FROM  trafficBase  with(nolock)
     WHERE hh is not null  -- suppress any rollup rows
     GROUP BY yy,mm
     ORDER BY yy DESC, mm DESC
GO

------------------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'TotalTraffic')
       drop view TotalTraffic
GO
--
CREATE VIEW TotalTraffic
-------------------------------------------------------------
--/H Yearly  view of the traffic
--/U
--/T
-------------------------------------------------------------
AS
       SELECT TOP 30  str(yy,4) as date,
              sum(hits)      as hits,
              sum(English)   as English,
              sum(German)    as German,
              sum(Japanese)  as Japanese,
              sum(Project)   as Project,
               sum(SkyServer)        as SkyServer,
               sum(SkyService) as SkyService,
               sum(SQL)       as SQL
       FROM  trafficBase  with(nolock)
       WHERE hh is not null  -- supress any rollup rows
       GROUP BY yy
       ORDER BY yy DESC
GO

-------------------------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'SQLlogAll')
       drop table SQLlogAll
GO
CREATE TABLE SQLlogAll (
-----------------------------------------------
--/H Contains an entry for each SQL statement that has been executed and for each CAS job
--/U
--/T Currently the traffic is broken down into four branches.
--/T Later different granularities will also be considered.
-----------------------------------------------
-----------------------------------------------
       yy             smallint       NOT NULL,      --/D the year of the event
       mm             tinyint        NOT NULL,      --/D the month of the event
       dd             tinyint        NOT NULL,      --/D the day of the event
       hh              tinyint       NOT NULL,      --/D the hour of the event
       mi             tinyint        NOT NULL,      --/D the minute of the event
       ss             tinyint        NOT NULL,      --/D the second of the event
       seq            bigint IDENTITY(1,1),         --/D a uniqueifier
       theTime        datetime       NOT NULL,      --/D the timestamp version yy-mm-dd
hh-mm-ss.ssss
       logID          int            NOT NULL DEFAULT (0),         --/D unique ID of log
foreign key LogSource.logID
       clientIP       varchar(256)    NOT NULL DEFAULT ('0.0.0.0') ,       --/D the IP
address of the client
       requestor      varchar(32)     NOT NULL DEFAULT ('') ,      --/D typically the web
server name
       server         varchar(32)    NOT NULL DEFAULT (''),        --/D the name of the
database server
```

```
	dbname		varchar(32)	NOT NULL DEFAULT (''),		--/D the name of the
database
	access		varchar(32)	NOT NULL DEFAULT (''),		--/D the kind of
access (collab, web, cas,...)
	elapsed		real		NOT NULL DEFAULT (0.0),		--/D the lapse
time of the query
	busy		real		NOT NULL DEFAULT (0.0),		--/D the total
CPU time of the query
	[rows]		bigint		NOT NULL DEFAULT (0),		--/D the number of
rows generated
	totalIO		bigint		NOT NULL DEFAULT (0),		--/D the number of IOs
(reads and writes) generated
	statement	varchar(8000)	NOT NULL DEFAULT (''),		--/D the command
executed
	error		int		NOT NULL DEFAULT (0) ,		--/D the error code if
any
	errorMessage	varchar(1000)	NOT NULL DEFAULT (''),		--/D the error
message if any
	isvisible	tinyint		NOT NULL DEFAULT (0),		--/D should this event
be visible (0:no, 1:yes)
	primary key (yy desc , mm desc, dd desc, hh desc , mi desc, ss desc, seq desc,
theTime, logID, clientIP,requestor, server)
)
GO
--============================================================================
if exists (select * from dbo.sysobjects where name = N'SQLlog')
	drop view SQLlog
GO
CREATE VIEW SQLlog AS
	SELECT *
	FROM SqlLogAll
	WHERE isVisible = 1
GO
--============================================================================

--============================================================
if exists (select * from dbo.sysobjects where name = N'SqlStatementLog')
	drop table SqlStatementLog
GO
--
CREATE TABLE SqlStatementLog (
------------------------------------------------
--/H The SQL statements submitted directly by end users (rather than website generated
ones)
--/U
--/T Records are inserted at the start of the query.
--/T At the end of the query, a corresponding SqlPerfomrnceLog record is generated.
------------------------------------------------
	theTime		datetime	NOT NULL DEFAULT (getdate()), --/D the timestamp
	procid		smallint	NOT NULL DEFAULT(0),		--/D the processid of
the query
	server		varchar(32)	NOT NULL DEFAULT(''),		--/D the name of the
database server
	dbname		varchar(32)	NOT NULL DEFAULT(''),		--/D the name of the
database
	sql		varchar(7800)	NOT NULL DEFAULT('')		--/D the SQL statement
)
GO
--
ALTER TABLE SqlStatementLog WITH NOCHECK ADD
	CONSTRAINT [pk_SqlStatementLog] PRIMARY KEY  CLUSTERED
	(theTime,server,dbname)
GO

--============================================================
if exists (select * from dbo.sysobjects where name = N'SqlPerformanceLog')
	drop table SqlPerformanceLog
GO
--
CREATE TABLE SqlPerformanceLog (
------------------------------------------------
```

```
--/H The cost of the SQL statements submitted directly by end users
--/U
--/T When a query compleltes the time, row count,and other attributes of the query are
added to the log.
--/T The corresponding SqlQueryLog tells what the statement is and where it came from.
-----------------------------------------------
       theTime datetime       NOT NULL DEFAULT (getdate()), --/D the timestamp
       elapsed real           NOT NULL DEFAULT (0.0),              --/D the lapse time of
the query
       busy    real           NOT NULL DEFAULT (0.0),              --/D the total CPU
time of the query
       procid  smallint       NOT NULL DEFAULT(0),          --/D the processid of the
query
       server  varchar(32)    NOT NULL DEFAULT(''),         --/D the name of the database
server
       dbname  varchar(32)    NOT NULL DEFAULT(''),         --/D the name of the database
       [rows]  bigint         NOT NULL DEFAULT (0)          --/D the number of rows
generated
)
GO
--
ALTER TABLE SqlPerformanceLog WITH NOCHECK ADD
       CONSTRAINT [pk_SqlPerformanceLog ] PRIMARY KEY CLUSTERED
       (theTime,server,dbname)
GO

--
-------------------------------------------------------------------------------
-- SQL log tables and views.
-------------------------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'SqlLog')
       drop view SqlLogLocal
GO

CREATE VIEW SqlLogLocal
-----------------------------------------------------------
--/H The view joining the two SQL logs
--/U
--/T
-----------------------------------------------------------
AS
       SELECT s.theTime,
           sql,
           elapsed,
           busy,
           rows
       FROM SqlStatementLog s  with(nolock), SqlPerformanceLog p  with(nolock)
       WHERE s.theTime = p.theTime
GO
-------------------------------------------------------------------------
-- new version of SQL Logs used by spExecute to record more info.
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-- SQL log tables and views.
-------------------------------------------------------------------------------

-----------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'SqlStatementLogUTC')
       drop table SqlStatementLogUTC
GO
--
CREATE TABLE SqlStatementLogUTC (
-----------------------------------------------
--/H The SQL statements submitted directly by end users (rather than website generated
ones)
--/U
--/T Records are inserted at the start of the query.
--/T At the end of the query, a corresponding SqlPerfomrnceLog record is generated.
-----------------------------------------------
       theTime        datetime       NOT NULL DEFAULT (getUTCdate()),--/D the timestamp
        webserver       varchar(64)   NOT NULL DEFAULT(''),         -- the url
```

```
        winname         varchar(64)    NOT NULL DEFAULT(''),         -- the windows name of
the server
         clientIP               varchar(16)    NOT NULL DEFAULT(''),         -- client IP
address
        server          varchar(32)    NOT NULL DEFAULT(''),         --/D the name of the
database server
        dbname          varchar(32)    NOT NULL DEFAULT(''),         --/D the name of the
database
        access          varchar(32)    NOT NULL DEFAULT(''),         --/D The website DR1,
collab,...
        sql             varchar(7800)  NOT NULL DEFAULT(''),         --/D the SQL statement
        isVisible       int            NOT NULL DEFAULT(1),          --/D flag says
statement visible on intenet
                                                                     --/D collab activity
is logged but not public
        PRIMARY KEY CLUSTERED (theTime,webserver,winname,clientIP)
)
GO

-------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'SqlPerformanceLogUTC')
        drop table SqlPerformanceLogUTC
GO
--
CREATE TABLE SqlPerformanceLogUTC (
-----------------------------------------------
--/H The cost of the SQL statements submitted directly by end users
--/U
--/T When a query compleltes the time, row count,and other attributes of the query are
added to the log.
--/T The corresponding SqlQueryLog tells what the statement is and where it came from.
-----------------------------------------------
        theTime         datetime       NOT NULL DEFAULT (getUTCdate()),      --/D the
timestamp
         webserver       varchar(64)   NOT NULL DEFAULT(''),         -- the url
        winname         varchar(64)    NOT NULL DEFAULT(''),         -- the windows name of
the server
         clientIP               varchar(16)    NOT NULL DEFAULT(''),         -- client IP
address
        elapsed         real           NOT NULL DEFAULT (0.0),               --/D the lapse
time of the query
        busy            real           NOT NULL DEFAULT (0.0),               --/D the total
CPU time of the query
        [rows]          bigint         NOT NULL DEFAULT (0),         --/D the number of
rows generated
        procid          int            NOT NULL DEFAULT(0),          --/D the processid of
the query
        error           int            NOT NULL DEFAULT(0),          --/D 0 if ok,
otherwise the sql error #, negative numbers are generated by the procedure
        errorMessage    varchar(2000)  NOT NULL DEFAULT(''),         --/D the error
message.
        PRIMARY KEY CLUSTERED (theTime,webserver,winname,clientIP)
)
GO


-------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'ErrorLog')
        drop table ErrorLog
GO
--
CREATE TABLE ErrorLog (
-----------------------------------------------------------
--/H Errors encountered by the harvester are logged here.
--/U
--/T
-----------------------------------------------------------
        currentTime datetime NOT NULL,
        context varchar(64) NOT NULL,
        errorMsg varchar(512) NOT NULL
)
```

```
GO
--


------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'SqlLogUTC')
       drop view SqlLogUTC
GO
--
CREATE VIEW SqlLogUTC
------------------------------------------------------------
--/H The view joining the two SQL logs
--/U
--/T
------------------------------------------------------------
AS     SELECT s.theTime, s.webserver, s.winname, s.clientIP, s.server, s.dbname,
              elapsed, busy, rows,
              sql, access, isVisible, procID, error, errorMessage
       FROM SqlStatementLogUTC s  with(nolock), SqlPerformanceLogUTC p  with(nolock)
       WHERE s.theTime = p.theTime
         and s.webserver=p.webserver
         and s.winName = p.winName
         and s.clientIP = p.clientIP

GO


----------------------
-- Procedures
----------------------

-------------------------------------------------------------------
--
-- imports recent weblogs, and updates statistics
-- first deletes weblog entries newer than the mindate.
-- also deletes corresponding elements in aggregate table (TrafficBase).
-- then imports weblogs since the min date (INCLUSIVE).
-- then recomputes aggregates.

-----------------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'spCopyWebLogs')
       drop procedure  spCopyWebLogs
GO
--
CREATE PROCEDURE spCopyWebLogs AS
------------------------------------------------------------
--/H Copies recent weblogs from web servers to the database
--/U
--/T Looks in the LogSource table for weblogs that are
--/T  ACTIVE and XCOPY
--/T For each such weblog, it
--/T   deletes entries from that log created since the minday of that log.
--/T   copies each such log to the weblog
--/T   advances the minday timestamp of that weblog.
------------------------------------------------------------
BEGIN
       SET NOCOUNT ON
       TRUNCATE TABLE  WebLogTemp                    -- clean out the scratch table
       DECLARE @date varchar(32),  @newDate datetime,       -- old date (when log was
current)
                                                     -- new date (bringing log up to date)
              @logID  int,                           -- the id of this log source
              @MinYY int, @minMM int, @minDD int,    -- yy mm dd verson of date
              @command varchar(256),                 -- command to do file copy
              @location VARCHAR(32),                 -- site that has log (e.g. JHU
              @service VARCHAR(32),                  -- service: SKYSERVER, SKYSERVICE,
SQL,...
              @instance VARCHAR(32),                        -- Which one V1, V2, ...
              @framework VARCHAR(32),                -- .NET, ASP+, SQL,...
              @product VARCHAR(32),                  -- EDR, DR1, DR2,
              @pathname VARCHAR(1000),               -- name to find log
```

```
               @tstamp DATETIME                      -- how current is the database (last
update time)
        ---------------------------------------
        -- Get dates
        SET @newDate = GETUTCDATE()
        SELECT  @tstamp = min(tstamp),
                @MinYY = datepart(yyyy,min(tstamp)),
                @MinMM = datepart(mm,  min(tstamp)),
                @MinDD = datepart(dd,  min(tstamp)),
                @date  = convert(varchar(10),min(tstamp),120) -- get yyyy-mm-dd
        FROM  LogSource
        WHERE  method = 'XCOPY' and isvisible = 1 and status = 'ACTIVE'
         DELETE weblogAll
                  WHERE  (  (yy > @MinYY) or
                            (yy = @MinYY and mm > @MinMM) or
                            (yy = @MinYY and mm = @MinMM and dd >= @MinDD)
                         ) and logID in (select logid FROM LogSource
                                         WHERE method = 'XCOPY'
                                           and isvisible = 1
                                           and status = 'ACTIVE'
                                        )
        ---------------------------------------------
        -- for each Active-Xcopy LogSource, copy its log across.
        DECLARE WebLogs CURSOR
          FOR   SELECT logID, location, service, instance, framework, product, pathname
                FROM LogSource
                WHERE method = 'XCOPY' and isvisible = 1 and status = 'ACTIVE'
          FOR   UPDATE of tstamp
        OPEN WebLogs
        FETCH NEXT FROM WebLogs INTO @logID, @location, @service, @instance, @framework,
@product, @pathname
        WHILE (@@fetch_status = 0)
                BEGIN
                ---------------------------------------------
                -- clean out the weblogs since the min time we have seen.
                -- PRINT 'LogSource ' + @location + ' ' + @service + ' '  + @instance + '
' + @date
--              set @command = 'cscript c:\WebLogImport\bcpWeblog.js ' + @pathname + ' ' +
@date
                set @command = 'c:\WebLogImport\bcpWeblog.exe ' + @pathname + ' ' +  @date
                print '*** doing bcp: ' + @command
                exec master..xp_cmdshell @command
                insert weblogAll
                        (yy, mm, dd, hh, mi, ss, logID, clientIP,  op, command, error,
browser, isVisible)
                   select  yy,mm,dd,
                        coalesce(cast(substring(second,1,2) as int),0) as hh,
                        coalesce(cast(substring(second,4,2) as int),0) as mi,
                        coalesce(cast(substring(second,7,2) as int),0) as ss,
                        @logID, clientIP, op,
                        -- if password is in clear in the command string, truncate it so
that
                        -- password is deleted
                        (case when command like '%password=%'
                            then substring(command, 1, patindex('%password=%', command)+8)
                            else command end) as command,
                        error, browser,
                        case when command like '/collab%' then 0 else 1 end  as isVisible
                        from WebLogTemp
                        order by yy desc, mm desc, dd desc, hh desc, mi desc, ss desc
                truncate table WebLogTemp

                --- mark that LogSource as current as of that time.
                UPDATE LogSource set tstamp = @newDate WHERE CURRENT OF WebLogs
                -- get next LogSource
                FETCH NEXT FROM WebLogs INTO @logID, @location, @service, @instance,
@framework, @product, @pathname
                END
        -- all resorces copied, close cursor and return.
        CLOSE WebLogs
        DEALLOCATE WebLogs
```

```
        SELECT  @tstamp = min(tstamp),
                @MinYY = datepart(yyyy,min(tstamp)),
                @MinMM = datepart(mm,  min(tstamp)),
                @MinDD = datepart(dd,  min(tstamp))
        FROM  LogSource
        WHERE  location='FNAL' and service='SKYSERVER' and isvisible = 1 and status =
'ACTIVE'
        EXEC InsertFNALWeblog @MinYY, @MinMM, @MinDD
        UPDATE LogSource set tstamp = @newDate
        WHERE  location='FNAL' and service='SKYSERVER' and isvisible = 1 and status =
'ACTIVE'
    --============================================================
    -- weblog copy is complete.
    --============================================================
    end
GO
-----------------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'spUpdateTrafficStats')
        drop procedure spUpdateTrafficStats
GO
--
CREATE PROCEDURE spUpdateTrafficStats AS
-------------------------------------------------------------
--/H Compute the recent log statistics (SQL and web)
--/U
--/T Looks in the LogSource table to find the last update time.
--/T deletes trafficBase entries created since the start of that day.
--/T computes the new values and inserts them in traffic base.
--/T then updetes the LogSource table to reflect that new time.
-------------------------------------------------------------
        BEGIN
        SET NOCOUNT ON
        -- update statiscs
        DECLARE @newDate DATETIME,                    -- the new time if update works
                @MinYY INT, @minMM INT, @minDD INT    -- the old time
        SET @newDate = GETUTCDATE()                   -- get the new time
        SELECT  @MinYY = datepart(yyyy,min(tstamp)),  -- get the old time
                @MinMM = datepart(mm,  min(tstamp)),
                @MinDD = datepart(dd,  min(tstamp))
        FROM  LogSource
        WHERE  service = 'TRAFFIC' and method = 'TSQL'

        ---------------------------------------------------------------
        -- truncating traffic base more recent that the min age
        DELETE trafficBase
         WHERE (yy >  @minYY)
            or (yy =  @minYY and mm > @minMM)
            or (yy =  @minYY and mm = @minMM and dd >= @minDD)

        ---------------------------------------------------------------
        -- updating the aggregate functions
        INSERT trafficBase (yy, mm, dd, hh, hits, English, German, Japanese,
Project,SkyServer, SkyService, SQL)
        SELECT yy, mm, dd, hh ,
                count(*) as hits,
                sum(case when command   like '%/en/%'     then 1 else 0 end),
                sum(case when command   like '%/de/%'     then 1 else 0 end),
                sum(case when command   like '%/jp/%'     then 1 else 0 end),
                sum(case when command   like '%/proj/%'   then 1 else 0 end),
                sum(case when service   =    'SKYSERVER'  then 1 else 0 end),
                sum(case when service   =    'SKYSERVICE' then 1 else 0 end),
                0                                    -- SQL
         FROM  weblogAll w join LogSource  l on w.logID=l.logID
         WHERE (yy > @minYY)
            or (yy = @minYY and mm > @minMM)
            or (yy = @minYY and mm = @minMM and dd >=  @minDD)
        GROUP BY yy, mm, dd, hh  --with rollup
        ORDER BY yy, mm, dd, hh  asc

        ---------------------------------------------------------------
        -- compute the SQL traffic
```

```
        update trafficbase
        set sql = s.SQL
        from trafficbase t join (
                                select yy,mm,dd,hh, count(*) as SQL
                                from SqlLogAll
                                where (yy > @minYY)
                                   or (yy = @minYY and mm > @minMM)
                                   or (yy = @minYY and mm = @minMM and dd >=  @minDD)
                                group by yy,mm,dd,hh) as s
              on t.yy=s.yy and t.mm=s.mm and t.dd=s.dd and t.hh=s.hh

        ------------------------------------------------------
        -- update the timestamp of the most recent traffic update
        UPDATE LogSource set tstamp = @newDate
         WHERE  service = 'TRAFFIC' and method = 'TSQL'
    ------------------------------------------------------------
    --completion message
    PRINT '*** Traffic Update completed succesfully'
end
GO
-----------------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'spCopySqlLogs')
        drop procedure  spCopySqlLogs
GO
--
CREATE PROCEDURE spCopySqlLogs AS
------------------------------------------------------------
--/H Copies recent SQL logs from SQL servers to the database
--/U
--/T Looks in the LogSource table for SQL logs that are
--/T  ACTIVE and accessible via the TSQL method
--/T For each such weblog, it
--/T   deletes entries from that log created since the min-day of that log.
--/T   copies sql log records since then to the central SQL log.
--/T   advances the minday timestamp of that sqllog in the logsource table.
------------------------------------------------------------
BEGIN
        SET NOCOUNT ON
        DECLARE @date varchar(32),                  -- old date (when log was current)
                @newDate datetime,                  -- new date (bringing log up to date)
                @logID  int,                        -- the id of this log source
                @MinYY int, @minMM int, @minDD int,   -- yy mm dd verson of date
                @command varchar(8000),                     -- command to do file copy
                @instance VARCHAR(32),                      -- Which one V1, V2, ...
                @pathname VARCHAR(1000),            -- name to find log
                @tstamp DATETIME                    -- how current is the database (last
update time)
        ---------------------------------------
        -- Get dates
        SET @newDate = GETUTCDATE()
        SELECT  @tstamp = min(tstamp),
                @MinYY = datepart(yyyy,min(tstamp)),
                @MinMM = datepart(mm,  min(tstamp)),
                @MinDD = datepart(dd,  min(tstamp)),
                @date  = convert(varchar(10),min(tstamp),120) -- get yyyy-mm-dd
        FROM   LogSource
        WHERE  instance != 'CasJobs' and
                ([service] = 'SQL' and method = 'TSQL' and isvisible = 1 and [status] =
'ACTIVE')

        ---------------------------------------------
        -- clean out the weblogs since the min time we have seen.
        DELETE SqlLogAll
           WHERE ( (yy > @MinYY) or
                   (yy = @MinYY and mm > @MinMM) or
                   (yy = @MinYY and mm = @MinMM and dd >= @MinDD) )
           AND logid IN
                  (SELECT logid
                   FROM LOGSOURCE
                   WHERE instance != 'CasJobs' and
```

```
              ([service] = 'SQL' and method = 'TSQL' and isvisible = 1 and [status] =
'ACTIVE'))

       ---------------------------------------------
       -- for each Active-TSQL LogSource, copy its log across.
       DECLARE SqlLogs CURSOR
         FOR   SELECT logID, instance, pathname
               FROM  LogSource
               WHERE instance != 'CasJobs' and
               ([service] = 'SQL' and method = 'TSQL' and isvisible = 1 and [status] =
'ACTIVE')
         FOR   UPDATE of tstamp
       OPEN SqlLogs
       FETCH NEXT FROM SqlLogs INTO @logID, @instance, @pathname
       WHILE (@@fetch_status = 0)
           BEGIN
               -- first check if linked server connection works, if not skip this server
               declare @ret int, @srv nvarchar(128), @len int, @end int, @msg
varchar(128)
               set @end = charindex(']',@pathname,1);
               if (@end > 2)
                   begin
                       set @len = @end - 2;
                       set @srv = substring(@pathname,2,@len);
                       begin try
                           exec @ret = sys.sp_testlinkedserver @srv
                       end try
                       begin catch
                           set @ret=sign(@@error);
                           set @msg = 'Failed to connect to server ' + @srv;
                           INSERT ErrorLog VALUES( getdate(), 'spCopySqlLogs', @msg );
                       end catch
                   end
               else
                   set @ret = 0;
               IF (@ret = 0)          -- if link test succeeded
                   BEGIN
                       set @command =
                          ' Insert SqlLogAll '
                       +  ' Select '
                       +  ' DATEPART (yyyy , theTime ) as yy,'
                       +  ' DATEPART (  mm , theTime ) as mm,'
                       +  ' DATEPART (  dd , theTime ) as dd,'
                       +  ' DATEPART (  hh , theTime ) as hh,'
                       +  ' DATEPART (  mi , theTime ) as mi,'
                       +  ' DATEPART (  ss , theTime ) as ss,'
                       +  ' theTime, '
                       +  cast(@logID as varchar(10)) + ' as logid,'
                       +  ' clientIP, '
                       +  ' webserver, '
                       +  ' server,'
                       +  ' dbname,'
                       +  ' access,'
                       +  ' coalesce(elapsed, 99999999) as elapsed,'
                       +  ' coalesce(busy, 99999999) as busy,'
                       +  ' coalesce(rows, 99999999) as rows,'
                       +  ' substring(sql,1,7950) as statement, '
                       +  ' coalesce(error, -2) as error, '
                       +  ' coalesce(errorMessage,''timeout'') as errorMesssage,'
                       +  ' isVisible'
                       +  ' from ' + @pathname + '.dbo.sqlLogUtc '
                       +  ' where theTime > ''' + substring(cast(@tStamp as
varchar(35)),1,11) + ' 00:00:00'''
                       +  ' order by theTime desc ' ;
                       --print @command
                       exec (@command)
                       --- mark that LogSource as current as of that time.
                       UPDATE LogSource set tstamp = @newDate WHERE CURRENT OF SqlLogs
                   END
               -- get next LogSource
               FETCH NEXT FROM SqlLogs INTO @logID, @instance, @pathname
```

```
            END
        -- all SQL Logs copied, close cursor and return.
        CLOSE SqlLogs
        DEALLOCATE SqlLogs
    --============================================================
    -- weblog copy is complete.
    --============================================================
    end
GO


-----------------------------------------------------------------------
if exists (select * from dbo.sysobjects where name = N'InsertFNALWeblog')
       drop procedure  InsertFNALWeblog
GO
--
CREATE  PROCEDURE InsertFNALWeblog( @minYY int, @minMM int, @minDD int ) AS
------------------------------------------------------------
--/H Copies recent weblogs from FNAL public web server to the database
--/U
--/T For now, copies logs directly from server at FNAL.
--/T DR3 hits are marked as logID=8 for convenience, DR2 logID=7.
------------------------------------------------------------
BEGIN
       DELETE weblogAll
                 WHERE  (  (yy > @minYY) or
                           (yy = @minYY and mm > @minMM) or
                           (yy = @minYY and mm = @minMM and dd >= @minDD)
                        ) and logID IN (7,8,9,10,11,12,13)
       INSERT weblogAll
       (yy, mm, dd, hh, mi, ss, logID, clientIP,  op, command, error, browser, isVisible,
referer, bytesOut, bytesIn, elapsed)
              select  yy,mm,dd,
                      coalesce(cast(substring(second,1,2) as int),0) as hh,
                      coalesce(cast(substring(second,4,2) as int),0) as mi,
                      coalesce(cast(substring(second,7,2) as int),0) as ss,
                      case when command like '%dr3%' then 8 else 7 end as logID,
                      clientIP, op,
                      -- if password is in clear in the command string, truncate it so
that
                      -- password is deleted
                      (case when command like '%password=%'
                          then substring(command, 1, patindex('%password=%', command)+8)
                          else command end) as command,
                      error, browser,
                      case when command like '/collab%' then 0 else 1 end  as isVisible,
                      cast(referer as varchar(500)), bytesOut, bytesIn, elapsed
                      from [SDSSSQL004.FNAL.GOV].Weblog.dbo.WebLog
                      where ( (yy > @minYY) OR
                              (yy = @minYY AND mm > @minMM) OR
                              (yy = @minYY AND mm = @minMM AND dd >= @minDD)
                            )
                      order by yy desc ,mm desc,dd desc,hh desc,mi desc,ss  desc
END
GO

-----------------
-- user stuff
-----------------
exec sp_addrole N'weblog'

if not exists (select * from dbo.sysusers
       where name = N'weblog' and uid < 16382)
       EXEC sp_grantdbaccess N'weblog', N'weblog'
exec sp_addrolemember N'db_datareader', N'weblog'
GO
if not exists (select * from dbo.sysusers
       where name = N'test' and uid < 16382)
       EXEC sp_grantdbaccess N'test', N'test'
exec sp_addrolemember N'db_datareader', N'test'
GO
```

```
if not exists (select * from dbo.sysusers
       where name = N'internet' and uid < 16382)
       EXEC sp_grantdbaccess N'internet', N'internet'
exec sp_addrolemember N'db_datareader', N'internet'
--
exec sp_addrolemember N'db_datareader', N'weblog'
GO
--
GRANT  SELECT, INSERT  ON SqlPerformanceLog  TO weblog
GRANT  SELECT, INSERT  ON SqlStatementLog    TO weblog
GRANT  SELECT                ON trafficBase               TO weblog
GRANT  SELECT                ON weblog             TO weblog
GRANT  SELECT                ON DailyTraffic       TO weblog
GRANT  SELECT                ON MonthlyTraffic     TO weblog
GRANT  SELECT                ON SqlLog             TO weblog
GRANT  SELECT                ON TotalTraffic       TO weblog
go
GRANT  SELECT                ON trafficBase               TO test
GRANT  SELECT                ON weblog             TO test
GRANT  SELECT                ON DailyTraffic       TO test
GRANT  SELECT                ON MonthlyTraffic     TO test
GRANT  SELECT                ON SqlLog             TO test
GRANT  SELECT                ON TotalTraffic       TO test
GO
GRANT  SELECT                ON trafficBase               TO internet
GRANT  SELECT                ON weblog             TO internet
GRANT  SELECT                ON DailyTraffic       TO internet
GRANT  SELECT                ON MonthlyTraffic     TO internet
GRANT  SELECT                ON SqlLog             TO internet
GRANT  SELECT                ON TotalTraffic       TO internet
GO
--================================================================================
-- PopulateLogSource.sql
-- 2003-10-26 Alex Szalay
-- Load the values of the LogSource and privacy tables
-- so that we know where to harvest from
-- 2003-10-29 jim: added weblog values
--================================================================================
SET NOCOUNT ON
delete LogSource
GO
-----------------<SKYSERVER weblogs>-------------------
-- skyserver.fnal.gov
INSERT LogSource VALUES(1, 'FNAL', 'SKYSERVER', 'EDR','skyserver.fnal.gov',    'ASP',
'EDR', 'WGET',
       '???? ', '2001-01-01', 1, 'ACTIVE');
-- skyserver.pha.jhu.edu
INSERT LogSource VALUES(2, 'JHU', 'SKYSERVER', 'EDR', 'skyserver.pha.jhu.edu', 'ASP',
'EDR', 'XCOPY',
       '\\skyserver\LogFiles\W3SVC1\', '2001-01-01', 1, 'INACTIVE');
-- skyserver.pha.jhu.edu
INSERT LogSource VALUES(3, 'JHU', 'SKYSERVER', 'V1', 'skyserver.pha.jhu.edu', 'ASP',
'EDR', 'XCOPY',
       '\\skyserver\LogFiles\W3SVC1\', '2001-01-01', 1, 'ACTIVE');
-- www.skyserver.org
INSERT LogSource VALUES(4, 'JHU', 'SKYSERVER', 'V3', 'www.skyserver.org',     'ASP+',
'DR1', 'XCOPY',
       '\\skyserver\LogFiles\W3SVC3\', '2001-01-01', 1, 'ACTIVE');
-- skyserver.sdss.org
INSERT LogSource VALUES(5, 'JHU', 'SKYSERVER', 'V4', 'skyserver.sdss.org',     'ASP+',
'DR1', 'XCOPY',
       '\\skyserver\LogFiles\W3SVC4\', '2001-01-01', 1, 'ACTIVE');
--  skyserver.sdss.org
INSERT LogSource VALUES(6, 'JHU', 'SKYSERVER', 'V5', 'skyserver.sdss.org',    'ASP+',
'DR1', 'XCOPY',
       '\\skyserver\LogFiles\W3SVC5\', '2001-01-01', 1, 'ACTIVE');
INSERT LogSource VALUES(7, 'FNAL', 'SKYSERVER', 'DR2', 'cas.sdss.org',        'ASP+',
'DR2', 'WGET',
       '????', '2001-01-01', 1, 'ACTIVE');
INSERT LogSource VALUES(8, 'FNAL', 'SKYSERVER', 'DR3', 'cas.sdss.org',        'ASP+',
'DR3', 'WGET',
```

```
       '????', '2001-01-01', 1, 'ACTIVE');

--=================<SKYSERVICE weblogs>==================
-- this is the skyservice.pha.jhu.edu
INSERT LogSource VALUES(1001, 'JHU', 'SKYSERVICE', 'V1', 'skyservice.pha.jhu.edu',
'.NET', 'DR1', 'XCOPY',
       '\\skyservice\LogFiles\W3SVC1\', '2001-01-01', 1, 'ACTIVE');
-- voservices.org
INSERT LogSource VALUES(1002, 'JHU', 'SKYSERVICE', 'V2', 'voservices.org',
'.NET', 'DR1', 'XCOPY',
       '\\skyservice\LogFiles\W3SVC2\', '2001-01-01', 1, 'ACTIVE');
-- voservices.net
INSERT LogSource VALUES(1003, 'JHU','SKYSERVICE',  'V3', 'voservices.net',
'.NET', 'DR1', 'XCOPY',
       '\\skyservice\LogFiles\W3SVC3\', '2001-01-01', 1, 'ACTIVE');
-- skyquery.net
INSERT LogSource VALUES(1004, 'JHU','SKYSERVICE', 'V4', 'skyquery.net',
'.NET', 'DR1', 'XCOPY',
       '\\skyservice\LogFiles\W3SVC4\', '2001-01-01', 1, 'ACTIVE');
-- skyquery.org
INSERT LogSource VALUES(1005, 'JHU','SKYSERVICE', 'V5', 'skyquery.org',
'.NET', 'DR1', 'XCOPY',
       '\\skyservice\LogFiles\W3SVC5\', '2001-01-01', 1, 'ACTIVE');
-- photo-z.net
INSERT LogSource VALUES(1006, 'JHU','SKYSERVICE', 'V7', 'photo-z.net',
'.NET', 'DR1', 'XCOPY',
       '\\skyservice\LogFiles\W3SVC7\', '2001-01-01', 1, 'ACTIVE');

--=================<SQL LOG Update>==================
INSERT LogSource VALUES(2001, 'FNAL','SQL',     'EDR',  'skyserver.fnal.gov/weblog',
'SQL',  'EDR',     'WSQL',
       'FNAL-EDR.WebLog.', '2001-01-01', 1, 'ACTIVE');
INSERT LogSource VALUES(2002, 'JHU','SQL',     'EDR',  'skyserver.pha.jhu.edu/weblog',
'SQL',  'EDR',     'WSQL',
       'FNAL-EDR.WebLog.', '2001-01-01', 1, 'INACTIVE');
-- SDSSDR1
INSERT LogSource VALUES(2003, 'JHU', 'SQL',     'SDSSDR1', 'SDSSDR1.weblog',       'SQL',
'DR1', 'TSQL',
       'SdssDr1.weblog', '2001-01-01', 1, 'ACTIVE');
-- SDSSAD2
INSERT LogSource VALUES(2004, 'JHU', 'SQL',     'SDSSAD2', 'SDSSAD2.weblog',       'SQL',
'DR1', 'TSQL',
       'SdssAD2.weblog', '2001-01-01', 1, 'ACTIVE');
-- SDSSAD3 -- NONE
-- SDSSAD4 -- NONE
-- SDSSAD5 -- NONE
-- SDSSAD6
INSERT LogSource VALUES(2005, 'JHU', 'SQL',     'SDSSAD6', 'SDSSAD6.weblog',       'SQL',
'DR1', 'TSQL',
       'SdssAD6.weblog', '2001-01-01', 1, 'ACTIVE');
-- LIBERTY -- NONE

--=================<TrafficUpdate>==================
INSERT LogSource VALUES(9001, 'JHU', 'TRAFFIC', 'V1',   'skyserver.sdss.org',
'ASP+', 'ALL', 'TSQL',
       'SdssAd2.weblog.dbo', '2001-01-01', 1, 'ACTIVE');
GO

------------------------------------------------------------------------
DELETE privacy
GO
INSERT privacy VALUES('JHU','skyserver.sdss.org',   'collab',  'COLLAB');
INSERT privacy VALUES('JHU','skyserver.pha.jhu.edu','collab',  'COLLAB');
INSERT privacy VALUES('JHU','skyserver.sdss.org',   'collabpw','COLLAB');
INSERT privacy VALUES('JHU','skyserver.pha.jhu.edu','collabpw','COLLAB');
GO
```

Appendix C: sqlLoader/schema/sql/spSQLSupport.sql

This is the source for the spExecuteSQL stored procedure that handles each query submitted by the user and logs the query beginning and end to the local Weblog DB.

```
--------------------------------------------------------------------------
--   spSQLSupport.sql
--   2001-11-01 Alex Szalay
--------------------------------------------------------------------------
-- History:
--* 2001-12-02 Jim:  added comments, fixed things
--* 2001-12-24 Jim: Changed parsing in SQL stored procedures
--* 2002-05-10 Ani: Added "limit" parameter to spExecuteSQL,
--*            spSkyServerFormattedQuery, spSkyServerFreeFormQuery so
--*            that 1000-record limit on output can be turned off.
--* 2002-07-04 Jim: Added sql command loging to spExecSQL (statements go to weblog).
--* 2003-01-20 Ani: Added Tanu's changes to spSkyServerTables,
--*            spSkyServerDatabases, spSkyServerFunctions and spSkyServerFreeFormQuery
--* 2003-02-05 Alex: fixed URL construction in support functions
--* 2003-02-06 Ani: Removed "en/" from URLs so it will be compatible with all
--*                 subdirectories (e.g. v4).
--*                 Updated spSkyServerFunctions to include SPs.
--*                 Changed "imagingRoot" to "imaging" for image FITS URLs
--*                 Changed "spectroRoot" to "spectro" for spSpec URLs
--*                 Changed "1d_10" to "1d_20" for spSpec URLs
--* 2003-03-10 Ani: Updated spSkyServerColumns to be same as fDocColumns.
--* 2003-11-11 Jim: Added clientIP, access, webserver... to spExecuteSQL
--*            revectored spSkyServerFreeFormQuery to call spExecuteSQL
--* 2004-08-31 Alex: changed replacei to fReplace everywhere to conform to naming
--* 2004-09-01 Nolan+Alex: added spExecuteSQL2
--* 2005-02-15 Ani: Added check for crawlers limiting them to x queries/min,
--*            SQL code for this provided by Jim.
--* 2005-02-18 Ani: Commented out "SET @IOs = ..." line in spExecuteSQL and
--*            spExecuteSQL2 because it was cause arithmetic overflow errors
--*            on DR3 sites (PR #6367).
--* 2005-02-21 Ani: Applied permanent fix for PR #6367 by including casts to
--*            bigint for @@TOTAL_READ and @@TOTAL_WRITE.
--* 2005-02-25 Ani: Added minute-long window to check whether a given
--*            clientIP is submitting more than the max number of queries
--*            in spExecuteSQL.  This is to allow legitimate rapid-fire
--*            queries like RHL's skyserver lookup tables to work fine.
--* 2006-02-07 Ani: Added syntax check capability in spExecuteSql, and also
--*            replaced cr/lf with space/lf so line number can be displayed
--*            for syntax errors (see PR #6880).
--* 2006-03-10 Ani: Added spLogSqlStatement and spLogSqlPerformance for
--*            CasJobs to use.
--* 2007-01-01 Alex: separated out spSQLSupport
--* 2007-08-27 Ani: Added "system" parameter to spExecuteSQL to allow it
--*            to distinuguish queries submitted by CAS tools from user
--*            queries.  Without this, system queries often run into
--*            the max #queries/min limit.
--* 2008-04-21 Ani: Added "maxQueries" parameter to spExecuteSQL so that
--*            the client can pass the throttle value to it.
--------------------------------------------------------------------------
SET NOCOUNT ON;
GO

--------------------------------------------------------------------
IF EXISTS (SELECT name FROM   sysobjects
          WHERE  name = N'fReplace' )
       DROP FUNCTION fReplace
GO
--
CREATE FUNCTION fReplace(@oldstr VARCHAR(8000), @pattern VARCHAR(1000), @replacement
VARCHAR(1000))
-----------------------------------------------
--/H Case-insensitve string replacement
```

```
-----------------------------------------------
--/T Used by the SQL parser stored procedures.
-----------------------------------------------
RETURNS varchar(8000)
AS
BEGIN
        -----------------------------------
        DECLARE @newstr varchar(8000);
        SET @newstr = '';
        IF (LTRIM(@pattern) = '') GOTO done;
        -----------------------------------
        DECLARE @offset int,
                       @patlen int,
                       @lowold varchar(8000),
                       @lowpat varchar(8000);
        SET @lowold = LOWER(@oldstr);
        SET @lowpat = LOWER(@pattern);
        SET @patlen = LEN(@pattern);
        SET @offset = 0
        --
        WHILE (CHARINDEX(@lowpat,@lowold, 1) != 0 )
        BEGIN
               SET @offset    = CHARINDEX(@lowpat, @lowold, 1);
               SET @newstr    = @newstr + SUBSTRING(@oldstr,1,@offset-1) + @replacement;
               SET @oldstr    = SUBSTRING(@oldstr, @offset+@patlen, LEN(@oldstr)-
@offset+@patlen);
               SET @lowold    = SUBSTRING(@lowold, @offset+@patlen, LEN(@lowold)-
@offset+@patlen);
        END
        ---------------------------------------
done:   RETURN( @newstr + @oldstr);
END
GO



--======================================================================
IF EXISTS (SELECT name FROM   sysobjects
           WHERE  name = N'fIsNumbers' )
        DROP FUNCTION fIsNumbers
GO
--
CREATE FUNCTION fIsNumbers (@string varchar(8000), @start int, @stop int)
-----------------------------------------------------------------
--/H Check that the substring is a valid number.
--
--/T <br>fIsNumbers(string, start, stop) Returns
--/T <LI>  -1: REAL (contains decimal point) ([+|-]digits.digits)
--/T <LI>   0: not a number
--/T <LI>   1: BIGINT    ([+|-] 19 digits)
--/T <br>
--/T <samp>  select dbo.fIsNumbers('123;',1,3);
--/T <br>  select dbo.fIsNumbers('10.11;'1,5);</samp>
-----------------------------------------------------------------
RETURNS INT
AS BEGIN
        DECLARE @offset int,          -- current offfset in string
               @char   char,          -- current char in string
               @dot    int,           -- flag says we saw a dot.
               @num    int;           -- flag says we saw a digit
        SET @dot = 0;                 --
        SET @num = 0;                 --
        SET @offset = @start;         --
        IF (@stop > len(@string)) RETURN 0;  -- stop if past end
        SET @char = substring(@string,@offset,1); -- handle sign
        IF(@char ='+' or @char='-') SET @offset = @offset + 1;
        ------------------
        -- process number
        ------------------
        WHILE (@offset <= @stop)                    -- loop over digits
```

```
        BEGIN                                                   -- get next char
                SET @char = substring(@string,@offset,1);
                IF (@char = '.')                                -- if a decimal point
                BEGIN                                                   -- reject duplicate
                        IF (@dot = 1) RETURN 0;
                        SET @dot = 1;  -- set flag
                        SET @offset = @offset + 1;  -- advance
                END             -- end dot case
                ELSE IF (@char<'0' or '9' <@char)  -- if not digit
                        RETURN 0;                                       -- reject
                ELSE                                                    -- it's a digit
                BEGIN                                                   -- advance
                        SET @offset = @offset + 1;
                        SET @num= 1;                            -- set digit flag
                END                                                     -- end digit case
        END                                                             -- end loop
        ----------------------------
        -- test for bigint overflow
        ----------------------------
        IF (@stop-@start > 19) RETURN 0;      -- reject giant numbers
        IF  (@dot = 0 and  @stop-@start >= 19 )
        BEGIN                                                           -- if its a bigint
                IF ( ((@stop-@start)>19) or           -- reject if too big
                        ('9223372036854775807' > substring(@string,@start,@stop)))
                   RETURN 0;                                    --
        END                                                             -- end bigint overflow
test
        IF (@num = 0) RETURN 0;                                 -- complain if no digits
        IF (@dot = 0) RETURN 1;                         -- number ok, is it an int
        RETURN -1 ;                                                     -- or a float?
END
GO


------------------------------------------------------------------------
IF EXISTS (SELECT name FROM   sysobjects
           WHERE  name = N'spExecuteSQL' )
        DROP PROCEDURE spExecuteSQL
GO
--
CREATE PROCEDURE spExecuteSQL (@cmd VARCHAR(8000), @limit INT = 1000,
        @webserver      VARCHAR(64) = '',   -- the url
        @winname        VARCHAR(64) = '',   -- the windows name of the server
        @clientIP       VARCHAR(16)  = 0,   -- client IP address
        @access         VARCHAR(64) = '',    -- subsite of site,  if 'collab' statement
'hidden'
        @system         TINYINT = 0,         -- 1 if this is a system query from a
skyserver page
        @maxQueries     SMALLINT = 60          -- maximum number of queries per minute
)
-------------------------------------------------------------------------------
--/H Procedure to safely execute an SQL select statement
-------------------------------------------------------------------------------
--/T The procedure casts the string to lowercase (this could affect some search
statements)
--/T It rejects strings continuing semicolons;
--/T It then discards duplicate blanks, xp_, sp_, fn_, and ms_ substrings.
--/T we are guarding aginst things like "select dbo.xp_cmdshell('format c');"
--/T Then, if the "limit" parameter is > 0 (true), we insist that the
--/T   statement have a top x in it for x < 1000, or we add a TOP 1000 clause.
--/T Once the SELECT statement is transformed, it is executed
--/T and returns the answer set or an error message.   <br>
--/T All the SQL statements are journaled into WebLog.dbo.SQLlog.
--/T <samp>EXEC dbo.spExecuteSQL('Select count(*) from PhotoObj')</samp>
-------------------------------------------------------------------------------
AS
        BEGIN
        SET NOCOUNT ON
        DECLARE @inputCmd varchar(8000)                 -- safe copy of command for log
        SET    @inputCmd = @cmd
        --      SET @cmd = LOWER(@cmd)+ ' ';            -- makes parsing easier
```

```
        SET    @cmd = @cmd + ' ';
        DECLARE @oldCmd     VARCHAR (8000);            -- temporary copy of command
        DECLARE @error      INT;                       -- error number
        DECLARE @errorMsg VARCHAR(100), @ipAddr VARCHAR(100);              -- error msg
        DECLARE @serverName varchar(32);               -- name of this databaes server
        DECLARE @dbName     VARCHAR(32);               -- name of this database
        SET    @serverName = @@servername;
        SELECT @dbname = [name] FROM master.dbo.sysdatabases WHERE dbid = db_id()
        DECLARE @i          INT;                       -- token scan offset
        DECLARE @isVisible  INT;                       -- flag says sql is visible to
internet queries
        SET @isVisible = 1;
        IF (UPPER(@access) LIKE '%COLLAB%') SET @isVisible = 0;  -- collab is invisible

        IF (@system = 0)       -- if not a system (internal) query from skyserver
            BEGIN
                -- Restrict users to a certain number of queries per minute to
                -- prevent crawlers from hogging the system.
                DECLARE @ret INT, @nQueries INT
                SET @maxQueries = 60   -- max queries per minute limit.

                -- first delete elements that are older than the window sampled.
                -- RecentRequests will typically have 4 * @maxQueries at peak
                -- times (at a peak rate of 4 queries/second).
                DELETE RecentQueries
                    WHERE lastQueryTime < DATEADD(ss,-60,CURRENT_TIMESTAMP)

                -- now check how many queries this IP submitted within the last minute.
                -- if more than @maxQueries, reject the query with an error message
                -- if not, insert IP into recent requests log and run query
                SELECT @nQueries=count(*) FROM RecentQueries WHERE ipAddr=@clientIP
                IF (@nQueries > @maxQueries)
                    BEGIN
                        SET @errorMsg = 'ERROR: Maximum ' + cast(@maxQueries as
varchar(3))
                              + ' queries allowed per minute. Rejected query: ';
                        GOTO bottom;
                    END
                ELSE
                    INSERT RecentQueries VALUES (@clientIP, CURRENT_TIMESTAMP)
            END  -- IF (@system = 0)    -- not a system query

        DECLARE @top varchar(20);
        SET @top = ' top '+cast(@limit as varchar(20))+' ';
        ------------------------------------------------------------------------
        -- Remove potentially dangerous expressions from the string.
        UNTIL:  BEGIN
                SET @oldCmd = @cmd;
                SET @cmd = dbo.fReplace(@cmd, '.xp_',   '#'); -- discard extended SPs
                SET @cmd = dbo.fReplace(@cmd, '.sp_',   '#'); -- discard stored
procedures
                SET @cmd = dbo.fReplace(@cmd, '.fn_',   '#'); -- discard functions
                SET @cmd = dbo.fReplace(@cmd, '.ms_',   '#'); -- discard microsoft
extensions
                SET @cmd = dbo.fReplace(@cmd, '.dt_',   '#'); -- discard microsoft
extensions
                SET @cmd = dbo.fReplace(@cmd, ' xp_',   '#'); -- discard extended SPs
                SET @cmd = dbo.fReplace(@cmd, ' sp_',   '#'); -- discard stored
procedures
                SET @cmd = dbo.fReplace(@cmd, ' fn_',   '#'); -- discard functions
                SET @cmd = dbo.fReplace(@cmd, ' ms_',   '#'); -- discard microsoft
extensions
                SET @cmd = dbo.fReplace(@cmd, ' dt_',   '#'); -- discard microsoft
extensions
                SET @cmd =      replace(@cmd, '  ' ,    ' '); -- discard duplicate spaces
                SET @cmd =      replace(@cmd, '  ' ,    ' '); -- discard duplicate spaces
                SET @cmd=       replace(@cmd,0x0D0A,   0x200A);     -- replace cr/lf with
space/lf
                SET @cmd=       replace(@cmd,  0x09,    ' ');     -- discard tabs
                SET @cmd =      replace(@cmd,   ';',    '#'); -- discard semicolon
                END
```

```
        IF (@cmd != @oldCmd) GOTO UNTIL;
        --------------------------------------------------------------------------------
        -- Insist that command is a SELECT statement or a syntax check
        IF (CHARINDEX('set parseonly',LOWER(@cmd),1) = 1)
                BEGIN
                        -- run the syntax check command and return
                        EXEC (@cmd)
                        IF (@errorMsg is null)
                                SELECT 'Syntax is OK'
                        RETURN
                END

       IF (CHARINDEX('select ',LTRIM(LOWER(@cmd)),1) != 1)
                 BEGIN
                        SET @errorMsg = 'error: must be a select statement: ';
                        GOTO bottom;
                 END;
        SET @i = CHARINDEX('select ',LOWER(@cmd),1) + 7; -- point just past it
        --------------------------------------------------------------------------------
        -- limit the output to at most 1,000 rows.
        IF (CHARINDEX('all ',LTRIM(LOWER(substring(@cmd,@i,100)))) = 1)
                 SET @i = CHARINDEX('all ',LOWER(@cmd),1) + 4; -- point just past it
        IF (CHARINDEX('distinct ',LTRIM(LOWER(substring(@cmd,@i,100)))) = 1)
                 SET @i = CHARINDEX('distinct ',LOWER(@cmd),1) + 9; -- point just past it
        IF (@limit > 0)
           BEGIN
             IF (CHARINDEX('top ',LTRIM(LOWER(substring(@cmd,@i,100)))) != 1)   -- if no
limit specified
                BEGIN
                SET @cmd = STUFF(@cmd,@i,0, @top)  -- add one
                END
             ELSE                                     -- a limit was included
                BEGIN                              -- assure that it is less than 1000.
                SET @i = CHARINDEX('top ',LOWER(@cmd),1) + 4
                DECLARE @count int;
                DECLARE @len int;
                SET @i = @i + (LEN(substring(@cmd,@i,1000)) -
LEN(LTRIM(substring(@cmd,@i,1000))));
                SET @len = CHARINDEX(' ',@cmd + ' ',@i) - @i;
                IF (dbo.fIsNumbers(@cmd, @i, @i+@len-1) = 1 )
                        SET @count = CAST(SUBSTRING(@cmd,@i,@len) as int);
                IF ((@count is null) or (@count < 1 ) or (@count > @limit))
                        SET @errorMsg = 'error: limit is '+ @top;
                END
           END
        --------------------------------------------------------------------------------
        -- execute the command, returning the rows.
bottom:
        IF (@errorMsg is null)                          -- if good,
                begin
                --- log the command if there is a weblog DB
                -- variables to track and log SQL performance.
                declare @startTime datetime, @endTime datetime
                declare @busyTime bigint, @rows bigint, @IOs bigint
                if (0 != (select count(*) from master.dbo.sysdatabases where name =
'weblog'))
                        begin
                        set @startTime = getUtcDate();
                        set @busyTime = @@CPU_BUSY+@@IO_BUSY
                        set @IOs = cast(@@TOTAL_READ as bigint)+cast(@@TOTAL_WRITE as
bigint)
                        insert WebLog.dbo.SqlStatementLogUTC
                       values(@startTime,@webserver,@winName, @clientIP,
                              @serverName, @dbName, @access, @inputCmd, @isVisible)
                        end

                --=====================================================
                -- EXECUTE THE COMMAND
                EXEC (@cmd)                             -- return the data
                select @rows = @@rowCount, @error = @@error
                -------------------------------------------------------
```

```
                -- record the performance when (if) the command completes.
                if (@startTime is not null)
                        begin
                        set @endTime = getUtcDate();
                        insert WebLog.dbo.SqlPerformanceLogUTC
                         values (@startTime,@webserver,@winName, @clientIP,
                                datediff(ms, @startTime, @endTime)/1000.0,      --
elapsed time
                                ((@@CPU_BUSY+@@IO_BUSY)-@busyTime)/1000.0,      -- busy
time
                                @rows, @@procid, 0,'')
-- rows returned
                        end

                end             -- end of good command case
        ------------------------------------------------------------------------
        -- bad input command case
        ELSE                    -- if error
               BEGIN
               IF (0 != (select count(*) from master.dbo.sysdatabases where name =
'weblog'))
                        begin
                        set @startTime = getUtcDate();
                       insert WebLog.dbo.SqlStatementLogUTC
                              values(@startTime,@webserver,@winName, @clientIP,
                                     @serverName, @dbName, @access, @inputCmd,
@isVisible)
                        insert WebLog.dbo.SqlPerformanceLogUTC
                              values(@startTime,@webserver,@winName, @clientIP,
                              0,0,0,@@procid, -1, @errorMsg)
                        end
                  SELECT @errorMsg + @cmd as error_message; -- return the error message
               END
        END
-------------------------------------------------------------------------------------
GO
--


--=====================================================================
IF EXISTS (SELECT name FROM   sysobjects
           WHERE  name = N'spExecuteSQL2' )
       DROP PROCEDURE spExecuteSQL2
GO
--
CREATE PROCEDURE spExecuteSQL2(
       @cmd varchar(8000),
       @webserver varchar(64) = '',   -- the url
       @winname varchar(64)   = '',   -- the windows name of the server
       @clientIP varchar(16)  = '0',   -- client IP address
       @access varchar(64)    = ''    -- subsite of site,  if 'collab' statement 'hidden'
)
-------------------------------------------------------------------------------
--/H Procedure to safely execute an SQL select statement
-------------------------------------------------------------------------------
--/T The procedure runs and logs a query, but does not parse
--/T it. <br>
--/T See also spExecuteSQL
-------------------------------------------------------------------------------
AS
BEGIN
       SET NOCOUNT ON
       --
       DECLARE @error int,            -- error number
               @errorMsg varchar(100),        -- error msg
               @serverName varchar(32), -- name of this databaes server
               @dbName varchar(32),    -- name of this database
               @isVisible int,                -- flag says sql is visible to internet
queries
               @startTime datetime,
               @endTime datetime,
```

```
            @busyTime bigint,
            @rows bigint,
            @IOs bigint
        --
        SET @isVisible = 1;
        SET @serverName = @@servername;
        SELECT @dbname = [name] FROM master.dbo.sysdatabases WHERE dbid = db_id();
        ------------------------------------------------------------
        IF (@errorMsg is null)        -- if good,
            BEGIN
               ---------------------------------------------
               --- log the command if there is a weblog DB
               -- variables to track and log SQL performance.
               ---------------------------------------------
               if (0 != (select count(*) from master.dbo.sysdatabases where name =
'weblog'))
                  begin
                       set @startTime = getUtcDate();
                       set @busyTime = @@CPU_BUSY+@@IO_BUSY
                         set @IOs = cast(@@TOTAL_READ as bigint)+cast(@@TOTAL_WRITE as
bigint)
                       insert WebLog.dbo.SqlStatementLogUTC
                              Values(@startTime,@webserver,@winName, @clientIP,
                              @serverName, @dbName, @access, @cmd, @isVisible)
                    end
               -------------------------
               -- execute the command
               -------------------------
               exec (@cmd)
               set @rows = @@rowCount
               ------------------------------------------------------------
               -- record the performance when (if) the command completes.
               ------------------------------------------------------------
               if (@startTime is not null)
                    begin
                       set @endTime = getUtcDate();
                       insert WebLog.dbo.SqlPerformanceLogUTC
                              values (@startTime,@webserver,@winName, @clientIP,
                              datediff(ms, @startTime, @endTime)/1000.0,      -- elapsed
time
                              ((@@CPU_BUSY+@@IO_BUSY)-@busyTime)/1000.0,      -- busy
time
                              @rows, @@procid, 0,'')                       -- rows
returned
                    end
            END            -- end of good command case
        ------------------------------------------------------------------------
        -- bad input command case
        ELSE                    -- if error
            BEGIN
               IF (0 != (select count(*) from master.dbo.sysdatabases where name =
'weblog'))
                    begin
                       set @startTime = getUtcDate();
                       insert WebLog.dbo.SqlStatementLogUTC
                              values(@startTime,@webserver,@winName, @clientIP,
                              @serverName, @dbName, @access, @cmd, @isVisible)
                       insert WebLog.dbo.SqlPerformanceLogUTC
                              values(@startTime,@webserver,@winName, @clientIP,
                              0,0,0,@@procid, -1, @errorMsg)
                    end
               SELECT @errorMsg + @cmd as error_message; -- return the error message
            END
        ------------------------------------------------------------------
END
GO
```

```
/*

 (1) have the sql.asp pass the following additonal  params to spExecuteSql()
       client IP address       -- this allows us to correlate the query with other web
log activity
       webServer               -- the url of the website making this request (gives
collab, collabpw, dr1, dr2, ... )

 (2) redefine the SQLstatement log to have the following fields
     go from the two fields (theTime, sql) to the following
 CREATE  TABLE SqlStatementLog (
       yy      int not null,   -- the year
       mm      int not null,   -- the month
       dd      int not null,   -- the day
       hh      int not null,   -- the hour
       mi      int not null,   -- the minute
       ss      int not null,   -- the second
       seq     int identity(1,1)  -- uniquifier
       clientIP  char(12)      not null default('',    -- ip address of client
       WebServer varchar(32)   not null default('',    -- name of webserver
       DB      varchar(32)     not null default(''),   -- name of database being queried
       access  varchar(32)    not null default(''),   -- the kind of access (collab,
public,..)
       procID  int             not null default(0),    -- process ID used in cancel
query
       SQL     varchar(7000) not null default(''),             -- the query
       isVisible bit,
       primary key (yy,mm,dd,hh,mm,ss,seq,clientIP)
       )
*/


-----------------------------------------------------------------------
IF EXISTS (SELECT name FROM   sysobjects
          WHERE  name = N'spLogSqlStatement' )
       DROP PROCEDURE spLogSqlStatement
GO
--
CREATE PROCEDURE spLogSqlStatement (
       @cmd VARCHAR(8000) OUTPUT,
    @webserver      VARCHAR(64) = '',   -- the url
       @winname       VARCHAR(64) = '',   -- the windows name of the server
    @clientIP       VARCHAR(16)  = 0,   -- client IP address
       @access        VARCHAR(64) = '',   -- subsite of site,  if 'collab' statement
'hidden'
       @startTime     datetime            -- time the query was started
)
-------------------------------------------------------------------------------
--/H Procedure to log a SQL query to the statement log.
-------------------------------------------------------------------------------
--/T Log the given query and its start time to the SQL statement log.  Note
--/T that we are logging only the start of the query yet, not a completed query.
--/T All the SQL statements are journaled into WebLog.dbo.SQLStatementlog.
--/T <samp>EXEC dbo.spLogSqlStatement('Select count(*) from
PhotoObj',getutcdate())</samp>
--/T See also spLogSqlPerformance.
-------------------------------------------------------------------------------
AS
BEGIN
       SET NOCOUNT ON
       DECLARE @error      INT;                    -- error number
       DECLARE @serverName varchar(32);            -- name of this databaes server
       DECLARE @dbName     VARCHAR(32);            -- name of this database
       SET     @serverName = @@servername;
       SELECT @dbname = [name] FROM master.dbo.sysdatabases WHERE dbid = db_id()
       DECLARE @isVisible  INT;                    -- flag says sql is visible to
internet queries
       SET @isVisible = 1;
       IF (UPPER(@access) LIKE '%COLLAB%') SET @isVisible = 0;  -- collab is invisible
       -------------------------------------------------------------------------
```

```
        --- log the command if there is a weblog DB
        if (0 != (select count(*) from master.dbo.sysdatabases where name = 'weblog'))
            begin
                insert WebLog.dbo.SqlStatementLogUTC
                values(@startTime,@webserver,@winName, @clientIP,
                      @serverName, @dbName, @access, @cmd, @isVisible)
            end
END
GO



-----------------------------------------------------------------
IF EXISTS (SELECT name FROM   sysobjects
          WHERE  name = N'spLogSqlPerformance' )
       DROP PROCEDURE spLogSqlPerformance
GO
--
CREATE PROCEDURE spLogSqlPerformance (
        @webserver      VARCHAR(64) = '',   -- the url
        @winname       VARCHAR(64) = '',   -- the windows name of the server
        @clientIP       VARCHAR(16)  = 0,   -- client IP address
        @access        VARCHAR(64) = '',   -- subsite of site,  if 'collab' statement
'hidden'
        @startTime     datetime,           -- time the query was started
        @busyTime      bigint      = 0,    -- time the CPU was busy during query execution
        @endTime       datetime    = 0,    -- time the query finished
        @rows           bigint      = 0,    -- number of rows returned by the query
        @errorMsg      VARCHAR(1024) = ''  -- error message if applicable
)
-------------------------------------------------------------------------------
--/H Procedure to log success (or failure) of a SQL query to the performance log.
-------------------------------------------------------------------------------
--/T The caller needs to specify the time the query was started, the number of <br>
--/T seconds (bigint) that the CPU was busy during the query execution, the    <br>
--/T time the query ended, the number of rows the query returned, and an error <br>
--/T message if applicable.  The time fields can be 0 if there is an error.
--/T <samp>EXEC dbo.spLogSQLPerformance('skyserver.sdss.org','',,'',getutcdate())</samp>
--/T See also spLogSqlStatement.
-------------------------------------------------------------------------------
AS
BEGIN
        SET NOCOUNT ON
        ------------------------------------------------------
        -- record the performance when (if) the command completes.
        IF ( (@startTime IS NOT NULL) AND (@startTime != 0) AND
             (@busyTime != 0) AND (@endTime != 0) AND (LEN(@errorMsg) = 0) )
            BEGIN
                INSERT WebLog.dbo.SqlPerformanceLogUTC
               VALUES (@startTime,@webserver,@winName, @clientIP,
                        DATEDIFF(ms, @startTime, @endTime)/1000.0,      -- elapsed time
                        ((@@CPU_BUSY+@@IO_BUSY)-@busyTime)/1000.0,      -- busy time
                        @rows, @@PROCID, 0,'')                                      -
- rows returned
            END
        ELSE
            BEGIN
                IF ( (@startTime IS NULL) OR (@startTime = 0) )
                    SET @startTime = GETUTCDATE();
                INSERT WebLog.dbo.SqlPerformanceLogUTC
               VALUES (@startTime,@webserver,@winName, @clientIP,
                      0,0,0, @@PROCID, -1, @errorMsg)
            END
END
GO

-----------------------------------------------------------------------
PRINT '[spSQLSupport.sql]: SQL Support procs and functions created.'
```