Project Château Kebob:

Big Data / BI SQL Audit Project

**Ayad Shammout & Denny Lee**

**2013**

Table of Contents

[1 Overview 2](#_Toc354523087)

[2 Implementation Overview 3](#_Toc354523088)

[3 The SQL Server Audit Database 4](#_Toc354523089)

[3.1 Tables 4](#_Toc354523090)

[3.2 Dimensions 4](#_Toc354523091)

[4 Uploading the Audit Logs 5](#_Toc354523092)

[4.1 Connections 5](#_Toc354523093)

[4.2 Package Variables 5](#_Toc354523094)

[5 Solution Deployment 9](#_Toc354523095)

[5.1 Big Data Audit Project 9](#_Toc354523096)

[5.1.1 Acquire the Solution 9](#_Toc354523097)

[5.1.2 Create the SQLAudit Database 9](#_Toc354523098)

[5.1.3 Create Hive External Table <*Place Holder for Denny to add some instructions>* 9](#_Toc354523099)

[5.1.4 Install the AuditLog\_To\_HadoopPackage SSIS package 9](#_Toc354523100)

[6 Create BI Semantic Data Model 10](#_Toc354523101)

[To create a connection to a SQLAuditLog Hive database 11](#_Toc354523102)

[7 Reporting 13](#_Toc354523103)

[8 Conclusion 14](#_Toc354523104)

# Overview

For many organizations large and small, understanding compliance is a core fundamental to meeting regulations requirements. Regulatory requirements such as Sarbanes-Oxley, PCI and HIPPA are a major driver for compliance. However, even companies that are not impacted by regulatory requirements often need to reach compliance with their own organizational policies. As you can see in Figure-1, Auditing is the key component in for all different regulations.

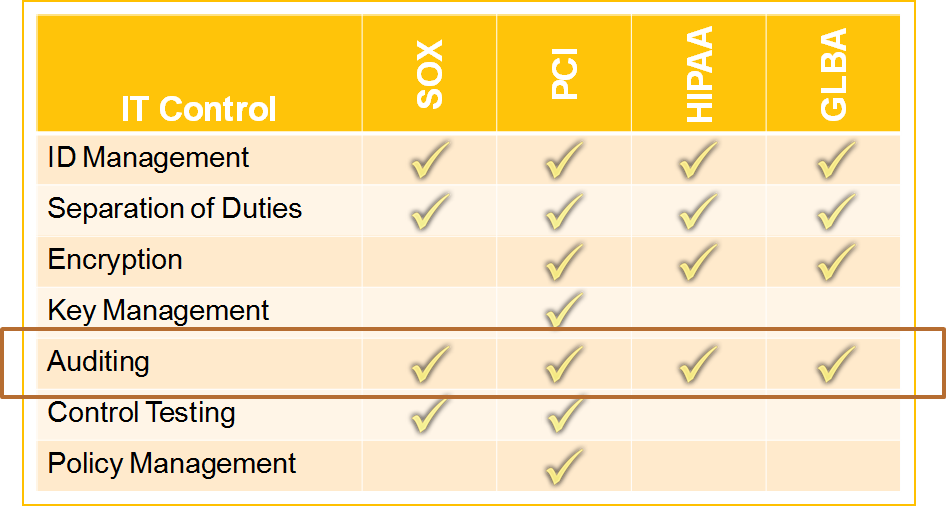


Figure 1

The purpose of this project is to expand on the “[Reaching Compliance: SQL Server 2008 Compliance Guide](http://www.microsoft.com/en-us/download/details.aspx?id=6808)” to more easily handle larger volumes of structured and unstructured data. The end goal is to gain richer and deeper insight using the latest analytics. To achieve this, we are building a Big Data-to-BI project involving HDInsight (Hadoop on Windows or Azure), SQL Server 2012, SQL Server Analysis Service 2012 Tabular, Integration Services, PowerPivot, and Power View.

The purpose of this SDK is to provide a set of tools and jumpstart with scripts to implement the Auditing project involving HDInsight, SQL Server 2012, PowerPivot and Power View.

# Implementation Overview

The basic implementation of the Auditing and Reporting solution is shown in the figure below.

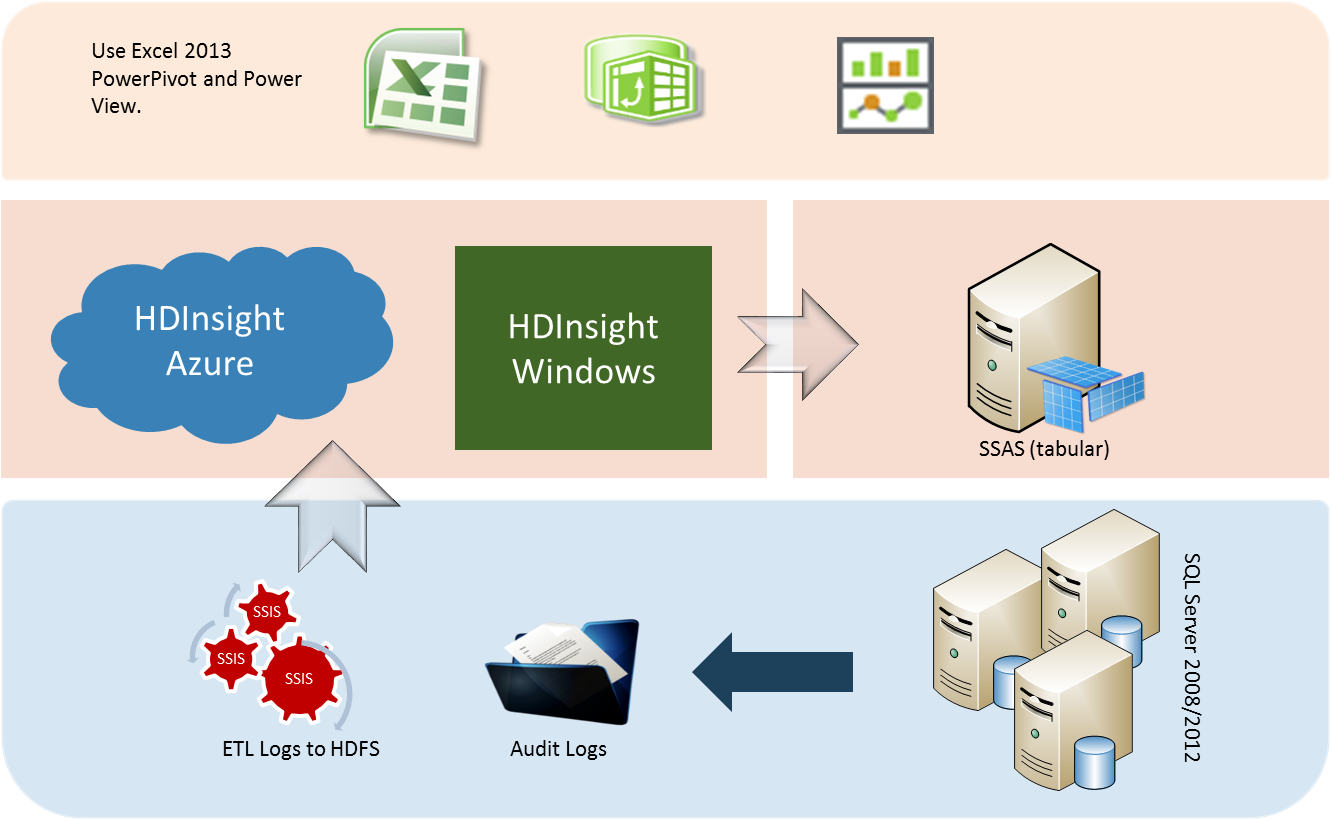


Figure 2

The general flow of data in this solution is that audit events are created on any number of SQL Servers (2008 and 2012) in the environment and are set to log to the file system. The audit logs will be stored directly to a central network file share. A scheduled SQL Server Agent Job runs an SSIS package that reads the audit log files, combines them into larger file sizes (250MB to 1GB file size) and uploads them to Azure BLOB Storage; the storage source for HDInsight on Azure.

Once audit logs stored in Azure BLOB Storage we use Hive, a data warehouse framework for Hadoop, which facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets.

Reports are created within Excel 2013 using HiveODBC to connect to audit logs in the form of Hive external tables in HDInsight to allow compliance auditors and server administrators to assess, observe trends, and ultimately take meaningful action for server compliance.

This information would then be fed back to the appropriate security, administrator and application development teams to enact policies to approve levels of compliance.

As the system evolves, teams may load additional application audit logs into the repository, which could help tie these SQL Server specific activities to application and business activities.

# The SQL Server Audit Database

The SQL Server audit database records the Audit log execution, processing and loading. It stores information about these actions as dimension members. The following are the tables created in SQLAuditLog Database.

## Tables

**AuditFile**

This table stores information about the auditfile created by the server audit; though there is an additional purpose. There is a bug in the current release of SQL Server that the Login Audit does not consistently capture the server instance name of the server that recorded the login. When this case is encountered, the SSIS package looks to the auditfile table to determine what server instance recorded the audit file currently being read. While the server name is not contained directly in the audit file, the first audit file in a given server audit will contain an audit changed event that will contain the server instance name. The audit files are tied together by an audit guid that uniquely identifies the audit across all server instances. Using these pieces of information, the audit file insert statement is able to determine which server instance recorded this file.

**ImportExecution**

Each time the SSIS package that processes the logs is run, it inserts a row into the ImportExecution table. The table also records the start and finish time of the package execution. The table not only tracks what SSIS package processing is performed, but also what logs are processed in which run. This allows you to uniquely track down which audit log transformation transaction (i.e. audit the audit).

**ImportFile**

As each file is processed as part of an import execution, a row is added to the import file and includes the number of rows read from that file.

## Dimensions

**dimAuditedAction**

Each audit log entry contains an action\_id that references an action in the DMV sys.dm\_audit\_actions. The AuditedAction table contains a list of all action\_ids with action description. Example actions include DML actions such as update, insert and delete, server operations such as user logged in or out, and database operations such as resizing a database or running a DBCC command.

**dimAuditClass**

Each action audited was performed against some object. The dimAuditClass table contains a list of classes that these objects belong to. The list of possible classes exists in sys.dm\_audit\_class\_type\_map. Example classes include Database, User, Application Role, Table, or View.

**dimDate**

Date is a common dimension, with several possible hierarchies. This helps in viewing audit data by date series.

**dimGeo**

The geography dimension is to map audit events IP address with locations. The dimGeo table contains a list of IP addresses and the city location to locate audited users.

# Uploading the Audit Logs

The audit files are uploaded to HDInsight Storage (Azure Blob Storage) with an SSIS package that reads all of the audit log files from a directory or file share, loads the new audit log data and combine them into a set of larger files, copies these files to HDInsight storage and archives closed audit log files to a specified archive directory or file share. The following sections describe the various components used by this SSIS package.

## Connections

There are two connections used by the SSIS package:

1. SqlAuditLogRepository. This connection should point to the SQL Audit Log repository database. The service account used by the SQL Server Service identified in this connection must have access to the directory that holds the audit logs.
2. AuditLogOutputFile: This connection is pointed to the directory or file share where the audit logs will be exported to HDInsight storage after they are being processed.

## Package Variables

The following variables are defined in the load audit logs package. The bolded rows specify information that may be provided to the package at execution time. Italic rows indicate that the variable’s value is determined at runtime based on the execution of a script, evaluation of an expression, returned from a stored procedure, etc. There are two variables (min and max event time) that are both bolded and italic. If their corresponding override flags are set, then they must be specified at package execution but if the flags are not set then the variables are determined through a variable initialization script.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | | Description | Sample Value |
| *file\_name* | *String* | | *Loop variable for the foreach loop that iterates through the files in the directory specified in the variable “LogFilePath”* | *Sql-Audit-20080707-021303\_D89E1867-D209-443A-9A61-2F5DB36D6DDD\_0\_128606566711250000.sqlaudit* |
| *file\_offset* | *Int64* | | *Used by the “getauditLogCommand” to tell SQL Server how far into the file to seek before it starts reading audit events.* |  |
| *maxEventTime* | | ***String*** | *Used in the where clause of the getAuditLogCommand to specify the greatest event time that should be read. By default, this will be 5 minutes prior to the start of the package execution. The values may be specified in conjunction with the overrideMaxEventTime variable when specifying a time window to load audit logs (for example, when loading historical logs)* | ***2008-07-16-01:24:45*** |
| overrideMinEventTime | **Boolean** | | See notes on minEventTime | **2008-07-16-01:24:45** |
| overrideMaxEventTime | **Boolean** | | See notes on maxEventTime | **False** |
| *minEventTime* | ***String*** | | *Used in the where clause of the getAuditLogCommand to specify the earliest event time that should be read. By default, this will be the last event time processed by the previous execution of this package. The values may be specified in conjunction with the overrideMinEventTime variable when specifying a time window to load audit logs (for example, when loading historical logs)*  *This value is initialized to ‘1/1/1900’.* |  |
| LogFilePath | **String** | | Path to the directory or file share that contains the audit log files that should be processed by this package. | C:\SQLAuditLogs |
| auditLogArchivePath | **String** | | Path to which the package will archive “closed files.” |  |
| *getAuditLogCommand* | *String* | | *Command generated by the SSIS package to read an audit log file.* | *select \* from [aud].[fn\_AuditFileGet]('c:\AuditLogFiles\AuditFile.sqlAudit', 0) where event\_time between ‘1/1/1900’ and ‘7/15/2008 01:24:45’* |
| *import\_id* | *Int32* | | *Identity value from the ImportExecution table. This id is placed into all of the fact table rows that are loaded into the repository database.* |  |
| *filesToArchive* | *Object* | | *ADO.NET DataSet with one table that contains a list of files that should be moved from the audit log directory to the audit log archive directory.* |  |
| *fileToArchive* | *String* | | *Loop variable used by the foreach loop that iterates over the rows in the filesToArchive data table.* |  |
| *UploadOutputFilePath* | *String* | | Path to the directory or file share that contains the processed audit log files that will be copied to HDInsight storage by this package. |  |
| *FileLocation* | *String* | | Variable which concatenate UploadOutputFilePath and Audit file name with timestamp. To be used as a dynamic filename in **AuditLogOutputFile** file connection. |  |

**Control Flow**

This following diagram describes the control flow of this SSIS package. The basic flow is

1. Start with initializing variables
2. Load logs that should be loaded
3. Archive logs that should be archived (if loaded before)
4. Generate Output file name with date time stamp
5. Load, read and process log
6. Upload (copy) output audit log files to HDInsight storage

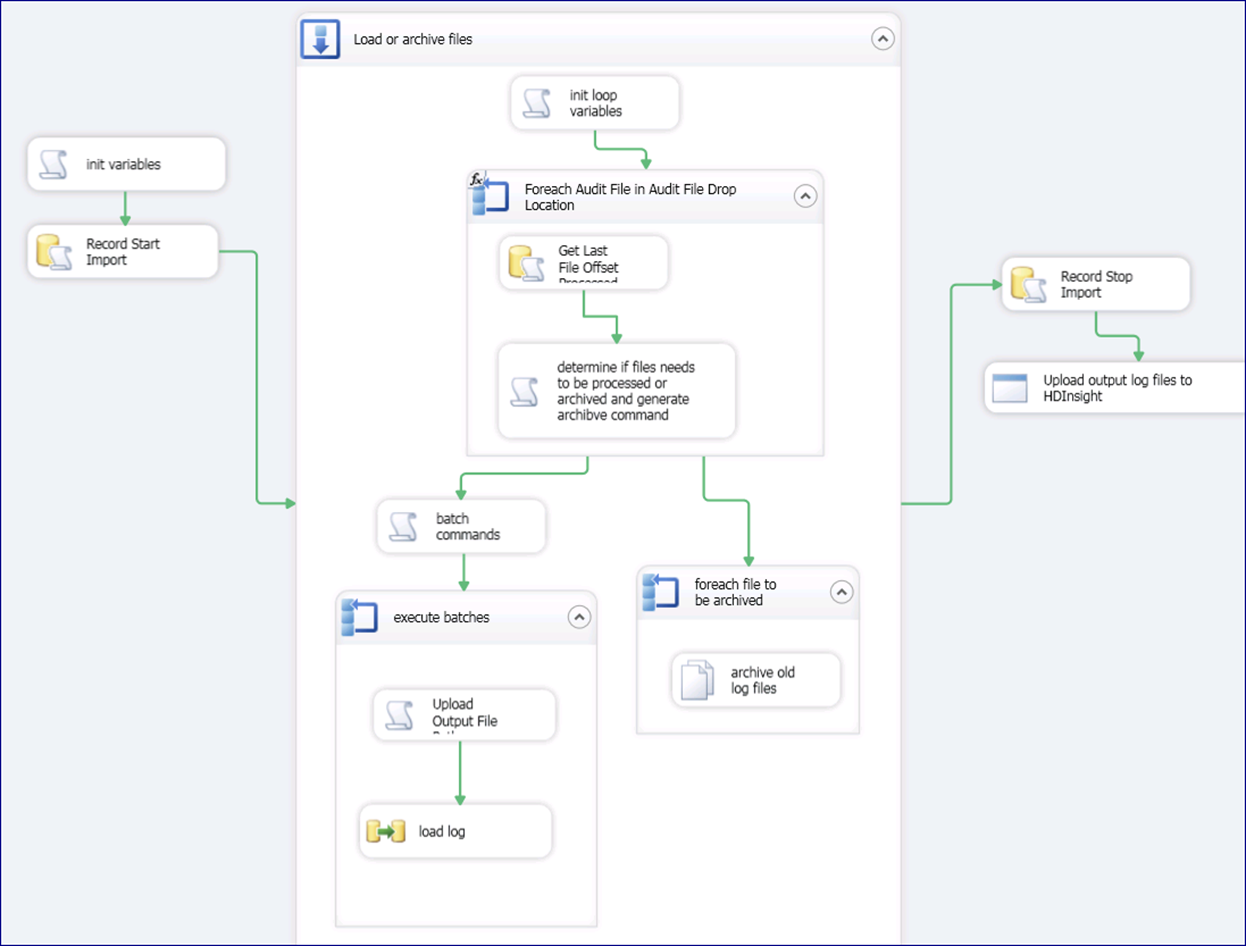


Figure 3

# Solution Deployment

## Big Data Audit Project

This project was a collaboration between Ayad Shammout (SQL & BI Consultant) and Denny Lee (SQL BI Product Team). The goal was to create a working template of an end-to-end audit project that obtains the audit logs, transforms the data, loads the data into HDInsight storage, and build PowerPivot and Power View reports. For more information, see the (to be published) *Audit Project Technical Spotlight*.

### Acquire the Solution

It should contain the following files:

* SQLAuditLog\_DB\_Backup.bak – the SQL DB backup file that creates your SQLAuditLog database
* SQLAuditLogs\_To\_HDInsight\_Package.dtsx – an SSIS package that performs the task of obtaining, extracting, transforming, and loading the audit data and archiving the audit logs
* HiveLinkedServer.sql – create the Linked Server connection to HDInsight.
* CreateExternalHiveTable.txt – create External Hive Table in HDInsight.

**Note: The SSIS package requires a copy tool i.e. AzCopy utility tool to copy data to Hadoop.** The utility can be downloaded from [Github.](https://github.com/downloads/WindowsAzure/azure-sdk-downloads/AzCopy.zip)

### Create the SQLAuditLog Database

This database will be tracking your audit log load process.

1. Restore the database from backup file **SQLAuditLog\_DB\_Backup.bak** in SQL Server Management Studio

### Create Hive External Table

This file contains a CREATE EXTERNAL TABLE command that allows you to create a Hive (data warehousing framework on top of Hadoop) table on top of the Azure Blob Storage (ASV) file folder. By using this Hive table, you can query it using HiveQL (Hive Query Language) – a SQL like query language. For more information about Hive, please reference <http://hive.apache.org>.

To run the CreateExternalHiveTable.txt:

* Update the LOCATION field to point to your ASV location (i.e. replace the ‘asv://…/’ section with the location you have placed your audit logs
* Run this either by copy/paste it into the Interactive Hive Console or use “hive –f” within the Hadoop Command Shell.

### Install the AuditLog\_To\_HadoopPackage SSIS package

This is the complete SSIS project which include the necessary dtsx file so that you can execute it.

The following connection and variables have to be set according to your environment.

#### Connections:

1. **SQLAuditLog:** this is a connection to **SQLAuditLog** SQL database which records the Audit log execution, processing and loading. *You need to change the server name in the connection properties to point to your SQL Server which hosts the SQLAuditLog db.*
2. **AuditLogOutputFile:** this is a connection to the directory or file share contains the processed audit log files that will be copied to HDInsight storage by this package.

#### Variables: You need to set the following variables:

1. **LogFilePath:** Path to the directory or file share that contains the audit log files that should be processed by this package. Example: C:\SQLAuditLogs
2. **AuditLogArchivePath:** Path to which the package will archive “closed files.

Example C:\SQLAuditLogs\Archive

1. **FileToArchive:** Loop variable used by the foreach loop that iterates over the rows in the filesToArchive data table. Example C:\SQLAuditLogs\Archive
2. **UploadOutputFilePath:** Path to the directory or file share that contains the processed audit log files that will be copied to HDInsight storage by this package.

Example C:\SQLAuditLogs\blob-data

**Execute the SSIS Package**

When the package is executed, it assumes that there are log files in the C:\SQLAuditLogs location (as noted above) and after the log files have been processed, they will be moved to the C:\SQLAuditLogs\Archive folder.

1. From a command prompt, change to your **SQLAuditLogs\_To\_HDInsight\_Package.dtsx** directory (e.g., C:\Program Files\Microsoft SQL Server\100\DTS\Packages\SQLAuditLoader)

Execute the command:

dtexec /File SQLAuditLogs\_To\_HDInsight\_Package.dtsx

1. From SQL Server Management Studio, create a job to execute the dtsx file.
2. From SQL Server Data Tool (SQL 2012), open the project and execute.

Because there are a lot of audit log files, it is suggested that you run this SSIS package periodically (e.g., every 3 hours or less) to ensure that the data is loaded in a timely manner.

# Create BI Semantic Data Model

After loading the Audit logs into HDInsight storage, we use Hive which is a data warehouse framework for Hadoop which facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets. Although Hive supports ad-hoc queries for Hadoop through HiveQL, query performance is often prohibitive for even the most common BI scenarios. A better solution is to bring relevant Hadoop data into SQL Server Analysis Services Tabular by using HiveQL. Analysis Services can then serve up the data for ad-hoc analysis and reporting. There is no direct way to connect an Analysis Services Tabular database to Hadoop. A common workaround is to create a Linked Server in a SQL Server instance using HiveODBC which uses it through OLE DB for ODBC. The HiveODBC driver can be downloaded from [here](http://hortonworks.com/download/download-archives/).

Once the HiveODBC driver is created, you can execute the **HiveLinkedServer.sql** script(included in this solution). SQL Server can serve as an intermediary and Analysis Server can connect to Hadoop via Hive Linked Server connection in SQL Server, so Hive appears as an OLE DB-based data source to Analysis Services.

The following components need to be configured to establish connectivity between a relational SQL Server instance and the Hadoop/Hive data warehouse:

* A system data source name (DSN) “**SQLAuditHive**” for the Hive ODBC connection. The DSN points to the host that is running Hadoop/Hive and specifies important connection parameters. System DSNs can be created by using the Data Sources (ODBC) administrative tool.
* A linked server object. The Transact-SQL script illustrates how to create a linked server that points to a Hive data source via MSDASQL. The system DSN in this example is called “Hive DW”.

EXEC master.dbo.sp\_addlinkedserver

@server = N'SQLHive', @srvproduct=N'HIVE',

@provider=N'MSDASQL', @datasrc=N'SQLHive',

@provstr=N'Provider=MSDASQL.1;Persist Security Info=True;User ID=UserName;

Password=pa$$word;

**Note**: Replace the User ID “UserName” and password “pa$$word” with a valid username and password to connect to Hadoop.

* An SQL statement that is based on an OpenQuery Transact-SQL command. The OpenQuery command connects to the data source, runs the query on the target system, and returns the ResultSet to SQL Server. The following Transact-SQL script illustrates how to query a Hive table from SQL Server:

SELECT \* FROM OpenQuery(SQLHive, 'SELECT \* FROM sql\_audit\_asv\_raw;')

Once the Linked Server is created on the computer running SQL Server, it is straightforward to connect Analysis Services to Hive in SQL Server Data Tools. You can start by creating a new SQL Analysis Services Tabular project

### To create a connection to a SQLAuditLog Hive database

1. In SQL Server Data Tools, click on the Model menu, and then click Import from Data Source.

This launches the Table Import Wizard which guides you through setting up a connection to a data source.

1. In the Table Import Wizard, under Relational Databases, click Microsoft SQL Server, and then click Next.
2. In the Connect to a Microsoft SQL Server Database page, in Friendly Connection Name, type SQLAuditLog DB from SQL.
3. In Server name, type the name of the SQL Server database that hosts the SQL Linked Server connection to Hadoop/Hive.
4. In the Database name field, click the down arrow and select SQLAuditLog, and then click Next.
5. In the Impersonation Information page, you need to specify the credentials Analysis Services will use to connect to the data source when importing and processing data. Verify Specific Windows user name and password is selected, and then in User Name and Password, enter your Windows logon credentials, and then click Next.

|  |
| --- |
| **NoteNote** |
| Using a Windows user account and password provides the most secure method of connecting to a data source. For more information, see [Impersonation (SSAS Tabular)](http://msdn.microsoft.com/en-us/library/gg492180). |

1. In the Choose How to Import the Data page, verify write a query that will specify the data to import is selected. Rename the query name to AuditEvents and in the SQL Statement window, type the following:

SELECT \* FROM OpenQuery (SQLHive, 'SELECT \* FROM sql\_audit\_asv\_raw;')

And then click Finish.

1. Next import the dimension from SQLAuditLog db.
2. Click on the Model menu, and then click Existing Connections.
3. This launches the existing connection, select the SQLAuditLog connection, and then click Open.
4. In the Choose How to Import the Data page, verify Select from a list of tables to choose the data to import is selected. You want to select from a list of tables and views, so click Next to display a list of all the source tables in the source database.
5. In the Select Tables and Views page, select the check box for the following tables: DimAuditActions, DimDate, DimGeo, and DimAuditClass. Click Finish.
6. Once the above tables were imported, now you can create a relationships between AuditEvents table and the dimension tables as shown in Figure 4 below.

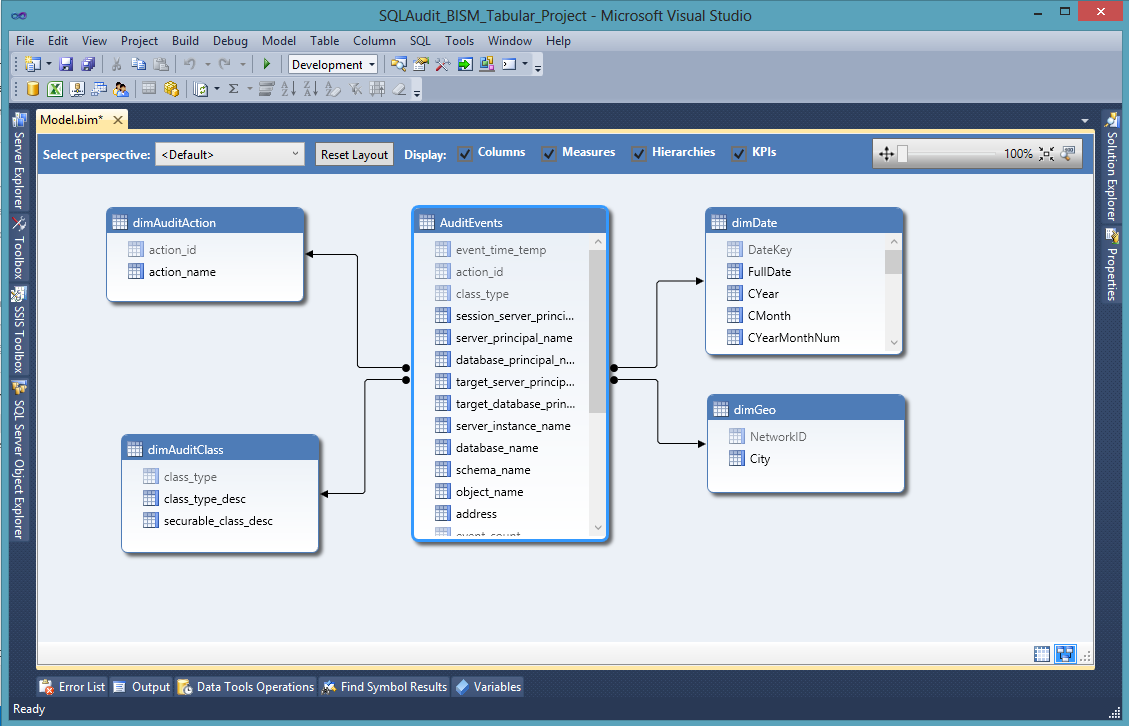
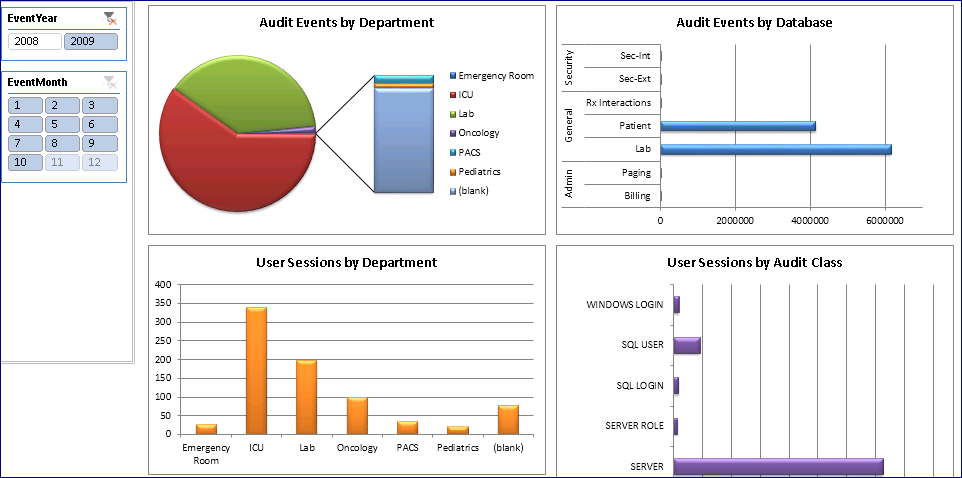


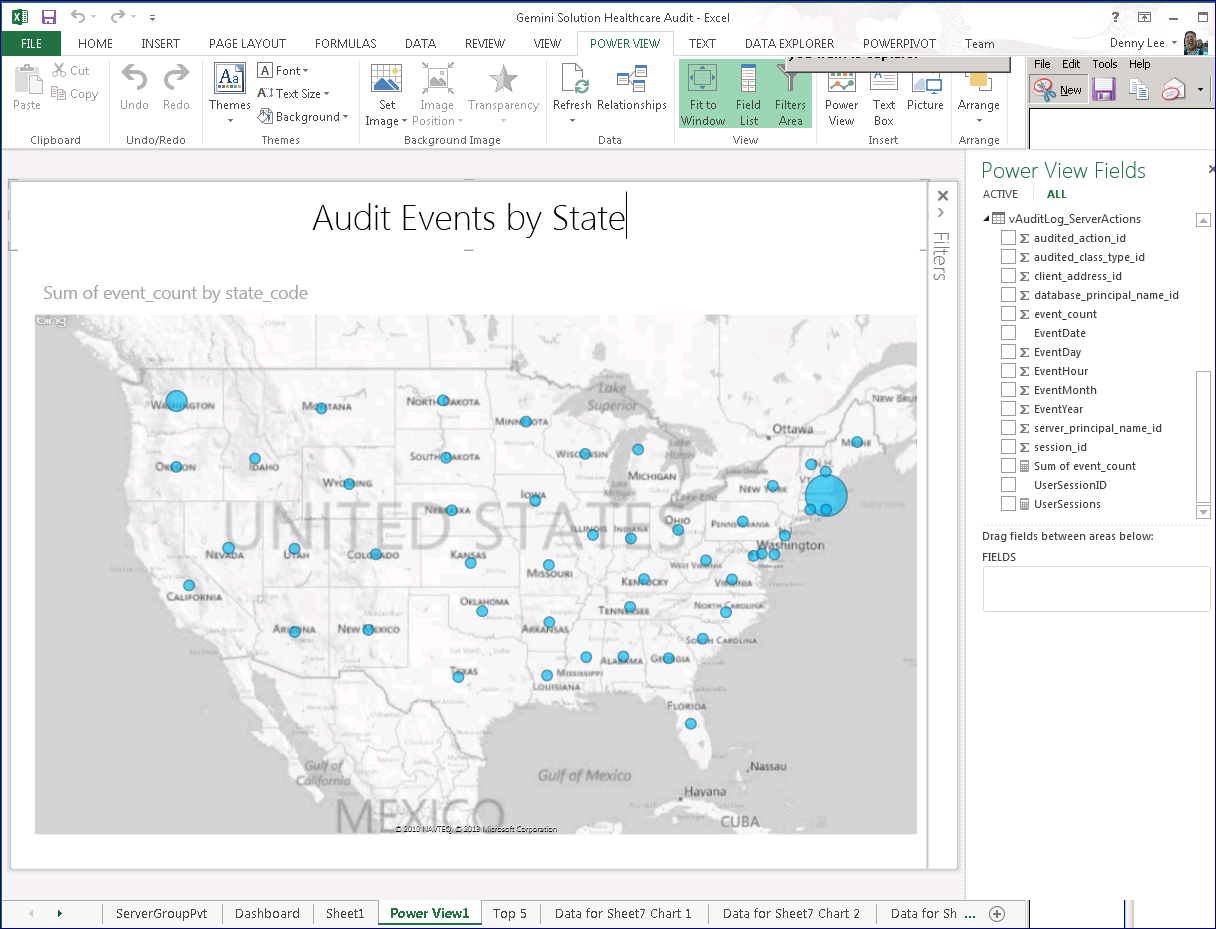
Figure 4

1. Now the Model is ready to be deployed to SQL Server Analysis Services (SSAS) Tabular instance.

# Reporting

After deploying the BI Semantic Data Model Tabular to SSAS Tabular instance, you can use [Power View](http://www.microsoft.com/en-us/bi/Products/PowerView.aspx) to start creating analytical reports against the Tabular database. Power View enables quick and easy visualization of Audit data and build dashboards and reports. Below are a sample of reports produced from Audit Data.





# Conclusion

**Known Issues**

* There are times where the audit logs do not record the server instance name; i.e., the server instance name field is empty within the audit logs. To work around this, the SSIS package will go through the entire audit log and determine the server instance name and assign that name to the entire log (because an audit log comes from only one server). But there are situations where even that will not work, because the entire file has no server instance name. To mitigate this, set up your audit logs with the naming convention

SQLAudit$*%Server$InstanceName%*\_%GUID%.sqlaudit

[aud].[fn\_GetServerInstanceName] will parse the name of the audit file, pull out the highlighted %Server$InstanceName%, and use this name if one does not already exist. To name your log file with this naming convention, when you create your server audit, alter the name of your audit with the format of SQLAudit$Server$InstanceName.