UCI INCIDENT DATA MART DESIGN

Business Intelligence

Brief Description

This data mart design is focused on how data from a csv file with over 140,000 rows from an event log collected by the University of California, Irvine Machine Learning Repository got extracted, transformed, and loaded into a data mart consisting of one FACT table and eight DIM tables using a Staging table.

Team Members:

Aaron Davila: adavila4@mail.usf.edu
Mariel Lanza: mariell@mail.usf.edu
Caleb Monestime: cmonestime@mail.usf.edu
Peyton Woble: pwoble@mail.usf.edu

Data Mart Design Description:

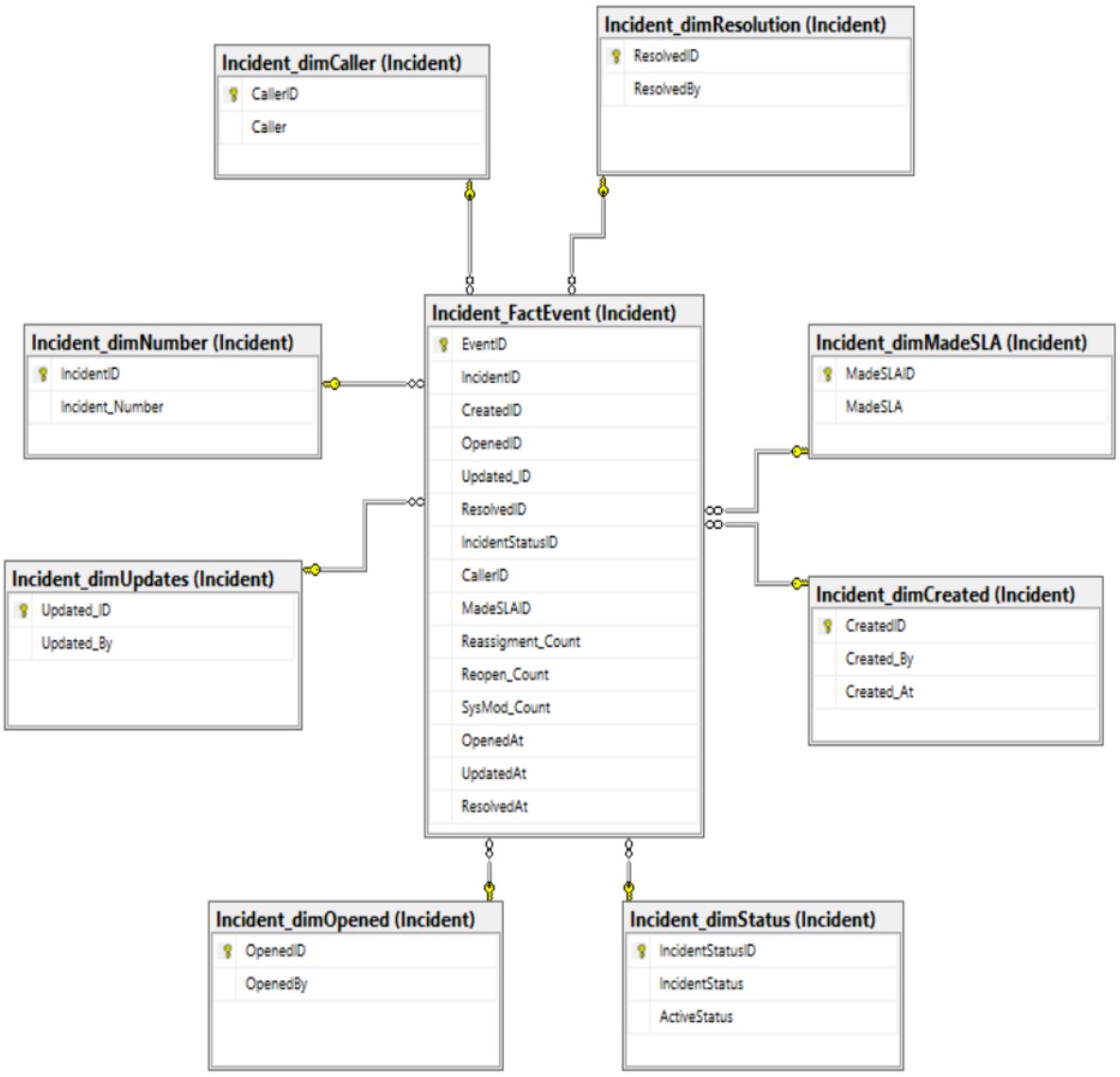
This document attempts to describe a proposed data mart design that can be used to analyze the *2017 Incident Management Process Enriched Event Log Data Set* provided by uci.edu. The design will be implemented and replicated in Microsoft SQL Server. SQL Server will be used to implement the analytical design in order to accommodate a variety of analytical technologies which could be used to perform the analysis. The analytical design is based off of the flat data structure included in the CSV file provided by uci.edu which examines a variety of incidents recorded by the University of California, Irvine Machine Learning Repository.

This document will examine the proposed analytical tables and relationships based on structure seen in this partial sample.

EventID	Incident Number	Incident Status	Active Status	Reassignment Count	Reopen Count	Sys Mod Count	Made SLA	Caller	Opened By	Opened At	Updated By	Updated At	Resolved By	Resolved At
95770	INC0023131	New	TRUE	0	0	0	TRUE	Caller 4484	Opened by 40	27/4/2016 11:04	Updated by 88	27/4/2016 11:08	Resolved by 100	27/4/2016 14:38
95771	INC0023131	Active	TRUE	0	0	1	TRUE	Caller 4484	Opened by 40	27/4/2016 11:04	Updated by 421	27/4/2016 14:37	Resolved by 100	27/4/2016 14:38
95772	INC0023131	Resolved	TRUE	0	0	2	TRUE	Caller 4484	Opened by 40	27/4/2016 11:04	Updated by 421	27/4/2016 14:38	Resolved by 100	27/4/2016 14:38
95773	INC0023131	Closed	FALSE	0	0	3	TRUE	Caller 4484	Opened by 40	27/4/2016 11:04	Updated by 908	2/5/2016 15:07	Resolved by 100	27/4/2016 14:38
95774	INC0023132	New	TRUE	0	0	0	TRUE	Caller 371	Opened by 17	27/4/2016 11:05	Updated by 908	27/4/2016 11:05	Resolved by 15	28/4/2016 15:18

In the above data sample taken from the source CSV file, the proposed analytical design will consider the measurable columns such as “Reassignment_Count”, “Reopen_Count”, “SysMod_Count”, “OpenedAt”, “UpdatedAt”, and “ResolvedAt” as “FACTs”. The “IncidentNumber”, “IncidentStatus”, “ActiveStatus”, “MadeSLA”, “Caller”, “OpenedBy”, “UpdatedBy”, and “ResolvedBy” columns will be used to derive “Dimensions”. These tables will be used together to form an analytical schema design. This document will illustrate the schema and describe its proposed design. The *2016 UCI Machine Learning Repository Event Log* data set will be extracted, transformed, and loaded (ETL) into the analytical schema design using a custom-built SQL Server Integration Services (SSIS) package developed with SQL Server Data Tools (SSDT) in Visual Studio. Additionally, analysis will be performed on the 2016 UCI Machine Learning Repository event log by creating an OLAP Cube using SQL Server Analysis Services. Data will be processed from the Data Mart tables and loaded into Analysis Services to populate the cube. Finally, we will use SQL Server Analysis Services to data mine the OLAP cube using visualization tools such as Tableau and Power BI.

Data Mart Schema Diagram:



Data Mart Meta Data:

Incident_FactEvent			
Column Name	**Data Type**	**Description**	**Key**
EventID	Int	Primary Key	Primary
IncidentID	Int	Unique Identifier for each Incident. Multiple events per incident. Foreign Key column to DimNumber	Foreign
CreatedID	Int	Unique identifier for each user that created an incident. Foreign key column to DimCreated	Foreign
UpdatedID	int	Unique Identifier for each user that made an update. Foreign Key column to DimUpdated	Foreign
OpenedID	Int	Unique Identifier for each user that opened an incident. Foreign Key column to DimOpened	Foreign
ResolvedID	Int	Unique Identifier for each user that resolved an incident. Foreign Key column to DimResolution	Foreign
StatusID	Int	Unique Identifier for each type of incident status. Foreign Key column to DimStatus	Foreign
CallerID	Int	Unique Identifier for each caller. Foreign Key column to DimCaller	Foreign
MadeSLAID	Int	Unique Identifier for each Made SLA status. Foreign Key column to DimMadeSLA	Foreign
Reassignment_Count	Int	Counts the number of times an incident got reassigned	None
Reopen_Count	Int	Counts the number of times an incident got reopened	None
SysMod_Count	Int	Counts the number of times an incident got updated	None
OpenedAt	Varchar(20)	Date and time the incident was opened	None
UpdatedAt	Varchar(20)	Date and time the incident was updated	None
ResolvedAt	Varchar(20)	Date and time the incident was resolved	None

DimNumber			
Column Name	**Data Type**	**Description**	**Key**
IncidentID	Int	Primary Key	Primary
Incident_Number	varchar(10)	Number of the incident	None

DimCreated			
Column Name	**Data Type**	**Description**	**Key**
CreatedID	Int	Primary Key	Primary
Created_By	Varchar(25)	Name of user who created` the incident	None
Created_At	Varchar(20)	Date and time the incident was created	None

DimStatus			
Column Name	**Data Type**	**Description**	**Key**
IncidentStatusID	Int	Primary Key	Primary
IncidentStatus	Varchar(20)	Status of the incident	None
ActiveStatus	Varchar(5)	True or False if the incident is active	None

DimMadeSLA			
Column Name	**Data Type**	**Description**	**Key**
MadeSLAID	Int	Primary Key	Primary
MadeSLA	Varchar(5)	True or False if the incident followed SLA guidelines	None

DimCaller			
Column Name	**Data Type**	**Description**	**Key**
CallerID	Int	Primary Key	Primary
Caller	Varchar(11)	Name of caller	None

DimOpened			
Column Name	**Data Type**	**Description**	**Key**
OpenedID	Int	Primary Key	Primary
OpenedBy	Varchar(25)	Name of user who opened the incident	None

DimUpdates			
Column Name	**Data Type**	**Description**	**Key**
Updated_ID	Int	Primary Key	Primary
Updated_By	Varchar(15)	Name of user who updated an incident	None

DimResolution			
Column Name	**Data Type**	**Description**	**Key**
ResolvedID	Int	Primary Key	Primary
ResolvedBy	varchar(20)	Name of user who resolved an incident	None

Data Mart ETL Description:

The ETL process has been designed with SQL Server Data Tools (SSDT) using the SQL Server Integration Services (SSIS) Template in Visual Studio. The advantage of having a custom designed SSIS package available to the Analyst offers a strategic advantage in the event that any rapid re-design changes required to the overall analytical design arise. First, design changes can be implemented to the table structure. Then, the SSIS package would be modified appropriately so that the data can be easily truncated and reloaded into all Schemas.

SSIS Package Details:

CONTROL FLOW NAME	TYPE OF TASK	DATA FLOW NAME	DESCRIPTION
Get Data from CSV to IncidentStaging Table	Data Flow Task	STEP1-Get Data from CSV	Connects to the .csv source data file and extracts the data
Get Data from CSV to IncidentStaging Table	Data Flow Task	STEP2-Data Conversion	Transforms all columns from .csv file to correct format for staging table
Get Data from CSV to IncidentStaging Table	Data Flow Task	STEP3-LoadDataIntoDataMart	Loads transformed data into UCIIncidentStaging table
Load DimNumber	Data Flow Task	STEP1-Get IncidentNumber from Staging Table	Selects distinct incident numbers from staging table
Load DimNumber	Data Flow Task	STEP2-Load into DimNumber	Loads all distinct incident numbers into DimNumber
Update IncidentID into Staging Table	Execute SQL Task		Updates Primary Key column from DimNumber into Staging Table
Load DimCreated	Data Flow Task	STEP1-Get Created_By and Created_At from Staging Table	Selects distinct Created_by and Created_At combinations from staging table
Load DimCreated	Data Flow Task	STEP2-Load into DimCreated	Loads all distinct Created_By and Created_At combinations into DimCreated
Update CreatedID into Staging Table	Execute SQL Task		Update Primary Key column from DimCreated into staging table

Load DimUpdates	Data Flow Task	STEP1-Get Updated_By from Staging Table	Selects distinct updated by users from staging table
Load DimUpdates	Data Flow Task	STEP2-Load into DimUpdates	Loads all distinct updated by users into DimUpdates
Update Updated_ID into Staging Table	Execute SQL Task		Updates Primary Key column from DimUpdates into Staging Table
Load DimOpened	Data Flow Task	STEP1-Get OpenedBy from Staging Table	Selects distinct opened by users from staging table
Load DimOpened	Data Flow Task	STEP2-Load into DimOpened	Loads all distinct opened by users into DimOpened
Update OpenedID into Staging Table	Execute SQL Task		Updated Primary Key column from DimOpened into Staging Table
Load DimResolution	Data Flow Task	STEP1-Get ResolvedBy from Staging Table	Selects distinct resolved by users from staging table
Load DimResolution	Data Flow Task	STEP2-Load into DimResolution	Loads all distinct resolved by users into DimResolution
Update ResolvedID into Staging Table	Execute SQL Task		Update Primary Key column from DimResolution into Staging Table
Load DimStatus	Data Flow Task	STEP1-Get IncidentStatus and ActiveStatus from Staging Table	Selects distinct IncidentStatus and ActiveStatus combinations from staging table
Load DimStatus	Data Flow Task	STEP2-Load into DimStatus	Loads all distinct IncidentStatus and ActiveStatus combinations into DimStatus
Update StatusID into Staging Table	Execute SQL Task		Update Primary Key column from DimStatus into staging table
Load DimCaller	Data Flow Task	STEP1-Get Caller from Staging Table	Selects distinct callers from staging table
Load DimCaller	Data Flow Task	STEP2-Load into DimCaller	Loads all distinct callers into DimCaller
Update CallerID into staging table	Execute SQL Task		Update Primary Key column from DimCaller into staging table
Load DimMadeSLA	Data Flow Task	STEP1-Get MadeSLA from Staging Table	Selects distinct MadeSLA from Staging Table
Load DimMadeSLA	Data Flow Task	STEP2-Load into DimMadeSLA	Loads distinct MadeSLA into DimMadeSLA
Update MadeSLAID into Staging Table	Execute SQL Task		Update Primary Key column from DimMadeSLA into staging table

Load FactUCIIncident	Data Flow Task	STEP1-Get Staging Table Data	Gets data from the staging table
Load FactUCIIncident	Data Flow Task	STEP2-Load into FactUCIIncident	Loads data from staging table into FactUCIIncident