# ServiceNow Database

ISM 4402 Business Intelligence

## Brief Description:

The goal of this document is to lay down a template to interpret the ServiceNow database and the data gathered. In this document we have included a data mart design with the data structures, schema diagram, data mart metadata, and the ETL process we followed.


**Team Members:**

Justin Hartman

Bethy Gomez

Nahisha Nobregas

Mark Shim

# Data Mart Design Description:

The Data Mart we were instructed to create was for Incident Ticket data from the IT Service management software known as ServiceNow. The original flat file CSV data structure is from the UCI Machine Learning Repository and was donated on 07/14/2019 by an IT company. We used the relational database program known as Microsoft SQL Server to build our Data Mart and run analytics. The incident_event_log.csv that we downloaded originally carried 36 attributes and had 141,712 events and 24,918 incidents recorded. Out of the 36 attributes we selected 12 to focus on for our database.

**The Facts of our database are as follows:**

ReassignCount

ReopenCount

SysModCount

SysCreatedBy

SysUpdatedBy

ResolvedAt

SourceRowID (Incident Number)

**The Dimensions of our database are as follows:**

IncState

SysCreatedAt

Priority

CloseCode

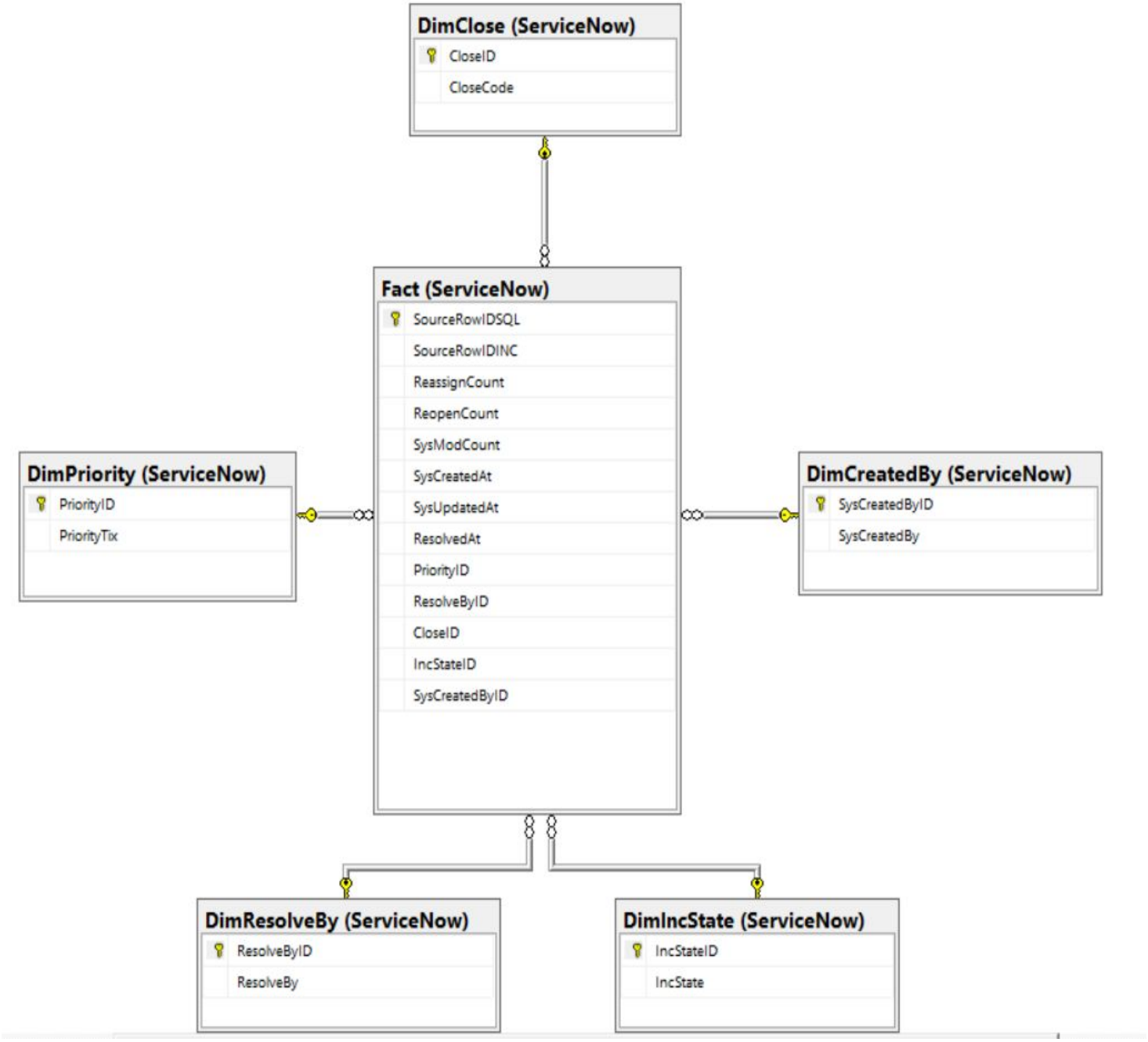
ResolveBy

The 12 attributes that we pulled were then used in a Schema for analytical purposes in SQL Server. After we created the table within the schema, we used Visual Studio SSDT along with SSIS to extract the Flat File CSV data, transform the data all to String/four-bit-unsigned data types and then load the data into the ServiceNow database that we created in SQL Server. Once the data was loaded into our Stage Table, a new table was created titled ServiceNow.Transformed. This table would allow for the necessary transformations of Int and DateTime data types during the ETL process. After using Visual Studio to perform the ETL process again, data is now loaded into both the ServiceNow.StageTable & the ServiceNow.Transformed table. Finally, the dimension tables are created and populated, along with the creation of the Fact table. The Fact table is then linked to the dimension tables through FK's, which are the PK's of the dimension tables. Lastly, the ETL process is run again, populating the Stage table, Transformation table, dimension tables, and the fact table.

| SourceI... | SourceRowIDINC | ReassignCo... | ReopenCou... | SysMod... | SysCreatedAt_Transformed | SysUpdatedAt_Transformed | ResolvedAt_Transformed | PriorityTix | Priori... | IncState |
|---|---|---|---|---|---|---|---|---|---|---|
| 101 | INC0000045 | 0 | 0 | 0 | 2016-02-29 01:23:00.000 | 2016-02-29 01:23:00.000 | 2016-02-29 11:29:00.000 | 3 - Mode... | 103 | New |
| 102 | INC0000045 | 0 | 0 | 2 | 2016-02-29 01:23:00.000 | 2016-02-29 08:53:00.000 | 2016-02-29 11:29:00.000 | 3 - Mode... | 103 | Resolved |
| 103 | INC0000045 | 0 | 0 | 3 | 2016-02-29 01:23:00.000 | 2016-02-29 11:29:00.000 | 2016-02-29 11:29:00.000 | 3 - Mode... | 103 | Resolved |
| 104 | INC0000045 | 0 | 0 | 4 | 2016-02-29 01:23:00.000 | 2016-03-05 12:00:00.000 | 2016-02-29 11:29:00.000 | 3 - Mode... | 103 | Closed |

Note: not all data is included in this view

# Data Mart Schema Diagram:



# Data Mart Meta Data:

| FACT TABLE | | | |
|---|---|---|---|
| **Column Name** | **Data Type** | **Description** | **Key** |
| **SourceRowIDSQL** | int | PrimaryKey | Primary |
| **SourceRowIDINC (Incident Number)** | int | Incident Number | None |
| **ReassignmentCount** | int | Number of times the incident has the group or support analysts changed. | None |
| **ReopenCount** | int | Number of times the incident resolution was rejected by the caller. | None |
| **SysModCount** | int | Number of incident updates until that moment. | None |
| **SysCreatedAt** | datetime | Incident system creation date and time. | None |
| **SysUpdatedAt** | datetime | Incident system update date and time. | None |
| **ResolvedAt** | datetime | Incident user resolution date and time. | None |

| | | | |
|---|---|---|---|
| **CloseID** | int | Foreign Key for Close Dimension | Foreign |
| **PriorityID** | int | Foreign Key for Priority Dimension | Foreign |
| **SysCreatedByID** | int | Foreign Key for SysCreatedBy Dimension | Foreign |
| **ResolvedByID** | int | Foreign Key for ResolvedBy Dimension | Foreign |
| **IncStateID** | int | Foreign Key for IncidentState Dimension | Foreign |

| **DIMENSION TABLE CLOSE** | | | |
|---|---|---|---|
| **Column Name** | **Data Type** | **Description** | **Key** |
| **CloseID** | int | A unique identifier | Primary |
| **CloseCode** | varchar(150) | Identifier of the resolution of the incident. | None |

| DIMENSION TABLE PRIORITY | | | |
|---|---|---|---|
| **Column Name** | **Data Type** | **Description** | **Key** |
| **PriorityID** | int | A unique identifier | Primary |
| **PriorityTix** | varchar(150) | Calculated by the system based on impact and urgency. | None |

| DIMENSION TABLE INCIDENT STATE | | | |
|---|---|---|---|
| **Column Name** | **Data Type** | **Description** | **Key** |
| **IncStateID** | int | A unique identifier | Primary |
| **IncState** | varchar(150) | Eight levels controlling the incident management process from opening until closing. | None |

| DIMENSION TABLE SYSCREATEDBY |
|---|

| Column Name | Data Type | Description | Key |
|---|---|---|---|
| **SysCreatedByID** | int | A unique identifier | Primary |
| **SysCreatedBy** | varchar(150) | Identifier of user who registered the incident. | None |

| DIMENSION TABLE RESOLVEDBY | | | |
|---|---|---|---|
| **Column Name** | **Data Type** | **Description** | **Key** |
| **ResolveByID** | int | A unique identifier | Primary |
| **ResolveBy** | varchar(150) | Identifier of the user who resolved the incident. | None |

# DATA MART ETL DESCRIPTION:

The ETL process was designed using the SQL Server Data Tools (SSDT) using the SQL Server Integration Services (SSIS) Template in Visual Studio 2017. We began the process by selecting the facts and dimensions we wanted to work with from the CSV Flat File we got from the UCI Machine Learning Repository. We then separated the data using a comma delimiter, this allowed for the data to be placed into the rows cleanly. A staging table including these facts and dimensions was created in SQL Server. This table included new columns that we created for particular dimension columns that will later serve as foreign keys in our fact table, all the columns were set to Varchar(150). Then, we transformed the data types of those facts and dimensions that were chosen to String (150) and four-byte unsigned int for the transformation process. We then mapped the columns of the CSV to the SQL columns that we wanted the data to be associated with. The data was then loaded onto our data mart staging table using OLE DB Source. The next step was to get our data from the staging table to the newly created Transformed table. This was done through a data flow task which used a SQL command to select, cast, update, and convert the data of the Stage Table to the destination Transformed table. From here five more Data Flow tasks were added and coupled with SQL tasks to populate the Dimension tables and to update the Transformed table. At the very end of the control flow, another Data Flow Task is added that loads the selected columns of the Transformed table to the Fact Table. Having this SSIS package available allows the database to be quickly remade in the event that any design changes are required or the database has to be recreated. Below is a more in depth view of our SSIS ETL process.

# SSIS Package Details:

| CONTROL FLOW NAME | TYPE OF TASK | DATA FLOW NAME | DESCRIPTION |
|---|---|---|---|
| **ETL from CSV to Staging Table** | Data Flow Task | STEP1- Connect to CSV File | Connects to the .csv source data file and extracts the data |
| **ETL from CSV to Staging Table** | Data Flow Task | STEP2- Data Conversion | Conversion of Data Type and Length to populate Staging Table |
| **ETL from CSV to Staging Table** | Data Flow Task | STEP3- Load Source_Data_Staging | Loads data into Source_Data_Staging table |
| **Transform Staging Table** | Data Flow Task | STEP1- Source_Data_Staging Connection | Executes SQL Command to take data from Staging Table and convert data types |
| **Transform Staging Table** | Data Flow Task | STEP2- Load Source_Data_Staging_ Transformed | Loads converted data into Transformed Staging Table |
| **Load DimClose** | Data Flow Task | STEP1- Connect to Transformed Staging Table | Executes SQL command to select distinct CloseCode values from Staging Table |
| **Load DimClose** | Data Flow Task | STEP2- Load DimClose Table | Loads DimClose Table with CloseCode column |
| **Update CloseID into Staging Table** | Execute SQL Task | | Executes UPDATE query to assign the CloseID to Transformed Staging Table |
| **Load DimIncState** | Data Flow Task | STEP1- Connect to Transformed Staging Table | Executes SQL command to select distinct IncState values from Staging Table |
| **Load DimIncState** | Data Flow Task | STEP2- Load DimIncState Table | Loads DimIncState table with IncState column |
| **Update IncStateID into Staging Table** | Execute SQL Task | | Executes UPDATE query to assign the IncStateID to Transformed Staging Table |
| **Load DimPriority** | Data Flow Task | STEP1- Connect to Transformed Staging Table | Executes SQL command to select distinct Priority values from Staging Table |

| | | | |
|---|---|---|---|
| **Load DimPriority** | Data Flow Task | STEP2- Load DimPriority Table | Loads DimPriority table with Priority column |
| **Update PriorityID into Staging Table** | Execute SQL Task | | Executes UPDATE query to assign the PriorityID to Transformed Staging Table |
| **Load DimResolved** | Data Flow Task | STEP1- Connect to Transformed Staging Table | Executes SQL command to select distinct ResolveBy values from Staging Table |
| **Load DimResolved** | Data Flow Task | STEP2- Load DimResolved Table | Loads DimReoslved table with ResolveBy column |
| **Update ResolveID into Staging Table** | Execute SQL Task | | Executes UPDATE query to assign the ResolveByID to Transformed Staging Table |
| **Load DimSysCreatedBy** | Data Flow Task | STEP1- Connect to Transformed Staging Table | Executes SQL command to select distinct SysCreatedBy values from Staging Table |
| **Load DimSysCreatedBy** | Data Flow Task | STEP2- Load DimSysCreatedBy Table | Loads DimSysCreatedBy table with SysCreatedBy column |
| **Update SysCreatedByID into Staging Table** | Execute SQL Task | | Executes UPDATE query to assign the SysCreatedByID to Transformed Staging Table |
| **Load Fact Table** | Data Flow Task | STEP1- Connect to Transformed Staging Table | Connects to the Transformed Staging Table and extracts the facts and ID columns |
| **Load Fact Table** | Data Flow Task | STEP2- Load Fact Table | Loads the Fact table with facts and Foreign Key columns |

**ETL Process in SSIS:**