Sri Lanka Institute of Information Technology

Data Warehousing and Business Intelligence

Assignment 01

Submitted by: - IT20268244

Kaleem F.A

Table of Contents

Declaration

I declare that this project report or part of it was not a copy of document done by any organization, university and other institute or a previous student project at SLIIT and was not copied from the internet or other resources.

Created by: -

IT number: IT20268244

Name: Amani Kaleem

Batch: Y3S1.05.02

Data Set Selection

Data Set Name**: Australia & New Zealand Road Crash Dataset**

Provided by: **kaggle.com**

This is a dataset that contains data about on when, where and under what conditions car accidents occur in Australia and New Zealand. It records every information related to each accident which includes the vehicle type that got accident, the driver details who drove the vehicle, the location details including detailed weather condition of the specified location. Moreover, it also describes details such as the road condition, lightning in the road and what type of crash occurred. Finally, it records the most important detail which is the number of casualties, fatalities and injuries that occurred due to the accident.

The original dataset has 6 csv files including more than 1 million records. Using this data set we could analyze on what basis most of the accidents occur in order to minimize the number of accidents in future in Australia and New Zealand.

Preparation of data sources

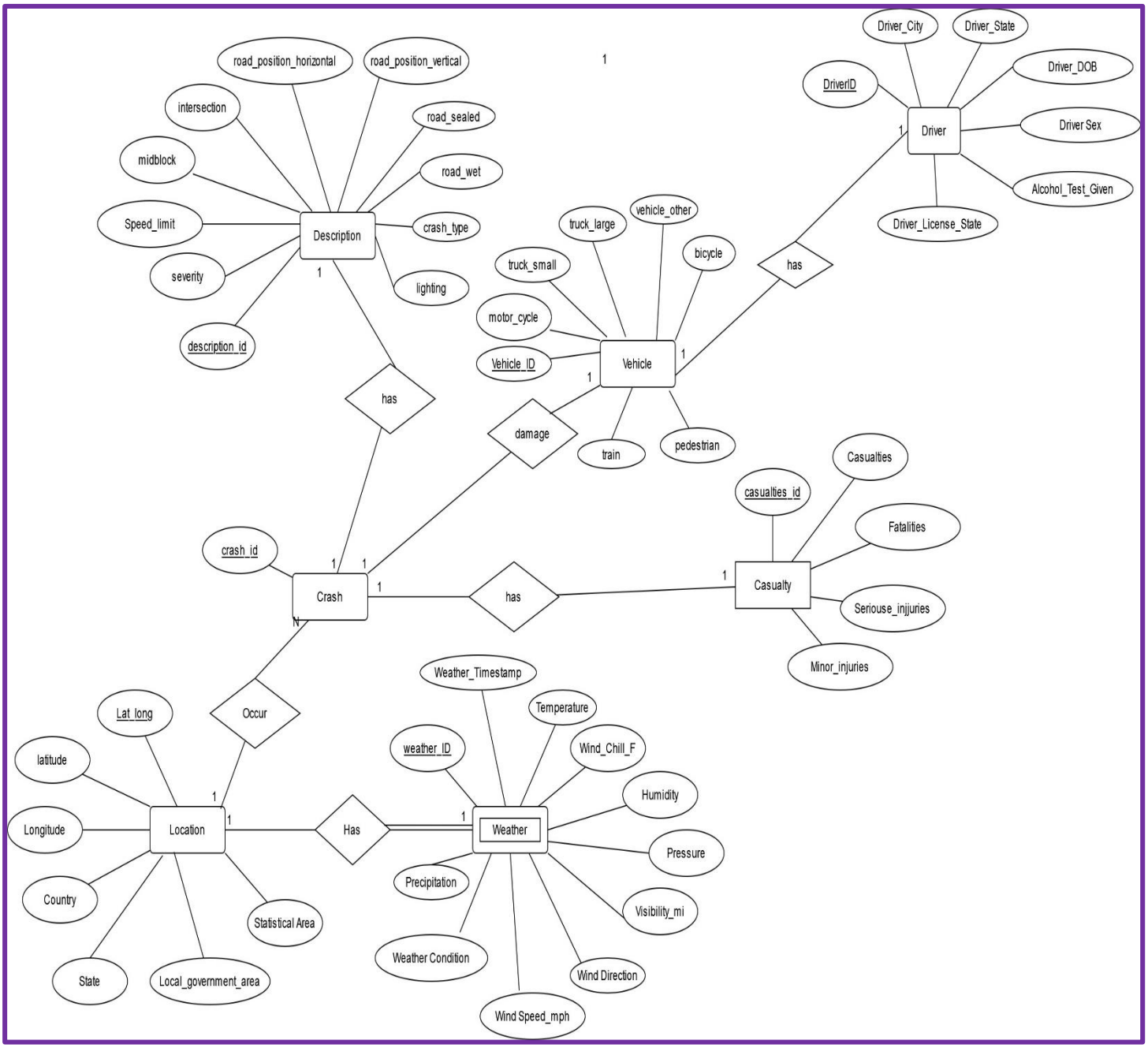
The Entire dataset content was given using different CSV format files. Thereby in preparing data source, some changes were t complete data source along with the project criteria. There by I added few additional source files and connected with my dataset accordingly. Finally, some changes was done to the format of the source data where I converted some CSV files to text and source database.

Final source data used for transformation process are as follows: -

- Road_Crash_SourceDB
  - dbo.Vhicle
  - dbo.Driver
  - dbo.Location
  - dbo.Crash
  - dbo.Casulty
- Description.txt – Description of the accident.
- Weather.txt – weather of the location the accident occurred.

  - According to the finalized dataset the data related to the vehicle that got accident is recorded in dbo.Vhicle . The information related to the driver who drove the vehicle while the accident occurred is saved in the dbo.Driver where driver is related with the vehicle table. The information related to the location the accident took place is recorded in dbo.Location while it is connected with Weather.txt which includes data related to weather condition while the accident took place. The dbo.Casulty contains data related to number of casualties, fatalities and injuries that took place during the incident. Description.txt is the source that is used to record all other details related to the accident. Finally the dbo.Crash is the source table that related to location, vehicle, description and date tables in the final data set.

Entity Relationship Diagram



Data Description

Source Type: Road_Crash_SourceDB

Table Name: Location

Column name	Datatype	Description
Lat_long	Varchar(90)	Unique Primary key
latitude	varchar(50)	Latitude of the location
Longitude	varchar(50)	Geographical longitude of the location
Country	varchar(50)	Country of the location
State	varchar(50)	The state location of accident belongs to.
Local_government_area	varchar(100)	The local government area the location belongs to.
Statistical Area	varchar(50)	The Statistical area of the country
Weather ID	int	References the weather table

Table Name: Casualties

Column name	Datatype	Description
casualties_id	Varchar (30)	Unique primary key
Casualties	Int	Number of casualty(deaths) during the accident
Fatalities	Int	Number of fatalities during the accident
Seriouse_injjuries	int	Number of serious injuries during the accident
Minor_injuries	int	Number of minor injuries during the accident

Table Name: Vehicle

Column name	Datatype	Description
Vehicle_ID	varchar(30)	Unique primary key
motor_cycle	int	Is the vehicle a moto cycle
truck_small	Int	Is the vehicle a small truck
truck_large	Int	Is the vehicle a larg truck
bicycle	Int	Is the vehicle a bicycle
pedestrian	Int	Is the vehicle a pedestrian
train	Int	Is the vehicle a train
vehicle_other	int	If vehicle is not type of mentioned above
Driver_id	int	References to the driver table

Table Name: Driver

Column name	Datatype	Description
id	Int	Unique primary key
Driver_City	varchar(30)	The vehicle driver's city
Driver_State	varchar(20)	Driver home state
Driver_License_State	varchar(10)	Check if the driver has a valid license
Driver_DOB	date	Driver date of birth
Driver Sex	varchar(1)	Driver sex as male or female
Alcohol_Test_Given	bit	Alchohol test results to check if driver was drunk

Table Name: Crash

Column name	Datatype	Description
crash_id	varchar(50)	Unique primary key
lat_long	varchar(90)	References location table
date_time_id	varchar(50)	References date table
Vehicle_ID	varchar(30)	References Vehicle table
casualties_id	varchar(30)	References Casualty table

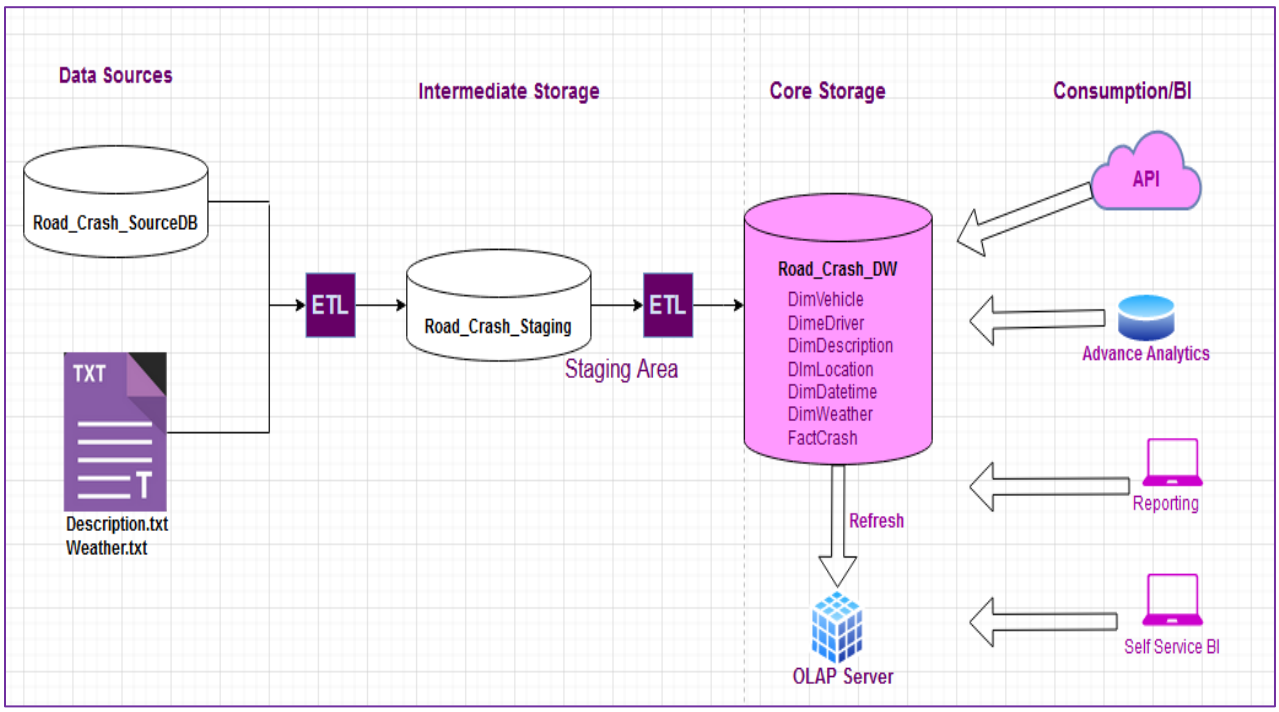
Weather.txt — Weather

Column name	Datatype	Description
id	int	Unique primary key
Weather_Timestamp	datetime2(7)	The date and time of the weather recorded at time of accident
Temperature	varchar(30)	Temperature at the time of accident
Wind_Chill_F	varchar(30)	Wind chill in faranhite at time of accident
Humidity	varchar(30)	Humidity of the weather condition
Pressure	float	Pressure of the weather
Visibility_mi	float	Visibility of the weather
Wind Direction	(varchar(40)	Wind direction at time of accident
Wind Speed_mph	varchar(30)	Wind speed at time of accident
Precipitation	varchar(30)	Precipitation of the weather
Weather Condition	(varchar(40)	Final weather condition

Description.txt — Description

Column name	Datatype	Description
description_id	int	Unique description ID primary key
severity	varchar(50)	Severity of the accident
Speed_limit	int	Speed of the accident
midblock	Bit	Was the accident in a midblock
intersection	Bit	Was the accident in a intersecting point
road_position_horizontal	varchar(50)	The horizontal road position
road_position_vertical	varchar(50)	The verticle road position of the accident
road_sealed	varchar(40)	Check if the road was sealed
road_wet	bit	Check if the road was wet during the accident
crash_type	varchar(30)	The type of road crash
lighting	varchar(40)	The road lightning condition while the accident occurred.

High Level Design Architecture



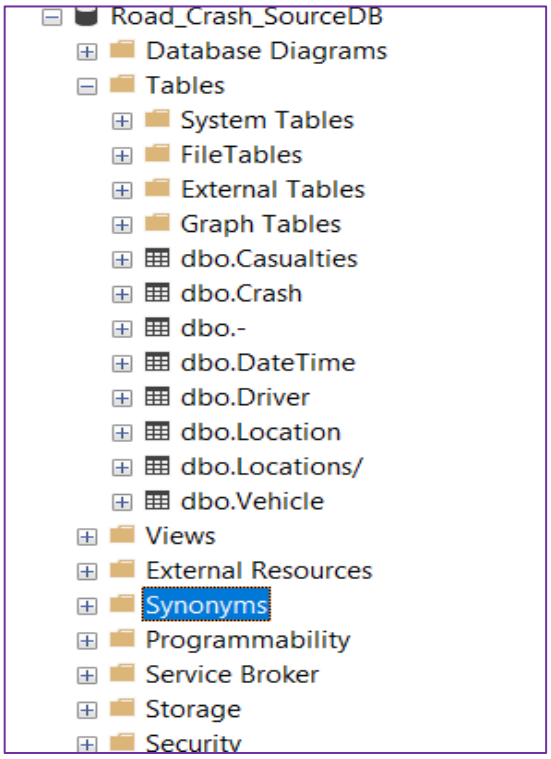
The high-level architecture solution of the Australia & New Zealand Road Crash is as given above. The entire dataset is provided by two separate source formats: Source database and Text respectively.

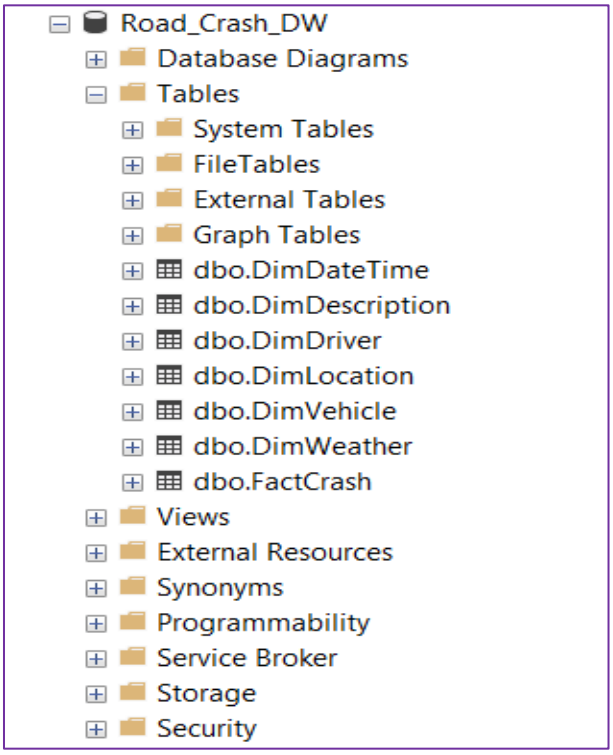
As the first step the entire source data Road_crash_SourceDB , Description.txt and Weather.txt is loaded to the staging database named Road_Crash_Staging. This database ultimately serves as a single database including all source data as one database.

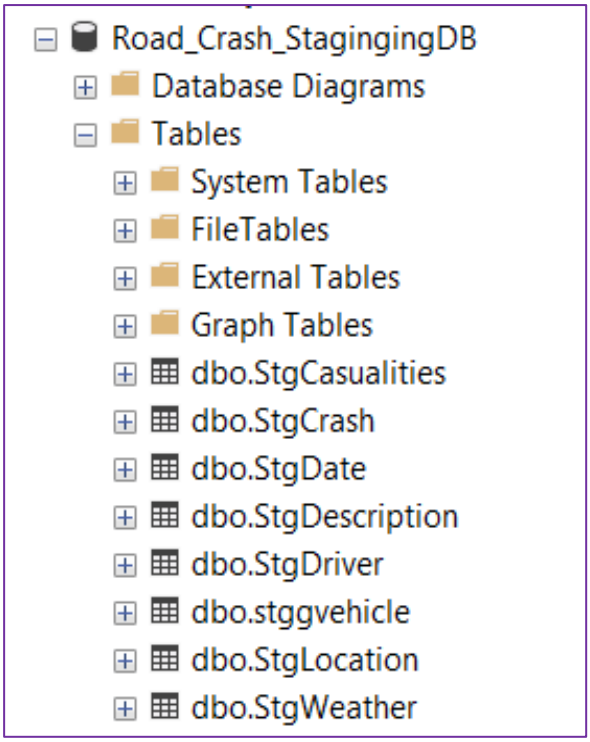
AS the next step once the data is loaded to the staging area the data is cleaned, validated and then necessary transformations are done in order to load data from the staging area to the datawharehouse(Road_Crash_DW).

The data from data warehouse is then refreshed to create OLAP cubes which can be used by end users in order to carry out Analysis on the data set. Here data visualization can be either done through OLAP server or directly from Data warehouse which is specifically called Self-service BI.

Data Storage Snapshots from SSMS

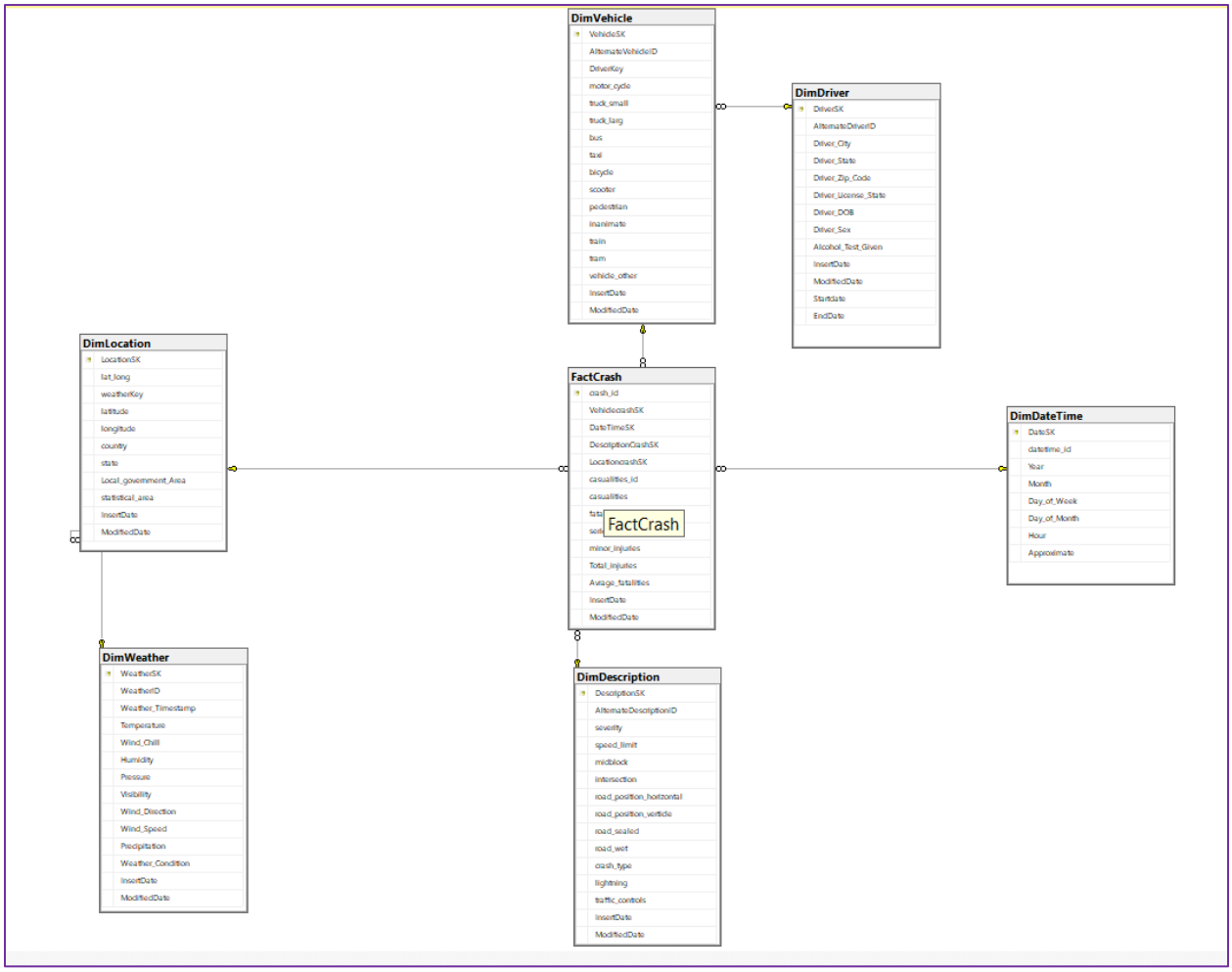






Data warehouse design & development

Relational Schema



The Schema designed for the Australia & New Zealand Road Crash Dataset is a snowflakes schema with six-dimension tables and one Road_Crash fact table. Thereby the entities are normalized.

The dimensions are uniquely identified by the Surrogate key, where additionally each dimension contains the business key provided via the source data base.

Hierarchical implementations are found in this schema

1. DimDriver is the hierarchical dimension for DimVehicle

2. DimLocation has DimWeather as a Hierarchy.

DimDriver table is identified as a slowly changing dimension with historical attributes, changing attributes and Fixed attributes where Type 2 ,Type 1 and Type0 implementations are being enforced, respectively.

Crash in a particular date is considered as the grain of the Fact_Transaction fact table.

Additional derived Calculations are done in fact table,

a. SumOfInjuries = Minor_injuries+ Major_Injuries
b. Average_Casualty = (casualties+ Fatalities)/2

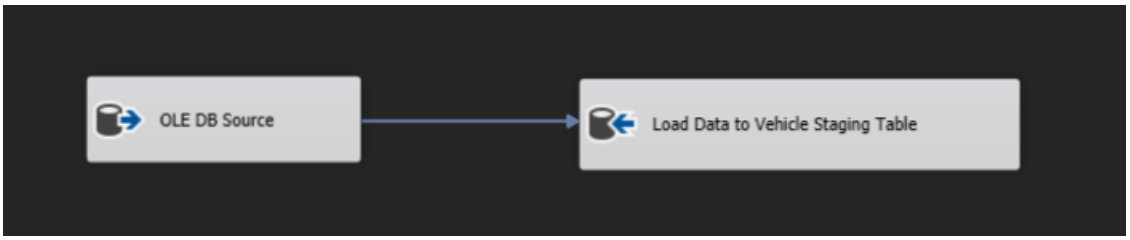
ETL Development

Data extraction from sources to staging area

Data extraction from source data to staging area is done using SQL Server Data tool 2018 environment. The Source Database and Text files created are used to load data to the staging databases. To extract data from Source Database tables, OLE DB Source is used in the data flow whereas text file extraction is done using Flat file source. From Source database to staging database lesser transformations are done, primary concern was on Data extraction and loading data to staging tables for further processing. The extracted Source data are finally loaded into the respective staging tables using OLE DB Destination.

Thereby following are some snapshots of different source data extractions to the staging database.

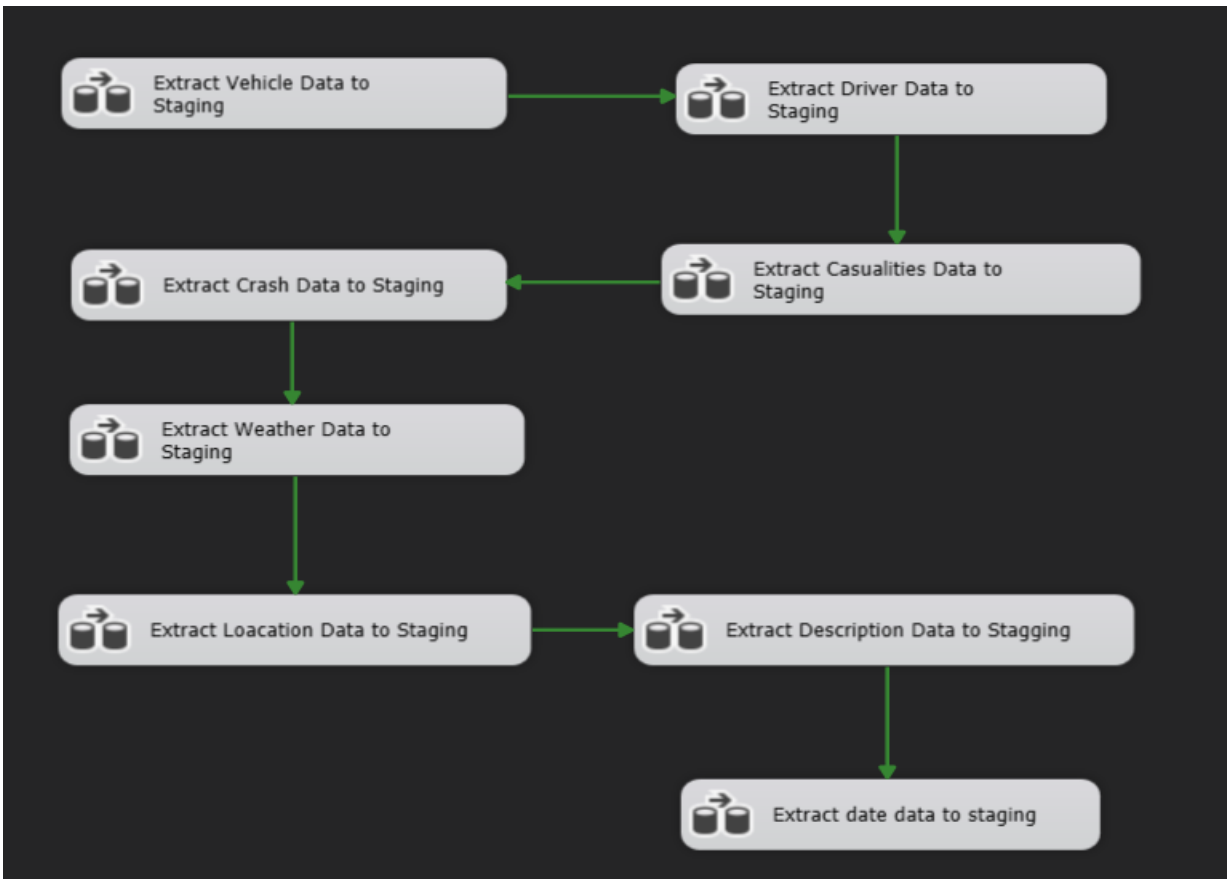
Loading Database tables to staging:



The above process was continued to load data from database tables to staging area.

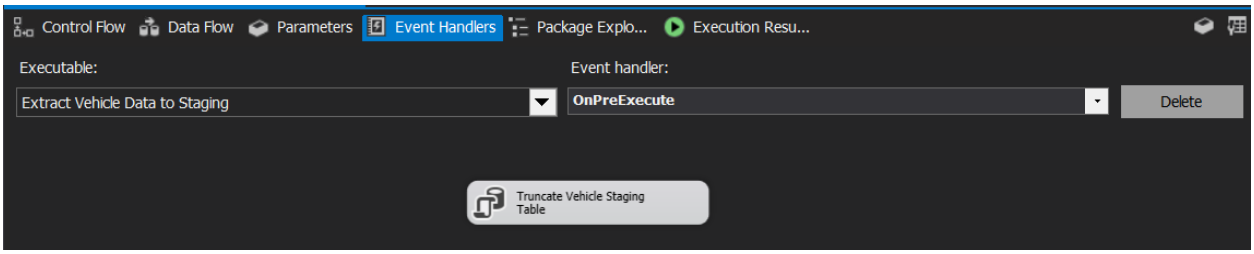
Loading data from flat file to staging database

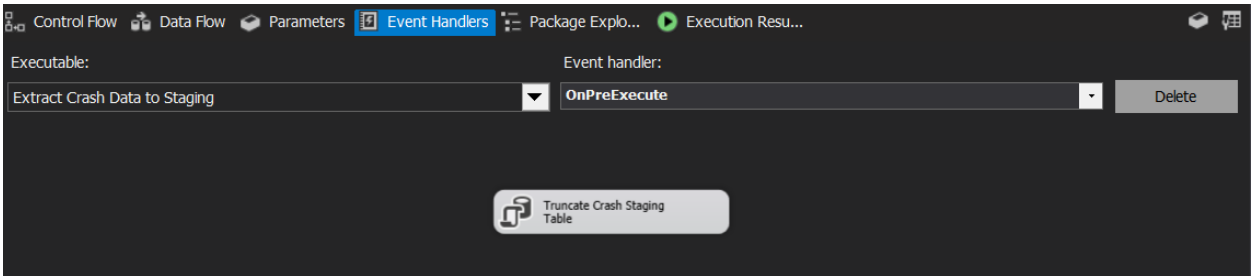
The overall process of loading data from data sources to staging database was done using the following execution order.



Moreover, to prevent data duplication while staging tables are repeatedly loaded, ON PRE EXECUTE event handlers are used to truncate data prior to loading each table from sources to staging database.

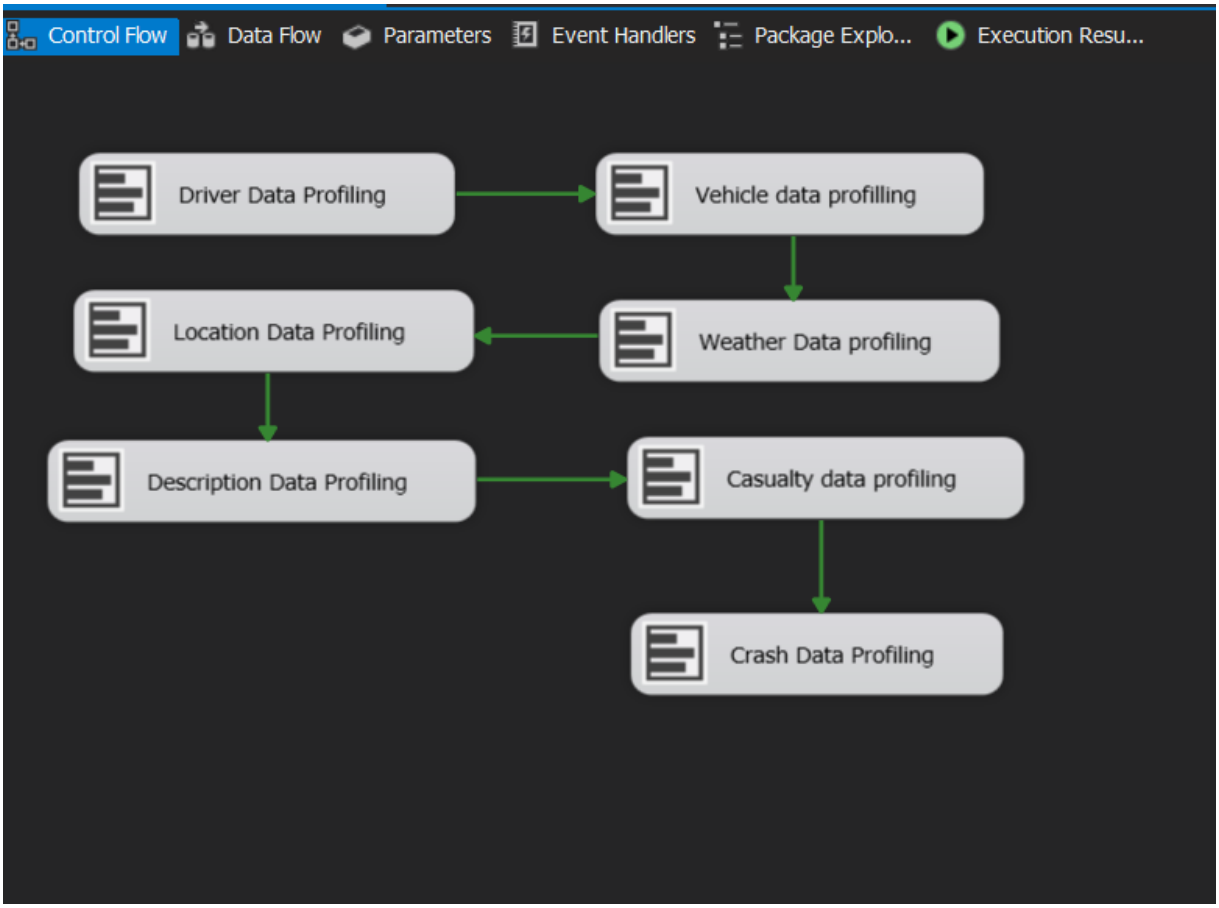
Following are snapshot of truncating the tables before loading (Only some snapshots are included and the same process has been followed for all the tables prior to loading)





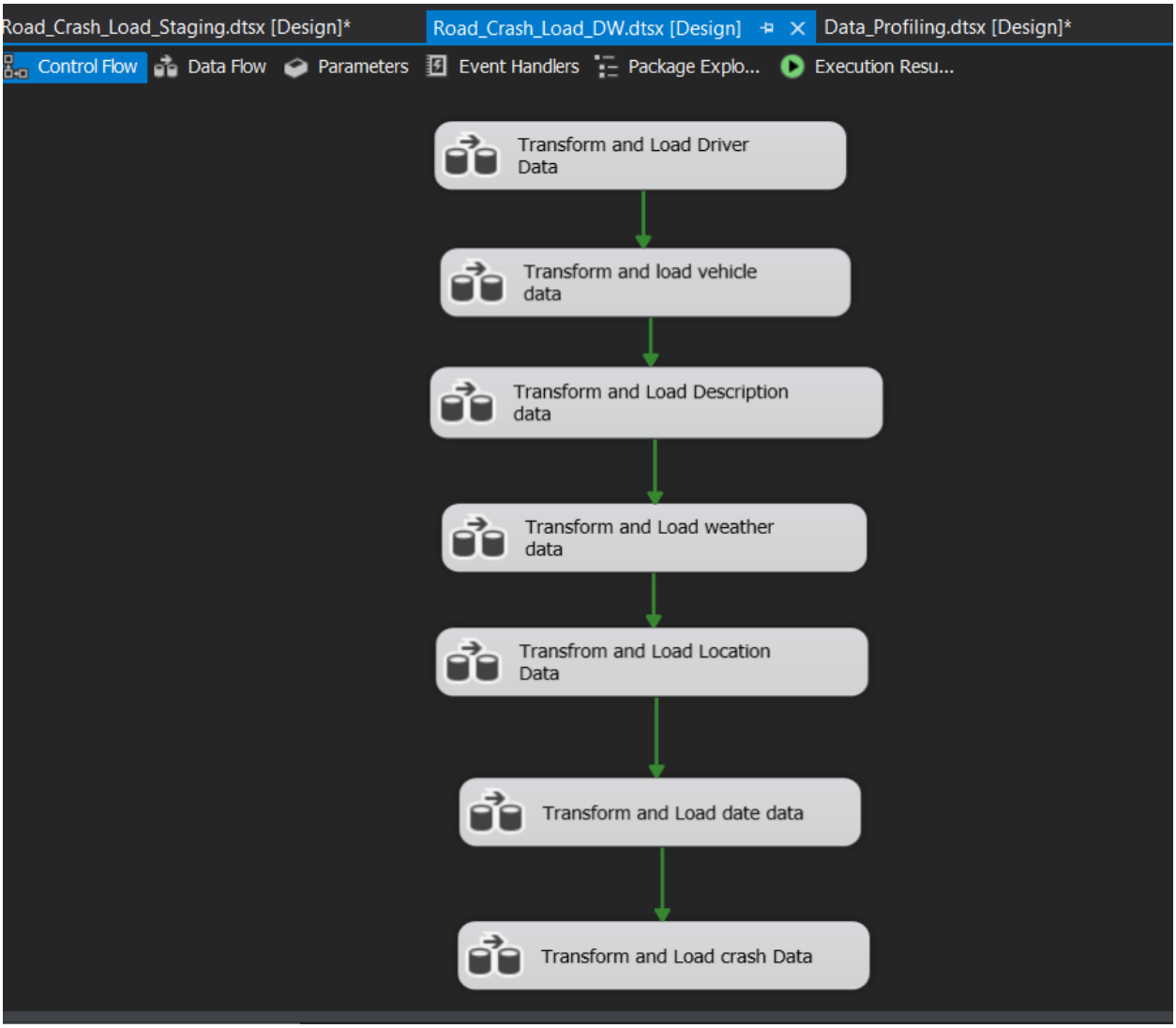
Data Profiling

Once the data was loaded in to the staging tables, data profiling was done in order to analyze how the data looks like to determine the type of transformations needed to perform on each data set and by profiling null value ratios, column length distributions and various other statistical information was able to be determined.



Data Extraction and Datawarehouse Loading

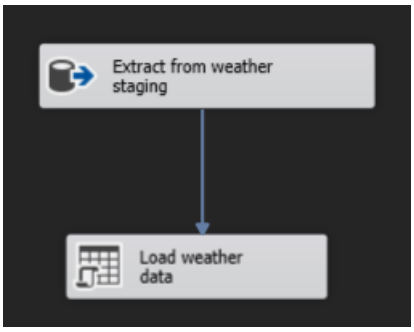
During the process of loading tables from the staging table to Datawarehouse the main concern was the order of execution by analyzing what are the dependencies and when they should be loaded. Thereby following shows the execution order of the data warehouse loading,

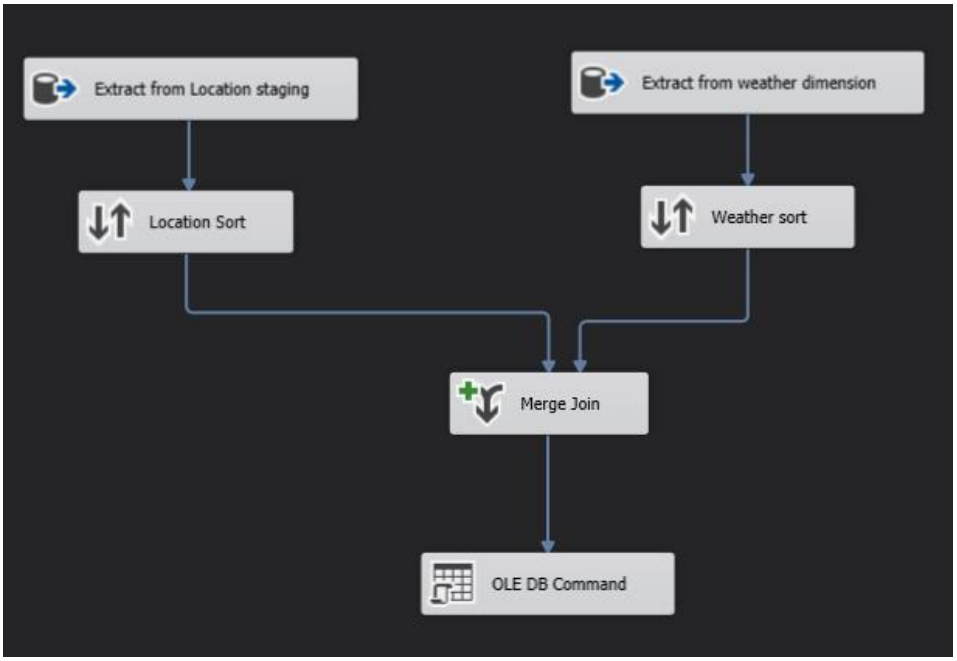


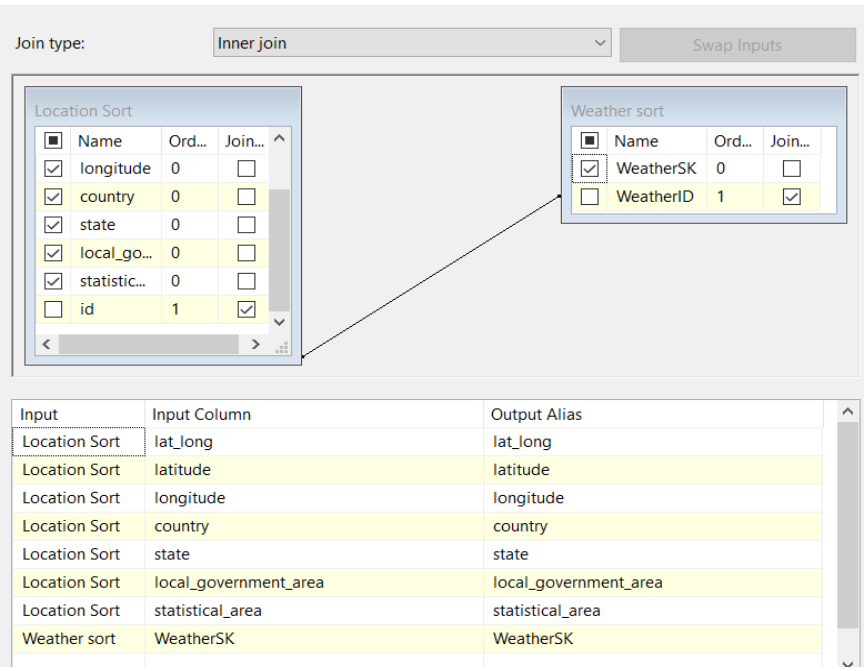
Loading Hierarchical dimensions

According to the relational schema it clearly depicts that there are two hierarchical dimensions which the first one is a driver and vehicle and the second is weather and location.

Accordingly, the Location and weather hierarchy is displayed below,







Input	Input Column	Output Alias
Location Sort	lat_long	lat_long
Location Sort	latitude	latitude
Location Sort	longitude	longitude
Location Sort	country	country
Location Sort	state	state
Location Sort	local_government_area	local_government_area
Location Sort	statistical_area	statistical_area
Weather sort	WeatherSK	WeatherSK

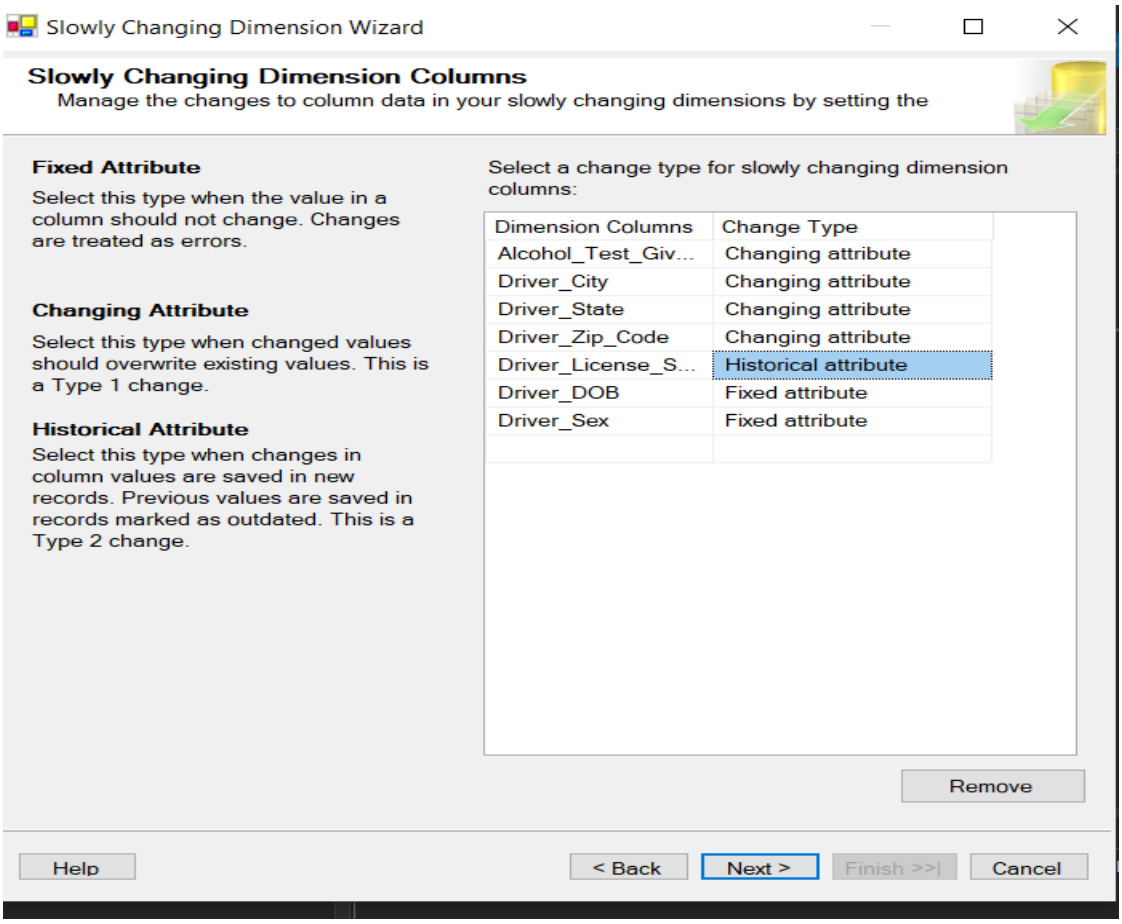
While loading location the location table was sorted using weather_id and Weather dimension table was also sorted using weather_id. Finally, both were merged using a merge join and then loaded to DimLocation.

(Similarly, the driver vehicle hierarchy also was loaded following the above steps.)

Slowly Changing Dimensions

Driver Dimension is assumed to be the slowly changing dimension of the data warehouse where history management of such data are needed in order to keep a track of the drivers who got accident and the reason for those drivers to face accident will be helpful to predict the ways of minimizing accident and will help in enforcing road laws. Moreover in driver table I have used derived column to replace NUILL alcohol values with N in DimDriver.

Finally Start date and end date columns were included in this dimension for history management purpose



In the slowly changing dimension wizard, the attribute change types were done. The attributes Alcohol test_given, Driver_city, Driver_State, Driver_zip_code was considered as Changing Attributes basically implementing TYPE 1 implementation where these attributes will be updated when source data values change.

The attribute Driver_License_State was considered as Historical attributes basically implementing TYPE 2 implementation where a new record will be inserted in the target table when these attributes change in source table in order to ensur history management

Rest of the attributes Driver_DOB and Driver_Sex are specified as fixed attribute basically implementing TYPE 0 where nothing happens to the target when these values are changed in the sourceDB.

Test Slowly Changing Dimension

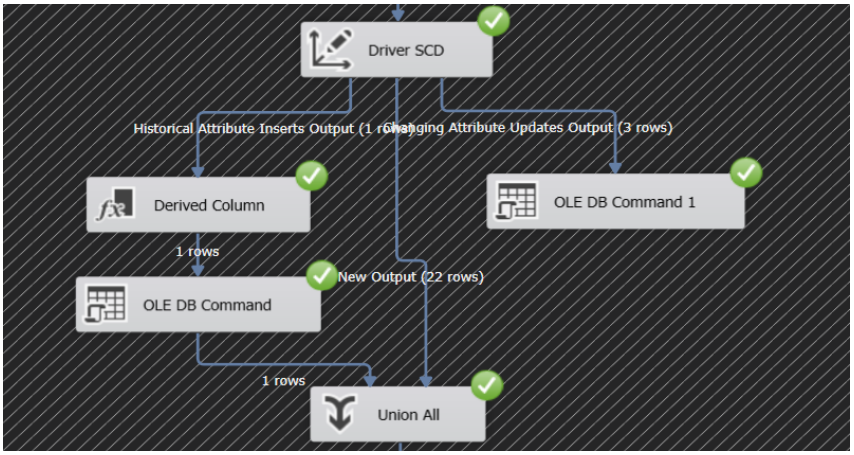
Following is a representation of the slowly changing dimension test

Type 1 Attribute – Changing Attributes

SQL CODE

```
Update [Road_Crash_StagingingDB].[dbo].[StgDriver]
set Driver_City = 'HAMMONTON'
where id = 765715
```

	DriverSK	AlternateDriverID	Driver_City	Driver_State	Driver_Zip_Code	Driver_License_State	Driver_DOB	Driver_Sex	Alcohol_Test_Given	InsertDate	ModifiedDate	Startdate	EndDate
1	1	765715	HAMMONTON	NJ	8057	NJ	1974-04-11	M	0	2022-05-08 11:28:45.293	2022-05-16 21:52:21.237	NULL	NULL

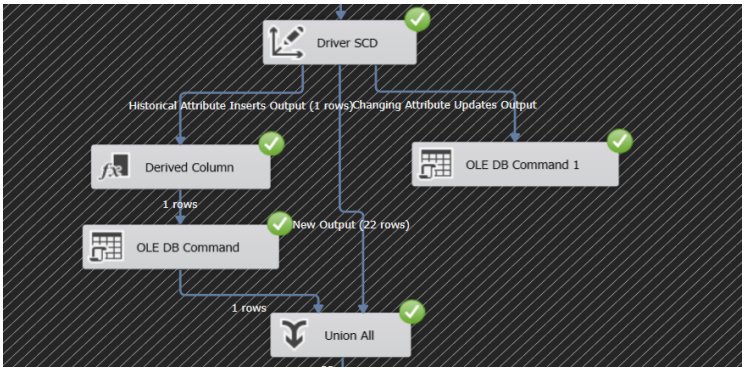


Type 2 Attribute – Historical Attributes

SQL Code

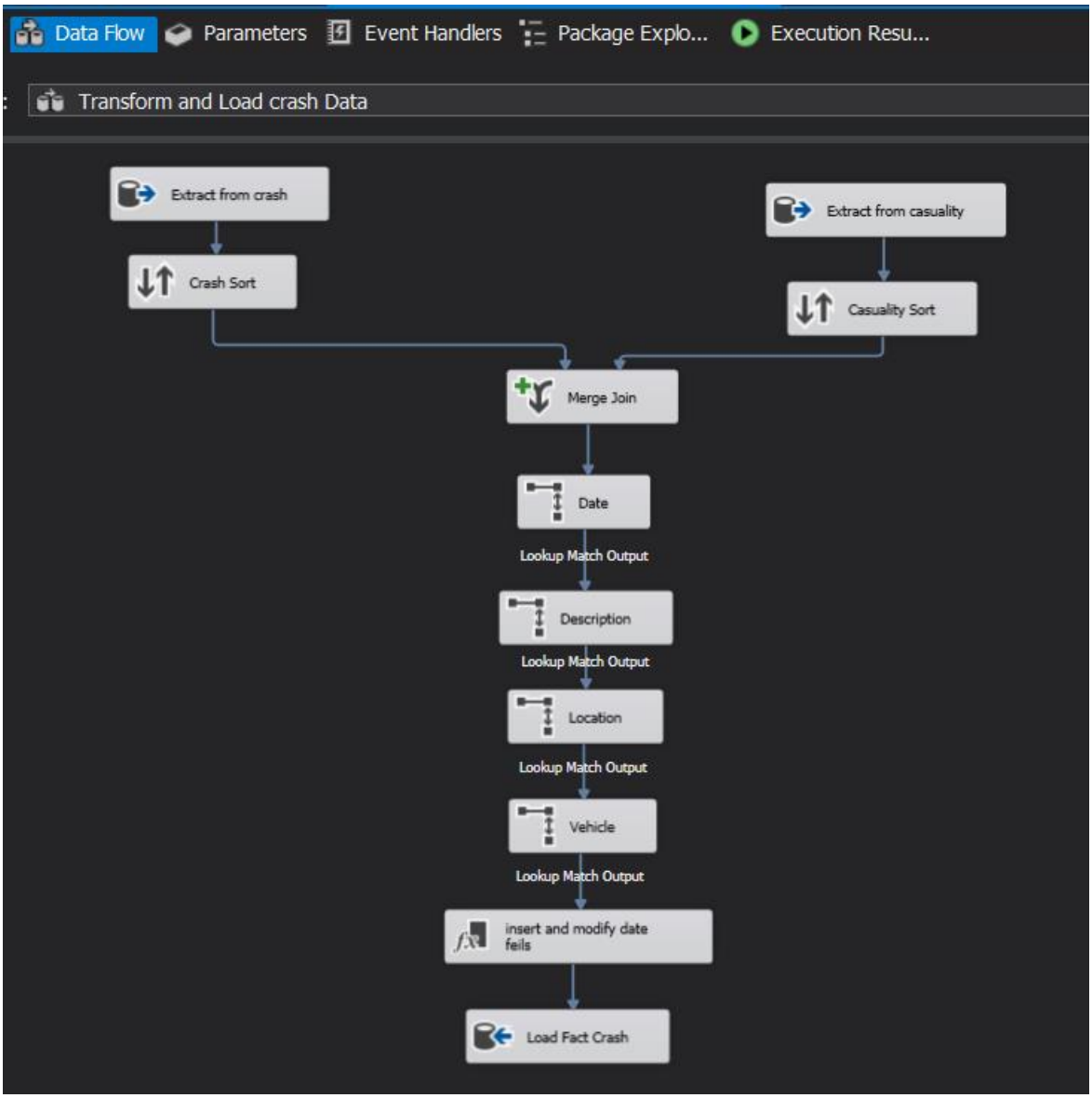
```
update [Road_Crash_StagingingDB].[dbo].[StgDriver]
set Driver_License_State = 'Npp'
where id = 602849
```

2	48753	602849	FAIRFIELD	NJ	7004	NJ	1991-07-21	F	0	NULL	2022-05-17 21:20:34.847	2022-05-09 11:18:32.000	2022-05-17 21:18:56.000
3	53498	602849	FAIRFIELD	NJ	7004	Npp	1991-07-21	F	0	NULL	NULL	2022-05-17 21:18:56.000	NULL



Fact Table

The final step of data extraction and loading to data warehouse is the process of loading data to Fact table. Before loading the fact table directly, a merging was done in order to merge casualty table with crash fact table using casualty_id.Here the crash table was sorted using casualty_id and the casualty table also was sorted using Casualty_id. Finally, both of them were merged using a merg join and loaded as factCrash. To complete it lookup components were used to obtain the key columns in the fact table. Fact table consists of four look up components referring to Dimvehicle, Dimdatetime, Dimdescription DimLocation respectively. The data flow for the Fact table loading is as given below.



Steps followed in Loading the fact table

1. First the crash table and the casualty table was sorted using casualty ID and merged together in order to create the FactCrash table in the data warehouse.
2. A lookup component was then used to extract the DateSK from Dim date by mapping datekey with fact table date_id in order lookup for the Surrogate key in DimDateTime.

3. Next another lookup was connected with to match the output of the date lookup and then it was used to extract the DescriptionSK from DimDescription by mapping alternatedescription_Id with description_ID of the fact table in order to lookup to the Surrogate key in DimDescription.

4. After that another lookup was connected to extract the LocationSK from DimLocation and it was used to map lat_long in location table with lat_long in the fact table in order to lookup to the Surrogate key in DimLocation.
5. Finnally, a lookup component was connected to match output of the location lookup and that was used to extract the VehicleSK from DimVehicle by mapping alternate vehicle_Id with vehicle _Id in the fact table in order to look up to the Surrogate key in DimVehicle.
6. Then Derived Column components are added to incorporate Insert date /Startdate into Transaction factcarsh table.
7. Finally, the combined data is loaded to the FactCrash in the datawarehouse with the usage of OLE DB Destination component.

Stored procedures used in the Data Warehouse table.

Since we are not maintaining the history of all tables except the Slowly changing dimension table Driver we can't track if any data was modified after inserting to the table. Therefore, as a solution stored procedure were created to identify the date of data inserted to the table and if any modifications were done to the existing records the modification date to be inserted accurately. So that we could identify if any data was changed after inserting.

Following are some Snapshot of those created Stored procedures,

```
CREATE PROCEDURE dbo.UpdateDimLocation
@latlong varchar(90),
@wweatherkey int,
@latitudeW varchar(50),
@wlongitude varchar(50),
@wcountry varchar (50),
@wstate varchar (50),
@localgov nvarchar(100),
@statarea varchar(50)
AS
BEGIN
if not exists (select LocationSK
from dbo.DimLocation
where lat_long = @latlong)
BEGIN
insert into dbo.DimLocation
(lat_long,weatherKey,latitude,longitude,country,
state,Local_government_Area,statistical_area,
InsertDate, ModifiedDate)
values
(@latlong, @wweatherkey, @latitudeW, @wlongitude, @wcountry, @wstate,
@localgov,@statarea,GETDATE(), GETDATE())
END;
if exists (select LocationSK
from dbo.DimLocation
where lat_long = @latlong)
BEGIN
update dbo.DimLocation
set
lat_long = @latlong, weatherKey = @wweatherkey ,
latitude = @latitudeW, longitude =@wlongitude,
country = @wcountry , state = @wstate , Local_government_Area = @localgov,
statistical_area = @statarea,
ModifiedDate = GETDATE()
where lat_long = @latlong
END;
END;
```

```
CREATE PROCEDURE dbo.UpdateDimVehicle
@VehicleID varchar (30),
@vdriverkey int,
@vmotorcycle int,
@vtrucksmall int,
@vtrucklarg int,
@vbus int,
@vtaxi int,
@vbycicle int,
@vscooter int,
@vpedestrian int,
@vinanimate int,
@vtrin int,
@vtram int,
@vvehicleother int
AS
BEGIN
if not exists (select VehicleSK
from dbo.DimVehicle
where AlternateVehicleID = @VehicleID)
BEGIN
insert into dbo.DimVehicle
(AlternateVehicleID,DriverKey,motor_cycle,truck_small,truck_larg,bus
,taxi,bicycle,scooter,pedestrian,inanimate,train,tram,vehicle_other,
InsertDate, ModifiedDate)
values
(@VehicleID, @vdriverkey, @vmotorcycle,@vtrucksmall,@vtrucklarg, @vbus,
@vtaxi, @vbycicle, @vscooter,@vpedestrian, @vinanimate, @vtrin, @vtram ,@vvehicleother,
GETDATE(), GETDATE())
END;
if exists (select VehicleSK
from dbo.DimVehicle
where AlternateVehicleID = @VehicleID)
BEGIN
update dbo.DimVehicle
set
DriverKey = @vdriverkey,
motor_cycle = @vmotorcycle,truck_small = @vtrucksmall,
truck_larg = @vtrucklarg,
bus = @vbus,taxi = @vtaxi,bicycle = @vbycicle,
scooter = @vscooter,pedestrian = @vpedestrian,
inanimate = @vinanimate,train =@vtrin,
tram = @vtram,vehicle_other = @vvehicleother,
ModifiedDate = GETDATE()
where AlternateVehicleID = @VehicleID
END;
END;
```

```
CREATE PROCEDURE dbo.UpdateWeather
@weatherid int,
@weathertime datetime2(0),
@temp  varchar(10),
@windchill varchar(10),
@humidity varchar (10),
@pressure float,
@visibilty float,
@winddirection varchar(40),
@windspeed varchar(10),
@precipitation varchar(10),
@weathercondition varchar(30)
AS
BEGIN
if not exists (select WeatherSK
from dbo.DimWeather
where WeatherID = @weatherid)
BEGIN
insert into dbo.DimWeather
(WeatherID, Weather_Timestamp,Temperature,Wind_Chill,Humidity,Pressure,
Visibility,Wind_Direction,Wind_Speed,Precipitation,
Weather_Condition,InsertDate, ModifiedDate)
values
(@weatherid, @weathertime, @temp,@windchill,@humidity,@pressure,@visibilty,
@winddirection,@windspeed,@precipitation,@weathercondition, GETDATE(), GETDATE())
END;
if exists (select WeatherSK
from dbo.DimWeather
where WeatherID = @weatherid)
BEGIN
update dbo.DimWeather
set
Weather_Timestamp = @weathertime,Temperature = @temp,
Wind_Chill = @windchill,Humidity=@humidity,
Pressure = @pressure,Visibility = @visibilty,
Wind_Direction = @winddirection,Wind_Speed = @windspeed,
Precipitation = @precipitation,
Weather_Condition = @weathercondition,
ModifiedDate = GETDATE()
where  WeatherID = @weatherid
END;
END;
```