**Advanced Spring Part 1 and 2**

**Carl Wentzel**

**P.S – I did not know I have done Part 2 as well**

**Data Understanding**

**School Data**

* Regional data (NSW, SA, VIC) school information and location
* Possible common columns Postcode & Suburb
* Contains no facts
* School\_Code is the unique value (Primary Key)

**Note:**

* Possibility combining each data set for school data into one as each dataset have the same columns
* Recreate an autoincrement for school\_code as there are duplicate school\_code numbers for different schools

**Transport Data**

* Regional data (NSW, SA, VIC) transport information and location
* Possible common columns Suburb
* Contains no facts
* Suburb & Stop\_Name unique key (Primary Key/Composite Key)

**Note:**

* Possibility combining each data set for transport data into one as each dataset have the same columns

**Crime Data**

* Crime information and location
* Possible common columns Suburb & Postcode
* Recorded Incidents fact
* Create a unique key (Primary Key/Composite Key) in staging table

**House Value Data**

* Regional data (NSW, SA, VIC) house value amount and location
* Possible common columns Suburb (NSW has Postcode so best exclude when staging)
* The house value is the fact
* Create a unique key (Primary Key) in staging table

**Note:**

* Possibility combining each data set for school data into one as each dataset have the same columns excluding NSW Postcode
* Remove all duplicates when loading into staging table.

**Rental Value Data**

* Regional data (NSW, SA, VIC) rental value amount and location
* Possible common columns Suburb
* The rental value is the fact
* Create a unique key (Primary Key) in staging table

**Note:**

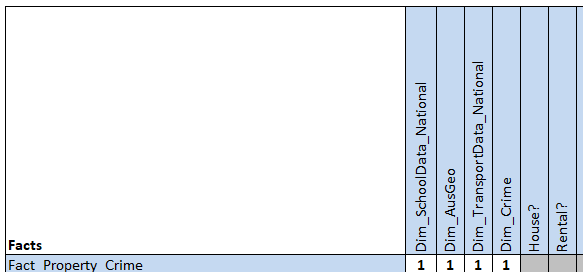
* Possibility combining each data set for school data into one as each dataset have the same columns

**Aus Location Data**

* Contains location data for Australia
* Lat and Long to pin point map locations
* No postcode included meaning Suburb will be used as the common column

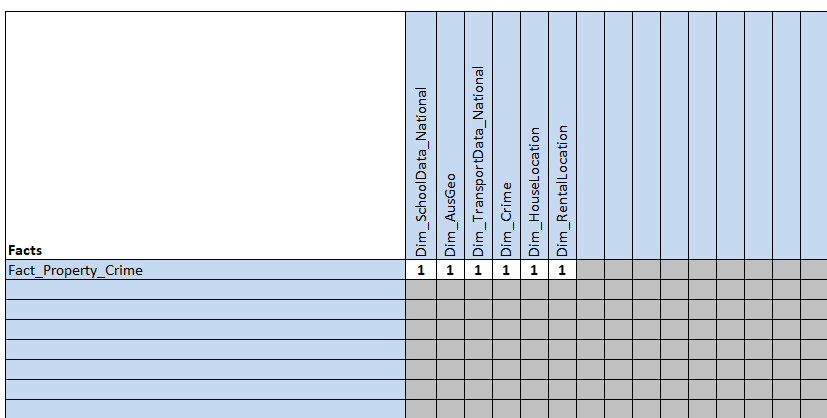
**BUS Diagram**

**Attempt 1 design:**

****

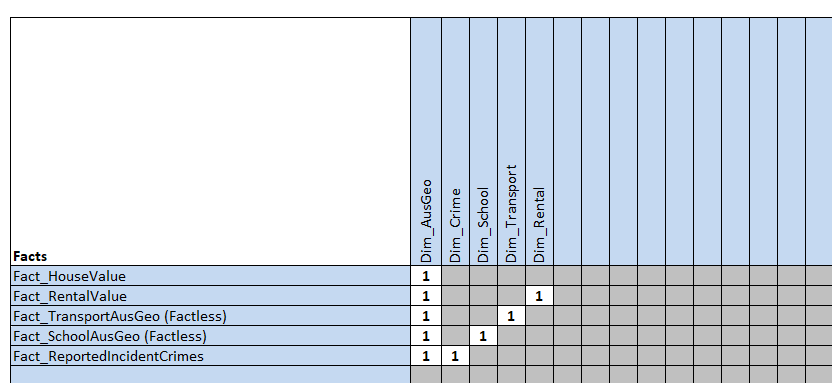
This design was my first design, as I thought having everything in the one Fact table was what I imagined the design to really be. I was uncertain with House and Rental but was willing to see what would happen in the implementation phase and alter from there.

**Attempt 2 design:**

****

After implementing the attempt 1 and running to a road block, this design looked closer to the goal but still had doubts on to have multiple fact tables that have a relationship with just two dimension tables.

**Attempt 3 design:**

****

This design made the most sense as different facts that were for different purposes did not feel right in one table. This also felt like it is what it should be. This could also resemble the Galaxy Schema where Dim\_Aus Geo will be used across all Fact tables.

**ERD (Entity Relationship Diagram)**

**A screenshot of a computer screen

Description automatically generated**

This ERD is a presumed scetch of the final result of what is to be achieved as the final result. Testing will be checked to see if its efficient.

**Data Loading**

****

For the loading & staging of the data, my aim was to make a data lakehouse database as a storage facility and then pipeline the data into the datawarehouse (separate database from the lakehouse) from staging to dimension and fact tables for the star schema design.

A screenshot of a computer

Description automatically generated

What I also wanted to ensure is that my naming convention were consistent, from source to destination, including all transformations which I later do in the staging process.

**Data Staging**

* Container Connection shot

**A screenshot of a computer

Description automatically generated**

* Close-Up Shot of naming conventions

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

During the process of staging, from the onboarding tasks I took with the necessary steps to reduce as much errors as possible during the transformation phase. This was by using RTRIM() for the common columns (suburb) to delete any spaces at the end of the string.

A screenshot of a computer

Description automatically generated

Another useful strategy was removing null value rows for data that had facts, such as HouseValue and RentalValue. Any data that did not have values was essentially not required for further analysis, and there was a very small minority of the data to make much of any impact to the final result.

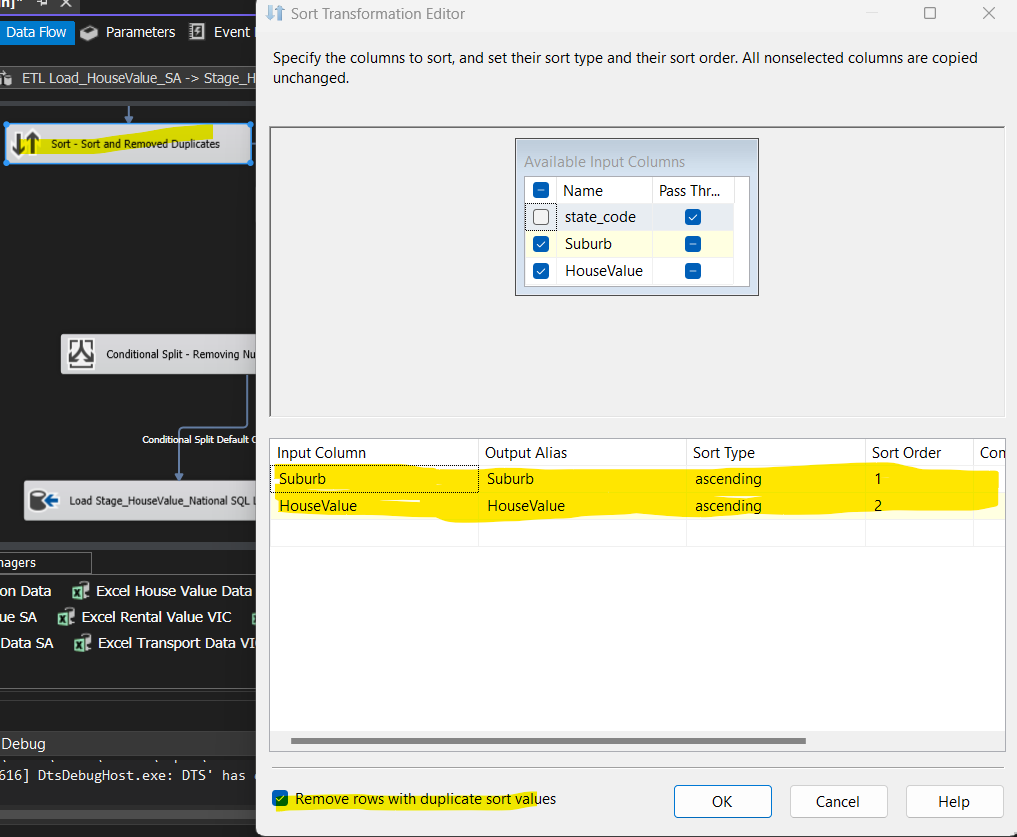
A screenshot of a computer

Description automatically generated

A screenshot of a computer

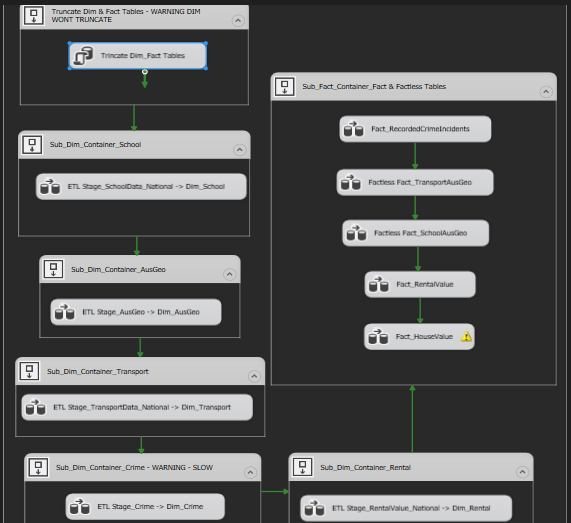
Description automatically generated

An added transformation that was useful (allthough not really needed as I found that I have ran an ETL twice to have duplicates) was knowing how to remove duplicates using Sort.

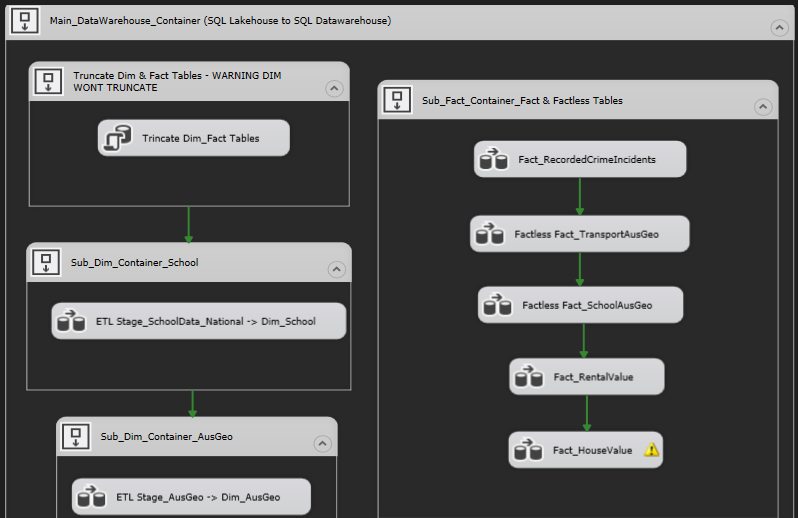


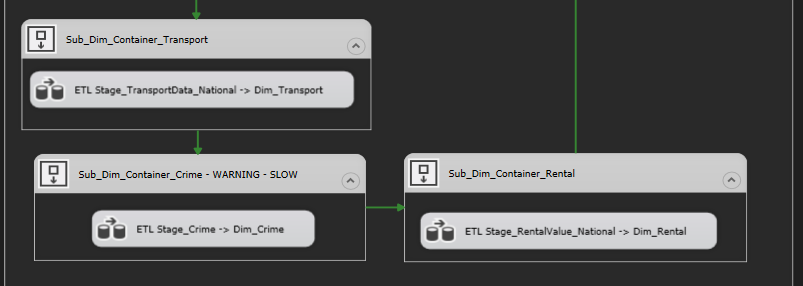
**Data Warehouse Star Schema**

* Container connection shot

****

* Close-Up Shot of naming conventions

****

****

This was still a process to get used to when creating Fact tables.

**NOTE FOR SELF**

Using 1 Key in the Fact table i.e. Dim\_Location

1. Use Staging Table with the fact.
2. Look Up the Dim\_Location table that has the common colum
3. Within the Look Up, connect the common columns and select the Dim\_Location key
4. In the destination, match up the columns as per the Fact table set up.

Using 2 Keys

1. Use Staging Table with the fact.
2. Look Up the Dim\_Location table that has the common colum
3. Within the Look Up, connect the common columns and select the Dim\_Location key
4. Have a second Look Up, connect with the Dim\_ that has the second Dim\_ Key OR have it at the source if its in the same database.
5. In the destination, match up the columns as per the Fact table set up.

Factless Fact Table

1. Source the Dim\_ table as the first table.
2. Look up the second Dim\_ table and connect common columns and checking the Dim\_ Key.
3. In the destination, match up the columns as per the Fact table set up.

Snowflake Design

1. Have an extra column for the Dim\_Key with the table directly connected to the Fact table.
2. Use that same table as the source.
3. Use Look Up to the second Dim\_ table with the common columns and select the Dim\_ Key
4. In the destination, match up the columns as per the Fact table set up.

With the Snowflake Design, its important to work in the outside layer and working in to the fact table. This requires concise planning vs. the Star Schema design where its one step Dim\_ tables.

**Issues**

**Common column inconsistency**

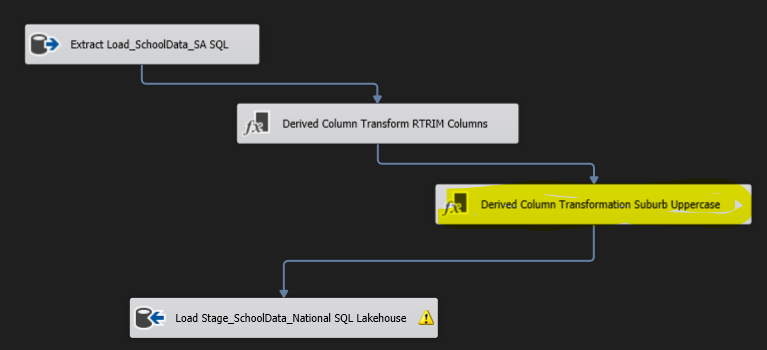
There was an error where the data would return NULL once flowing through from the stage tables to the fact tables, which required the suburb columns to connect to one another.

When identifying the suburbs, some of the suburbs would be in capital letters and some would be in first letter capital, rest small letters.

**Resolving the issue:**

With this in mind, the aim was to standardise and have all suburb columns throughout the datawarehouse be consistent. Through this process, all the suburb columns in the staging tables were transformed using a derived column, and made into all caps.

Doing a full refresh on the data in staging tables and fact tables allowed the data to flow with no issues.



A screenshot of a computer

Description automatically generated

**Performance issues**

Loading Dim\_Crime data from Stage\_Crime was an extremely slow process that would take 1-2 hours to fully load 360k rows x 6 columns with SCD (Slowly Changing Dimension).

This had to come with patience for the data to load, especially when changes were made to the data, such as the column consistency changes.

**Testing**

I have tested a view of the data in PowerBi, and the results were off. This was due to just connecting Suburb as the common columns.

What I haven’t concidered was that the same suburb name would appear in more than one state. For this I had to use look up in SSIS to link the state\_code (NSW, VIC, SA) and the suburb to which the data will result in a proper relationship/connection.

**Data Redundancy**

A team member raised concern on how to deal with data redundancy. This raised my eyebrows to look further into it. The crime data had multiple records of the same state/suburb/incident type but had a different number of incidents.

For this a query had to be made to aggregate the number of incidents and group by the supported columns shown to be duplicated.

This reduced the overall amount of data by over 100,000 rows.

This will reduced the amount of memory, and the amount of time when refreshes are required for the datawarehouse, for efficient reporting.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

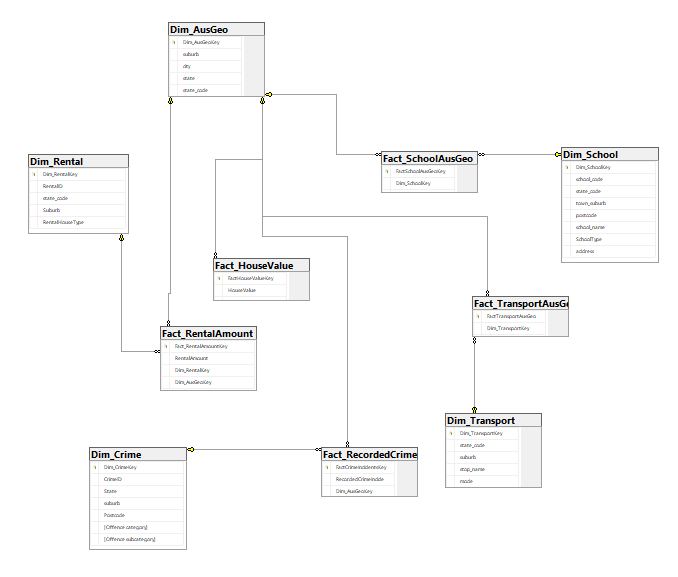
Description automatically generated

With this in mind, I want to further explore the aggregate command in SSIS to see if this could be done without a query.

After trialing out the aggregate command, when connecting to a look up, the aggregate breaks and a small percentage passed through. If connecting to something like conditional split, then the result shows properly. This could probably mean that when aggregating/grouping the data, it should first find a destination, then from that destination be used as a look up.

**Final Design**

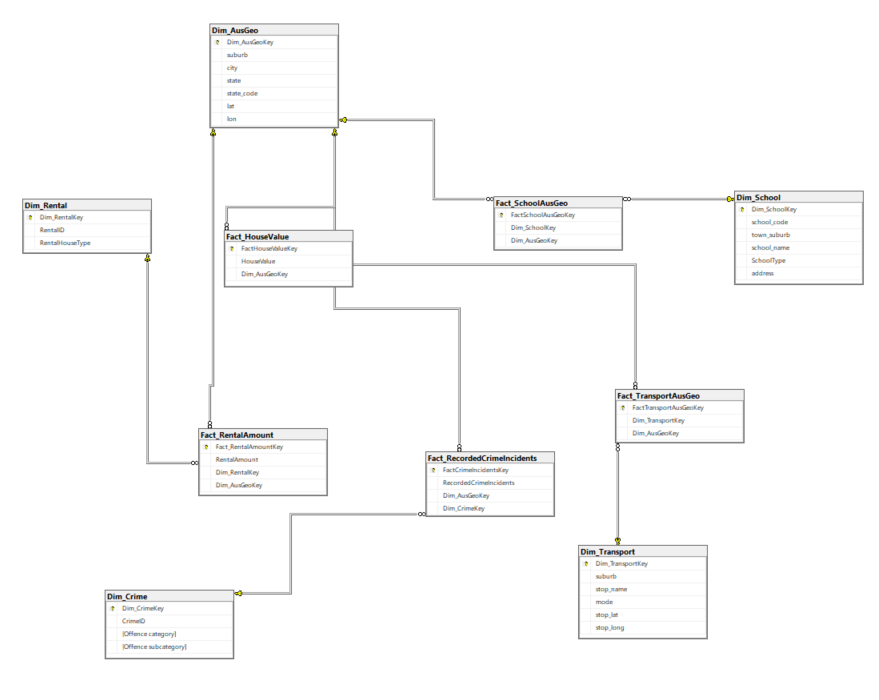
**1st design of Star Schema model**



As per the above diagram, there is a lot of duplicated variables that could be removed, such as the suburb. My goal is to reduce location data from all tables apart from Dim\_AusGeo.

**2nd Design of Star Schema Model**

After removing unwanted data columns, below is the final result:

****