Tech design: data warehouse testing

# Summary

The warehouse test engine will be built in Perl and composed of a number of modules.

It implements a series of checks on any data warehouse build provided to it.

The checks are:

1. For Each SOURCE SYSTEM
   1. Have all POLICY records been loaded
   2. Have all PREMIUM RISK records been loaded
   3. Have all PREMIUM PREMIUM records been loaded
   4. Have all CLAIM records been loaded
   5. Have all CLAIM MOVEMENT records been loaded
2. For each loaded row from each source system
   1. Can we reconstruct the inbound record from the warehouse and demonstrate a 1:1 mapping between the inbound records and the fact tables
   2. Have any business rules identified in the requirements been correctly implemented for each data item
3. For the warehouse overall
   1. Do we have data for each source system consistent with the anticipated amount based on all previous loads
   2. Do we have data for each year consistent with the anticipated amount based on all previous loads
   3. Do we have data for each carrier year consistent with the anticipated amount based on all previous loads
   4. Are the dimensions records unique in each dimension
4. For Performance testing of the warehouse
   1. Do the tests execute in the same timeframe as usual ( have we had a slowdown )

## Mod Handler

MOD HANDLER SHOULD:

Create an output file for the test run in the location c:\UDWTestResults

1. Timestamp this file
2. List all the input parameters
3. Take input parameters of The Entity under Test, The Source System that we wish to test and the target Database to Validate

Valid sources are:

1. PP
2. CMS
3. TCAS
4. Stingray

Valid Targets are

1. UAT
2. SIT

Valid Entities are:

1. Policy
2. Claim
3. claimMvmt
4. Premium

To identify a specific database within a server the ODBC connection is changed.

## Mod Source

Mod Source is responsible for building two left side data tables for comparison by the rules engine with the right side target tables. It should return both a FACT and a DIMENSION table.

Mod source will require the following routines but will only ever run & return one of them

1. sub tcas\_policy{}
2. sub cms\_policy{}
3. sub pp\_policy{}
4. sub stingray\_policy{}
5. sub tcas\_premium{}
6. sub cms\_premium{}
7. sub pp\_premium{}
8. sub stingray\_premium{}
9. sub tcas\_claim{}
10. sub cms\_claim{}
11. sub pp\_claim{}
12. sub stingray\_claim{}
13. sub tcas\_claimMvmt{}
14. sub cms\_claimMvmt{}
15. sub pp\_claimMvmt{}
16. sub stingray\_claimMvmt{}

## Rules Modules

The rule modules implement a source and target agnostic set of rules .

### Table Compare

Table compare performs a column by column row by row comparison of two tables passed to it. The tables need to share a common set of column names.

This is implemented as a full outer join between the left and right tables filtering on any row that contains a NULL on either side, and passes where there are no records in the result set.

Because MySQL does not support the full outer join this is implemented as a UNION operation on a LEFT OUTER JOIN and a RIGHT OUTER JOIN which is logically equivalent to the FULL OUTER JOIN operation.

### Are all Dimensional Records unique

Q4) Are all Dim. Records there

Q5) Can we show a 1 – 1 relationship between source and target

Q6) Is line by line item data correct

## We are dealing with:

{ Array } [ Handler] - data, output file, what DB did we test, who.entity.source

1. Source
2. Type Of Entity
3. Target DB To Validate

* Set up reporting files
* Make & Check DB Connections [Mod DB]
* Passes These Elements To:

[ Mod Policy ] or [Mod Claim ] or [ Mod Claim Mov ] or [Mod Prem]

Load Rules

[Mod CMS] {Parameters, YOA, Carrier} {Entity Type}

Source Test Target

|  |
| --- |
| Generates Table |
| Run Rules |

Sel Mod CSM Policy Tables Mod Policy

AB Premium Tables

Claim Movement

|  |
| --- |
| Generates Table |
| Run Rules |

Claim Mod Prem

## Design

|  |
| --- |
| Date |
| DB |
| Entity |
| Source |

Handler = inputs needed DB {Source & Target) IN

|  |
| --- |
| Output |
| Table 1 |
| Table 2 |

Mod\_Target (Entity) SourceHandler

Mod\_Source (Entity) SourceHandler

Mod\_Source ( Entitiy , Source)

IF Src = PP Then { IF Ent = Pol { } }

Mod Rules

Column – names

Vaules   
Keys   
RowCounts

With Left Table   
Get all Col Nmes

Select \* From Table.

Join Right Table on

Col name = Colname

Components

* Mod\_Handler\_F

Mod\_Policy\_F Mod\_Policy\_D

Mod\_Claim\_F Target x2 Mod\_Claim\_D

Mod\_Premium\_F Mod\_Prem\_D

Mod\_ClaimMovement\_F Mod\_ClaimMovement \_D

Mod\_Source (entity, Source)

16 options

Mod\_DB ( String)

Mod Rules

**Modularization**

Contents

[Summary 2](#_Toc508699560)

[Mod Handler 2](#_Toc508699561)

[Mod Source 2](#_Toc508699562)

[Mod Rules 3](#_Toc508699563)

[We are dealing with: 3](#_Toc508699564)

[Design 4](#_Toc508699565)

[Mod\_Source 6](#_Toc508699566)

[Mod\_Database 6](#_Toc508699567)

[Mod\_TCAS 6](#_Toc508699568)

# Mod\_Source

This module utilizes the Mod\_database subroutines to call the data required e.g. CMS Claims data, and then creates an oracle table and sends the data to the created table for comparison with source data.

# Mod\_Database

This is the module that contains all database connections required for the: PP policy, PP premium, PP claims data, CMS policy, CMS premium, CMS claims data, and the TCAS policy, TCAS premium and TCAS claims data. They will be used interchangeably within other modules to call data where needed.

# Mod\_TCAS

This module takes TCAS policy, Claim, Premium and Claim Movement data from staging, creates a table relevant to type of data being tested and exports the data to the created table in MySQL. Also the module contains script that logs errors that occur during data insert and inserts the error in to an error table in MySQL.

## Mod DB\_TOOLS

This is a helper package implementing toys and tools for playing with the warehouse. No specific requirements identified initially however it now contains an error logger and a function to return all the column headers in an array passed to it.

Future development will include a function to automatically log a bug to Bugzilla.