OREGON DEPARTMENT OF TRANSPORTATION

ENTERPRISE DATA WAREHOUSE

ODOT Data Warehouse 2.0

AUTHORS
Nithin Reddy, Shyam Maddi, James McDonald

This document outlines the Oregon Department of Transportation’s Enterprise Data Warehouse.
# Contents

1.0 History of the Warehouse ................................................................. 2  
2.0 Why an Enterprise Data Warehouse .................................................. 2  
3.0 Current Data Mart Implementation .................................................... 3  
4.0 DW_FDI_OLAP & DW_HBO_OLAP COMPARISON ............................. 4  
5.0 Benefits of an Enterprise Data Warehouse ......................................... 6  
6.0 The EDW Data Model ......................................................................... 7  
7.0 Fact Audit Trail All ............................................................................ 9  
8.0 Steps to Achieve the EDW ................................................................. 12  
9.0 Tests Cases against the EDW ............................................................. 14  
   9.1 Loading of the FACT TABLES – data integrity ................................. 14  
   9.2 PowerBI ......................................................................................... 14  
   9.3 Report Builder .............................................................................. 16
1.0 History of the Warehouse

2005 – ODOT begins work on its first data warehouse
2009 – Data Warehouse moves into SQL 2008
2010 – More sources are brought in and more data marts are created
2012 – Data marts begin borrowing data from one another
2016 – Warehouse moves into SQL 2014
2017 – A vision of one Enterprise Data Warehouse begins

2.0 Why an Enterprise Data Warehouse

“You can catch all the minnows in the ocean and stack them together and they still do not make a whale,” Bill Inmon, January 8, 1998.

We are entering a critical time for our warehouse, where more sources are being brought into the warehouse and each data owner wants pieces of the other owner’s data. As our current architecture consists of siloed data marts we are seeing crossovers from one data mart to the next, which forces ETL developers to try to remain aware of the interconnections between data marts so as to not cause processing issues. This will eventually come to a head and force us to either untangle everything, or just start over. It is the vision of this document that the time to begin untangling is now so as to provide ODOT with a true enterprise data mart where data can be shared between owners in a formalized, documented, and secure way.
3.0 Current Data Mart Implementation

FINANCIAL DATA MART DATAFLOW

PROPOSED EDW DATA MART DATAFLOW

Further development will allow the cubes to be consolidated into one.
4.0 DW_FDI_OlAP & DW_HBO_OlAP Comparison

The best place to start the merging of solio’ed data marts is with DW_FDI_OlAP, DW_HBO_OlAP and DW_MASTER because the two finance data marts are so close to one another, with regards to database objects and data, and bringing in DW_Master allows for the merging of a couple tables that are conformed already and used by most all other data marts. Data being in one database instead of two, in case of production job failures, will be available to users faster than now.

Overview of DW_HBO_OlAP and DW_FDI_OlAP databases:

**DW_FDI_OlAP**
- 58 tables
- 46 views
- 13 stored procedures
- 1 function

**DW_HBO_OlAP**
- 45 tables
- 27 views
- 12 stored procedures
- 2 functions – actually one just duplicated in another schema

- DW_FDI_OlAP contains the 11 forecasting tables, 2 CService tables that are not in DW_HBO_OlAP.

- DW_HBO_OlAP contains the new features inventory tables, the greenbar report table, and the HBUD tables that are not in DW_FDI_OlAP.
- 24 tables between the two databases are different based upon more columns existing in DW_FDI_OLAP that are not in DW_HBO_OLAP.
  
  o DIM_AGREE, in DW_FDI_OLAP has revenue_source, part_ind, agy_srce that do not exist in DW_HBO_OLAP.
  o DIM_EA in DW_FDI_OLAP contains the FUEL_TYPE and FLEET_CODE columns that do not exist in DW_HBO_OLAP
  o DIM_PCS in DW_FDI_OLAP contains CF_STATUS which doesn’t exist in DW_HBO_OLAP
  o The FACT_AUDITTRAIL_* tables in DW_HBO_OLAP contain the document_date_SK column, Let_date_sk, and Completion_date_sk
  o DIM_VNDR in DW_HBO_OLAP contains EMPL_POSITION_CREW that doesn’t exist in DW_FDI_OLAP.

- There are index differences between the tables, but we can merge these together for the best performance between the two, tuning as we go.

- The stored procedures between the two are relatively the same, just copied in each to support the separation.

The size of the database is very similar as is the configuration of the database so we will be saving 70+GB of space per server, as well as memory and processing power of the server if we combine these databases.

As for DW_MASTER, we will be bringing in date, location, and agency tables that both databases use. *Please note: we have found some data in DW_MASTER is pulling from a source that no longer exists (HTOCS_REPORTING).*
5.0 Benefits of an Enterprise Data Warehouse

- We have one version of the warehouse truth.
- We limit the duplicated data across our warehouse saving space on the server, processing power, time, and the effort that goes into maintaining various copies of almost the exact same data, if not the exact same data.
- Security can be dialed in better because of a consistent understanding of what users need access to what objects and rather than just provide blanket select access we are forced to further refine our security model, working in conjunction with the business and program manager, relying upon data sharing agreements, reflecting their intent in database roles.
- The relationships across all dims and facts can be established, even across schemas, which will benefit report writers and our global understanding of what relates to what.
- We have one PowerDesigner model used to manage our warehouse environment, housing all tables/columns/schemas and definitions, which can be shared with end-users either through the PowerDesigner Portal, or we can send them a pdf/jpeg of the model.
- Data is easier to share between customers/reports/processes because well Defined schema names provide an ease of understanding of what data is available in our warehouse.

- We do not have to maintain two sets of ETLs that do almost exactly the same thing.
The EDW Data Model

From its inception, the EDW database was built using PowerDesigner, following DA standards as best as can be achieved, while still ensuring we break nothing as we move forward. This PowerDesigner model re-uses definitions from the TEAMS data model so that a column that exists in teams, will share the same definition as a column that exists in EDW.

From the EDW PowerDesigner data model we build a database and merge any changes into TFS where we capture security, views, procs, and any other developer objects outside of the tables and relationships of the database.
The diagram below shows a database with relationships and one without. As you can see the ability to understand what is related to what, is much easier with such relationships, and the supporting diagram/documentation.
7.0 Fact Audit Trail All

Further exploration of DW_FDI_OLAP and DW_HBO_OLAP showed that the audit trail bienniums have been split into their own tables, but in the EDW database we have merged all audit trail tables into a single partitioned table, making it easier for management of the data, without losing performance. This will also benefit the PowerDesigner model in that we will not have to relate the financial dimensions to multiple fact audit trail tables.

Therefore, merging our fact tables and not splitting them into separate biennium tables provides the benefit of not having duplicated meta data(table/columns) across multiple tables and ensuring consistency amongst our bienniums. Last, but not least the merging of these audit trail tables into one table provides the ETL developers the benefit of changes occurring in one place, not one place for each biennium, forcing maintenance and new tables to be created at each biennium’s inception.

Every two years at the time of biennial rollover, BIDW team would end up having almost NO rollover maintenance work if we have only ONE table.
MULTIPLE FACT AUDIT TRAIL TABLES

Each grouping below represents a separate fact_audit_trail, based upon biennium. This can cause confusion when establishing relationships and providing access to end-users to tables, where they can benefit from said relationships.
ONE FACT Audit Trail Table

Here we have merged all fact_audit_trail tables into a single table partitioned by biennium, at the database level. This will reduce maintenance across the database and the supporting ETLs.
8.0 Steps to Achieve the EDW

Below is a summary of the steps necessary to build, test, and implement the Enterprise Data Warehouse solution.

1. Reverse engineer DW_MASTER
2. Reverse engineer DW_HBO_Olap
3. Reverse engineer DW_FDI_Olap
4. Merge DW_HBO_Olap and DW_FDI_Olap together, keeping the larger of the objects (tables, columns)... meaning if one table has additional columns, these will be brought in, with analysis done to see the effect in both cubes.
5. Create business names in the PowerDesigner model and capture all definitions.
6. Create all relationships as foreign keys between facts and dims.
7. Once DW_HBO_Olap and DW_FDI_Olap are merged together, we merge them into DW_MASTER, and change the schema from dbo to FINANCE, or HBO, if there are specific HBO objects (tables). Except for the views, and that’s what allows for the quick transition into EDW, away from the other 3 databases.
8. Then we create the necessary TIODS objects to support the features inventory reports, placing them in the HBO schema.
9. Create the actual database EDW
10. Create filegroups and files to support the separation of each schema and data/indexes. Create additional filegroups for larger tables, such as the audit trail tables.
11. Create the necessary security/AD groups, utilizing database roles as the only way to gain access... ensuring nobody has db_datareader, except for those that should, developers, etc.
12. Copy the ETL’s for DW_FDI_Olap and DW_HBO_Olap, modifying them to redirect and write to EDW, and storing the ETLS in the SSIS DB catalog in the DM folder.
13. Run/test/fix ETL’s till they work.
14. Index tune for performance... creating the nix index/create index stored procs.
15. Create FK’s... creating the drop fk/create FK stored procs.
16. Copy the cubes, finance and HBO
17. Redirect the copied cubes to the EDW
18. Run/test/fix cubes till they work.
19. End-users test by redirecting reports to EDW.
20. If satisfied and see benefit, promote to prod, else delete everything so as to not confuse.
21. If success bring in other data mart data, implementing the same level of rigor that was done for the initial 3 data marts.
9.0 Tests Cases against the EDW

9.1 Loading of the FACT TABLES – data integrity

One thing that we’ve discovered is that the DW_FDI_OLAP database contains data that is either no longer in TEAMS or has changed in such a way that unless we feed the EDW database from DW_FDI_OLAP, for the initial load, we end up with foreign key violations if/when we try to load prior bienniums. This is a good thing in that we are better able to understand changes in data and how they affect pre-existing fact data. Further analysis should be done so as to understand what state the data discrepancy between the source system, the staging database, and the data mart are in.

9.2 PowerBI

PowerBI likes relationships between tables, and our end-users will benefit from the establishment of relationships. Below is a screenshot in PowerBI pointing to the DW_FDI_OLAP database and relying upon PowerBI to establish relationships based upon similarly named columns.
Below is the EDW database where relationships have been established between fact and dimension tables. PowerBI automatically picks up the relationships and uses them to help end-users write better, faster reports.

The use of relationships will provide our end-users writing reports with a better understanding of what data relates to what and how.
9.3 Report Builder

Report Builder 2016 also benefits from the pre-established relationships and automatically provides the SQL statement necessary to join tables that are related without the end-user having to figure out the joins.