Materialized Views

Euphoria, Reality and Implementation

Table of Contents

Euphoria

- Development
  - 3rd party ETL tool
    - Data Modeler / Business Analyst & Developer (with additional skills)
  - PL/SQL
    - Data Modeler / Business Analyst & Developer
  - Materialized Views
    - Data Modeler / Business Analyst

(note: no DBA specified)

Euphoria
(continued)

- Execution (test of an existing production aggregation process for three months of data / i.e. the daily process)
  - 3rd party ETL tool
    - Complete 3 months – 60 minutes
  - PL/SQL
    - Complete 3 months – 10 minutes
  - Materialized Views
    - Complete 3 months – 10 minutes

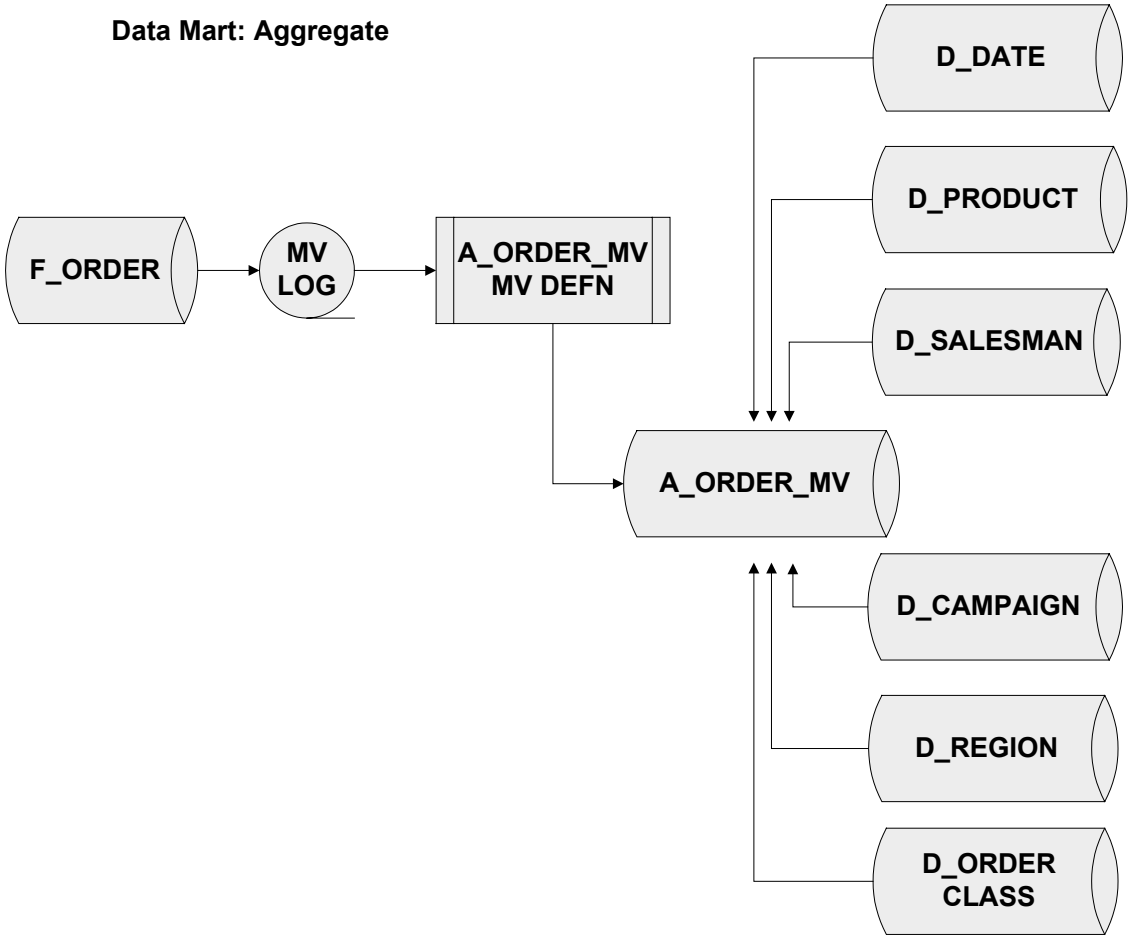
Euphoria
(continued)

- Execution (a fast refresh of a normal days' data volume)
  - Refresh of ODS ORDER Materialized View
    - 1 second. Yes, one second
  - Refresh of Data Mart A_ORDER aggregate materialized view
    - 3 seconds. Yes, three seconds (vs. one hour)
    - A bi-product was more accurate data (re-aggregate of all changed order data)

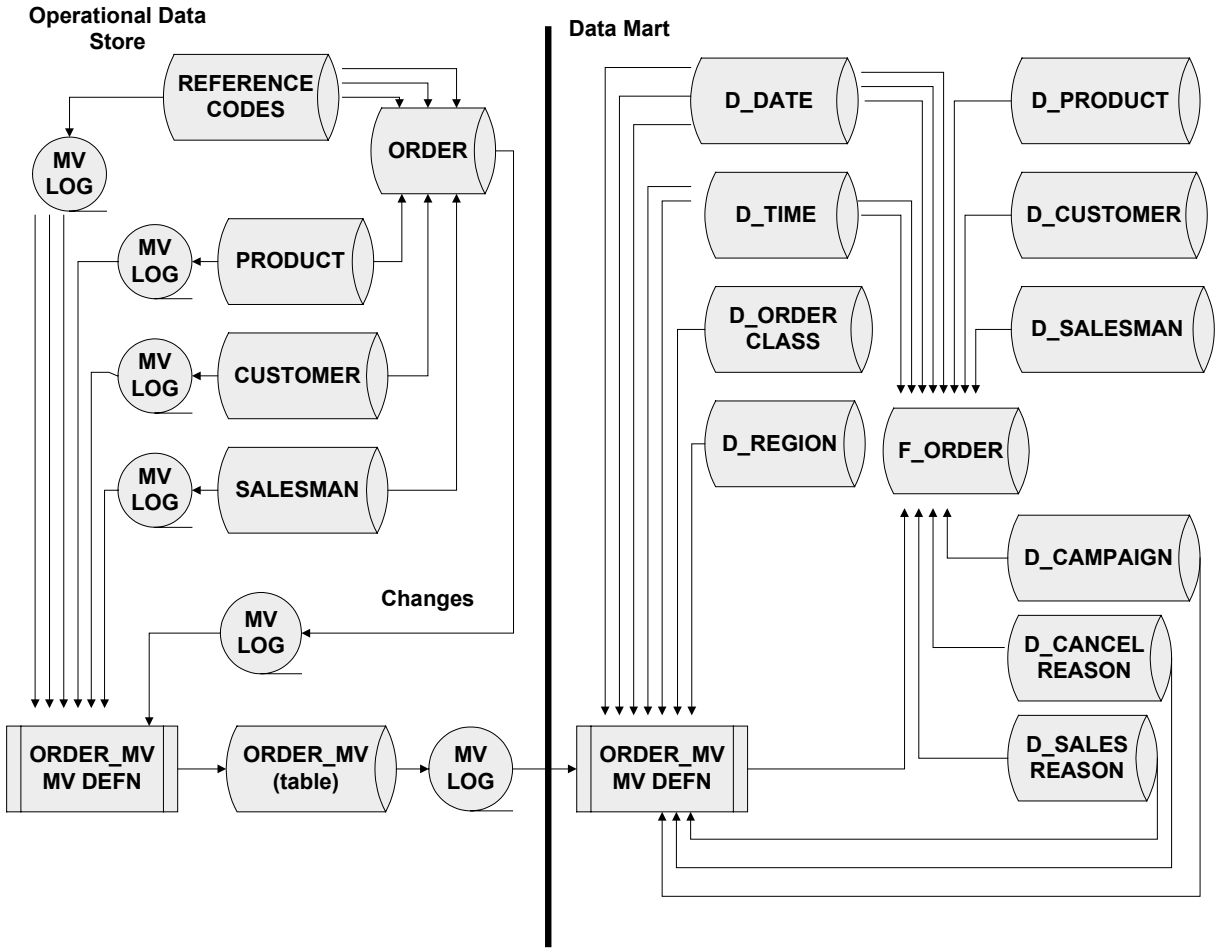
Reality
(Aggregate Materialized View)

- Out of the box
  - Materialized view log was investigated and created in ten minutes
  - Materialized view (aka: snapshot) was developed in ten minutes
  - Started testing refresh immediately
  - Started showing off, one minute later

Reality
(Aggregate Materialized View)



Reality
(ODS and Data Mart: Fact)



Reality
(with a FAST execution objective)

- Common error message:
  - ERROR at line nn:
    ORA-12015: cannot create a fast refresh materialized view from a complex query

- Can NOT use multiple joins to a table
- Can NOT use sub-queries
- Can NOT use mix outer and inner joins
- Can NOT use nested materialized views with connection strings to remote databases
- Can NOT create views on a table, and use both objects in your where clause

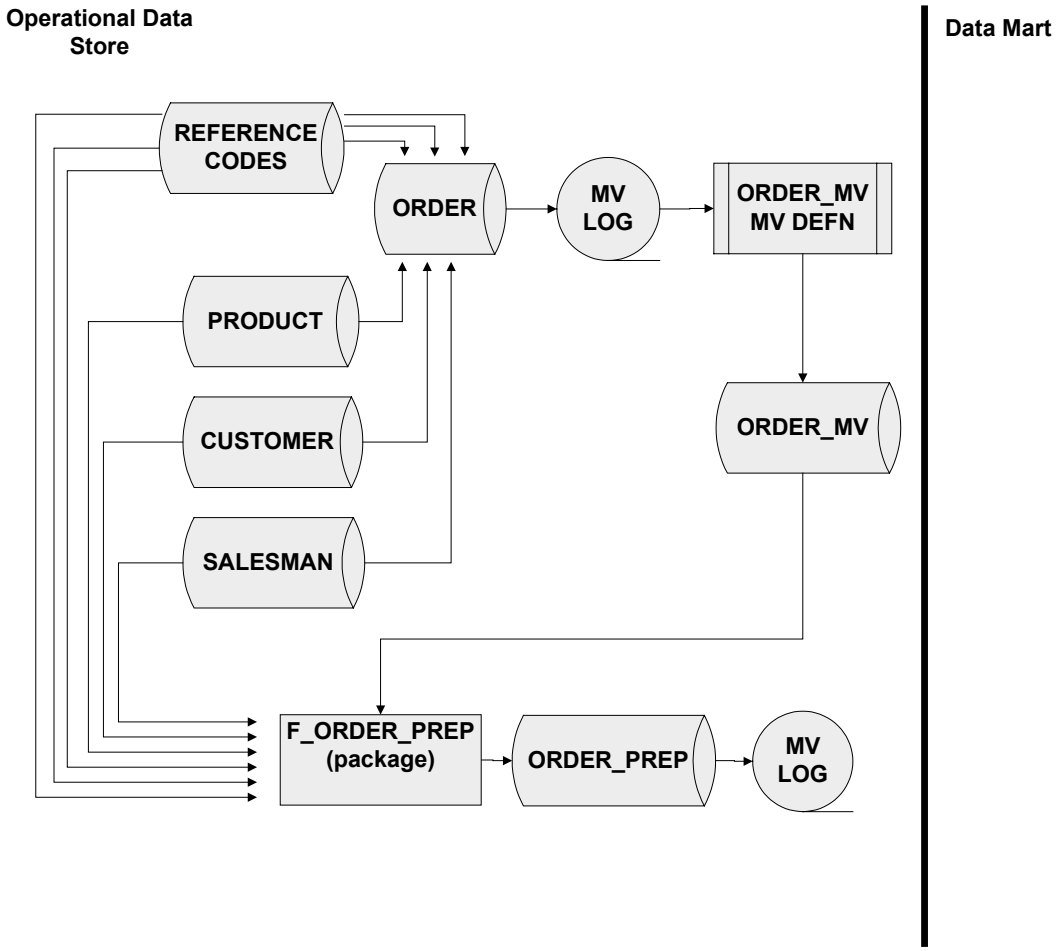
Implementation
(possible changes of existing designs)

- De-normalize data to include reference data
- De-normalize data to include dimensional ID(s)
- Use of ODS IDs as dimensional table ID(s) for dimensional tables
- Creation of dimension table ID(s) in ODS (instead of in mart)
- Include ODS IDs as columns in a target FACT
  - ODS primary key becomes FACT primary key (e.g. ORDER_ID)
  - Reduces use of composite dimensional keys as FACT primary key
- Use of 'with new values' property of materialized views

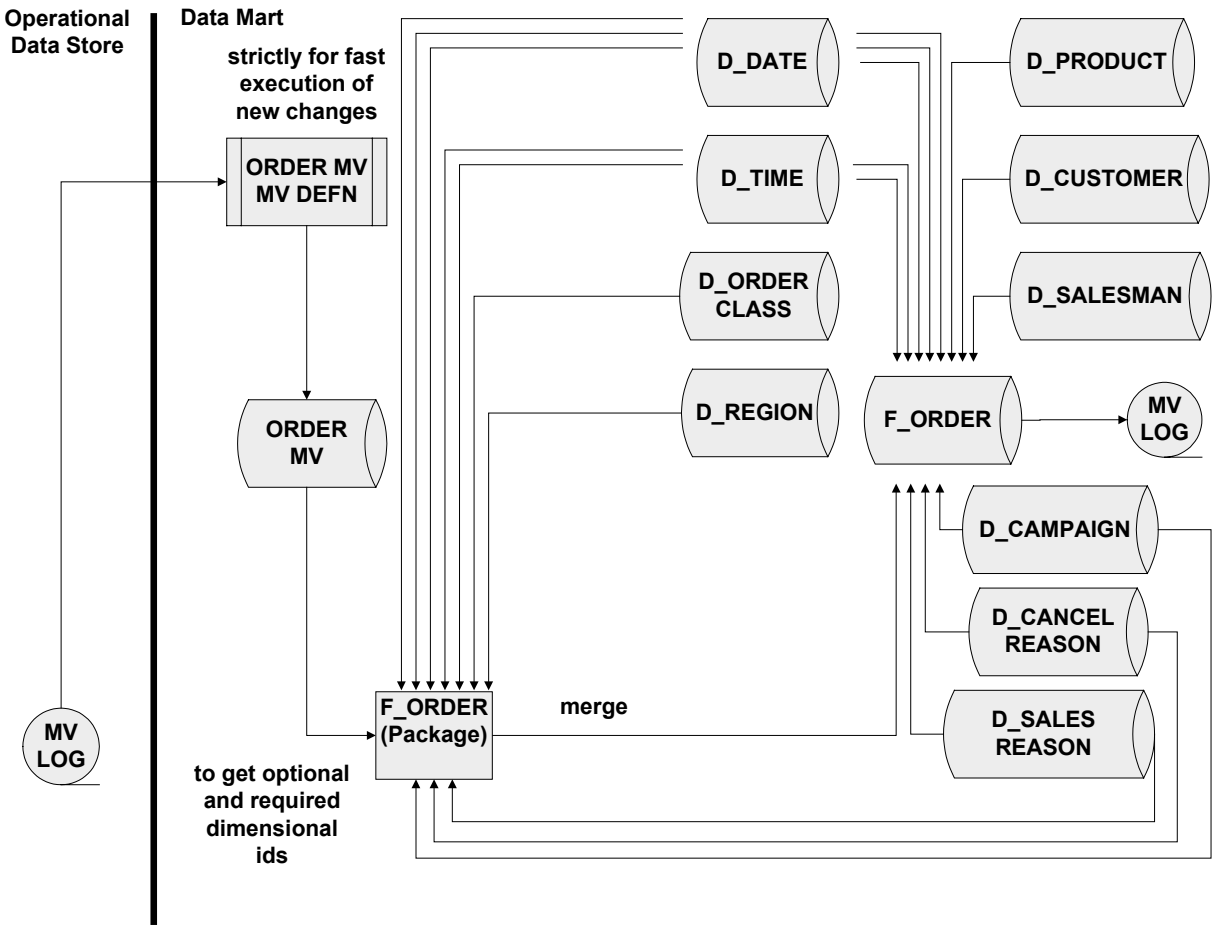
Implementation
(possible changes of existing designs)

- Use of PL/SQL replacing other vendor ETL tools
- Use of source system' primary keys or rowid(s) in target dimension or fact tables for fast execution objective
- Use of replication as part of "traditional" analysis, design, proto-cycling and application development
- Use of materialized view logs

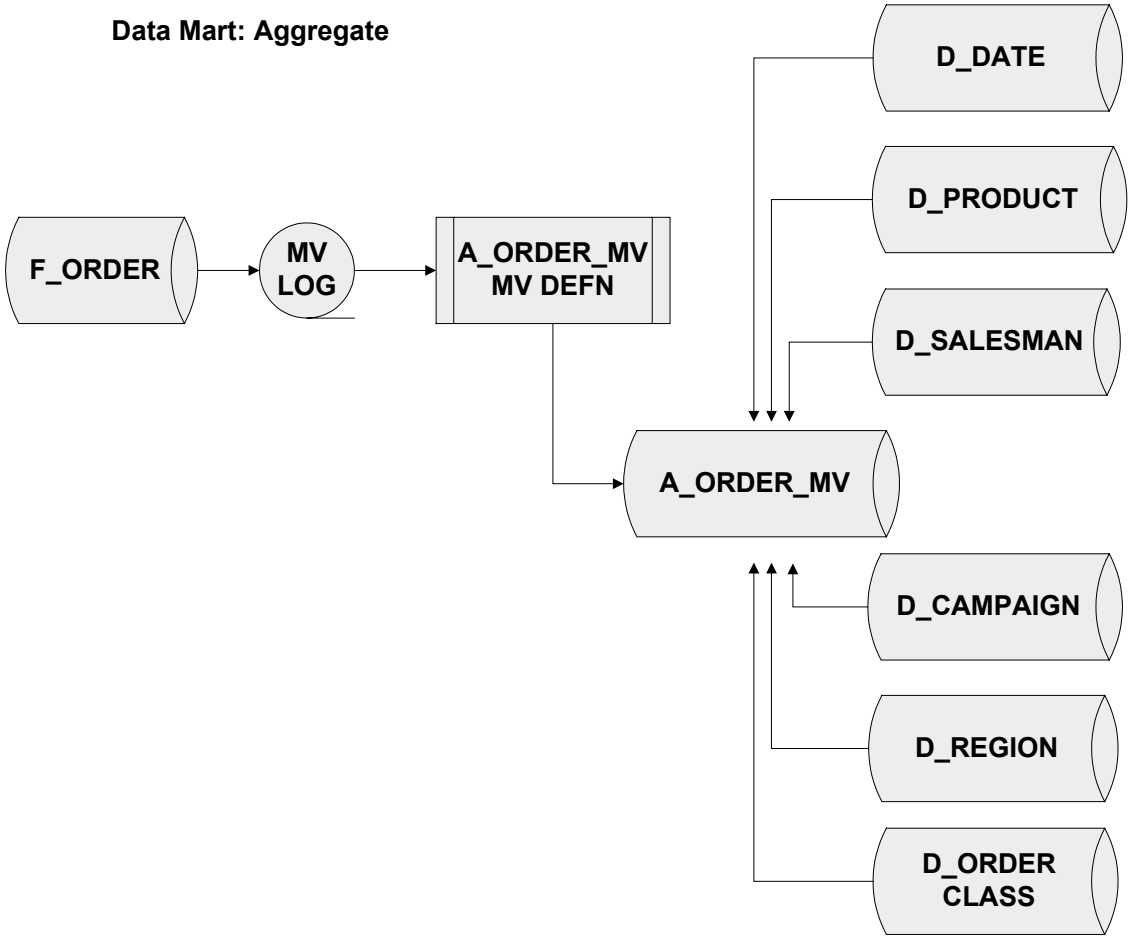
Implementation (ODS)



Implementation
(Data Mart: Fact)



Reality
(Aggregate Materialized View)



Tips & Techniques

- PURGE_LOG Procedure
  - DBMS_MVIEW.PURGE_LOG('TABLE_MV_NAME', NUMBER);
    - TABLE_MV_NAME = Name of the table or materialized view
    - NUMBER = Number of least recently refreshed materialized views whose rows you want to remove from materialized view log.
- Refresh of materialized views
  - DBMS_MVIEW.REFRESH ('MV_NAME', 'TYPE');
    - Refresh Type: f = fast, c = complete, n = never

Tips and Techniques
(continued)

- When multiple joins are required for a fast execution
  - Create a source system key (possible composite key) / target system key cross-reference table(s). e.g when two or more period dimensions are required for a materialized view fact
- Conversion of data
  - Use of pre-built tables for large amounts of data
  - Use of existing tables from current production assets
- You can build multiple logs (possible for multiple dimensional materialized views) off of one table

Tips and Techniques
(continued)

- Why use pre-built tables for materialized views?
  - Existing tables to be used in materialized views
  - Conversion of large amounts of data
  - Adding partitions for additional data
  - Dropping partitions for older data no longer required or that has passed out of SLA agreements

Tips and Techniques
(continued)

- Enable Query Rewrite
  - Set system parameter, you must set:
    - QUERY_REWRITE_ENABLED initialization parameter to TRUE, before using query rewrite
    - OPTIMIZER_MODE = all_rows, first_rows, or choose
    - COMPATIBLE = 8.1.0 (or greater)

Tips and Techniques
(continued)

- Enable Query Rewrite (continued)
  - You must also specify ENABLE QUERY REWRITE clause in the materialized view definition, if it is a candidate for its use
  - Allows optimizer to redirect user queries to aggregate table vs. the table the query was directed to use

Possible Land Mines

- Large logs of unneeded data
- Coordination requirements during parallel development efforts using the same table
- Urge to use materialized views as an ETL process

Possible Land Mines
(continued)

- Corrupted materialized views
  - While a materialized view refresh was in progress, we bounced the db with the source materialized view log
  - Errors
    - ORA-00955: name is already used by an existing object
    - ORA-12003: snapshot "owner"."snapshot name" does not exist
    - There are 2 entries in OBJ$ without corresponding entries in the user_snapshots and dba_registered_snapshots objects
  - Metalink Doc ID: Note: 221775.1

Bi-Products
(from exposure to materialized views)

- New standard designs for:
  - work queue processing (materialized views)
  - maintaining dimensional tables (materialized views)
  - maintaining fact tables (possibly, materialized views)
  - data mart aggregates (materialized views)
  - application replication (materialized views)