Building a dimensional data warehouse for RetailXpress

Oamen Modupe
oamenmodupe@gmail.com

TABLE OF CONTENTS

1. Understand the problem and define the use case.

We need to understand what the company/client does, what solution they want, why they are doing this, what the current process/pipeline looks like, and how we are going to achieve it. This is going to be gathered through a series of interviews.

RetailXpress is a mid-sized retail company specializing in e-commerce and physical stores. They sell a wide range of consumer products, including electronics, fashion, and home appliances. They rely on robust online and in-store systems to manage inventory, process transactions, and analyze customer behavior.

What they want: RetailXpress wants to enable self-service analytics for its business teams, including marketing, sales, and inventory management. The goal is to reduce reliance on the database for day-to-day analytics needs by implementing a data warehouse. This warehouse will serve as a single source of truth, allowing business analysts to generate their own reports and insights using BI tools like Tableau or Power BI.
They want to solve:

- Data Silos: Sales data is stored in OLTP databases, optimized for transactional queries but not for analytics. Product metadata is in a NoSQL document database, making it difficult to integrate with relational data for reporting. Inventory data is scattered across different systems.
- Fragmented Insights: Disparate data systems prevent teams from seeing a unified view of customers, products, and sales trends. The lack of centralized data prevents self-service tools from being fully utilized, hindering proactive decisions.

Why Solve This Problem?

- Improve Agility in Decision-Making: By enabling self-service analytics, business teams can directly interact with data to make faster decisions, improving agility in responding to market trends.
- Support Growth and Scalability: RetailXpress plans to expand to new markets. A scalable analytics platform will streamline operations, making it easier to analyze data across regions
- Boost Customer Satisfaction: Unified insights into customer behavior and inventory will improve personalization and stock availability, leading to better customer experiences
- Reduce load on the operational databases

How We Plan to Solve It

- Centralized Dimensional Data Warehouse: Build a dimensional warehouse using a modern cloud-based data platform (e.g., Snowflake, BigQuery, or Azure Synapse).

- Define star schema models to simplify complex data for non-technical users
- Data Integration Pipeline: Use tools like dbt for transformation and orchestration platforms like Apache Airflow to create robust ELT pipelines:
  Extract: Pull data from OLTP and document DB systems.
  Load: Store raw data in a cloud-based data lake or staging area.
  Transform: Normalize, clean, and model data into dimensions and fact tables.
- BI Enablement: Integrate the warehouse with Tableau or Power BI for visualization. Provide reusable dashboards and self-service reporting capabilities for business users.

a. Program and Project Planning

We go through a series of interviews to promote ideas and gather feedback like why this project is required, how it will benefit the whole organization, and how it will impact ROI and give a competitive advantage. We also do an assessment of readiness in a few areas such as how strong support is from investors(optional). We create a project plan that contains cost estimates, tools, and who and what team will be part of the project).

Is there strong leadership backing for the project?
Yes

Are all stakeholders aligned on objectives?
Yes

Are data sources accessible and well-documented?
The data sources are accessible, we have an ERD for the OLTP databases but we don't know how to connect the document db yet.

Is there a need for extensive data cleaning or integration?
Yes, from the data profiling results, we will determine how much cleaning is required.

Are the existing tools sufficient for ETL/ELT and warehousing?
We currently use Bigquery and dbt for this but might need to consider a data lake, Python for connecting to document db.

Should RetailXpress invest in new platforms like Snowflake or dbt?
dbt yes, Snowflake is not needed.

Does the internal team have the required skills, or will external hiring/training be needed?
We have project managers and data engineers with the required skills.

b. Project Plan

- **Cost Estimates**
  Cloud Platform: BigQuery fees
  Tools: dbt, Tableau/Power BI/Looker subscriptions
  Development Costs: Internal team time or external contractors
  Training: Cost of upskilling stakeholders in BI tools or SQL.

- **Timeline and Milestones**
  Month 1: Requirements gathering, data profiling, and conceptual design.
  Month 2-3: Dimensional modeling, ETL/ELT pipeline design, and implementation.
  Month 4: Testing, BI integration, and deployment.
  Ongoing: Maintenance and optimization.

- **Team Composition**
  Core Team:
    - Project Manager (PM)
    - Analytics Engineer
    - Data Engineer
    - BI Developer

  Advisors:
    - Business Analysts
    - Data Governance Lead

- **Tools**
  Data Warehousing: Snowflake/BigQuery
  ETL/ELT: Apache Airflow, dbt
  Visualization: Tableau or Power BI
  Version Control: Git/GitHub

- **Risk Management**
  Potential Risks:
    - Delays in data access.
    - Misalignment of business needs.
    - Performance issues in pipelines or BI tools.

  Mitigation Strategies:
    - Weekly alignment meetings with stakeholders.
    - Early testing on small datasets to validate pipeline performance.

2. Requirements Gathering

a. Conduct stakeholder and executive interviews.

- **Define business processes**
  After the interviews, we should have a good understanding of the operational activities performed by the business. The goal here is to generate a document that captures performance metrics that will later get translated into facts in a fact table. Each fact table has a single business process.

  Requirements

  1. Sales Transactions: Recording sales across online and physical stores to understand what is being sold, what sells the most, where and what sells the least, and track revenue, discounts, and returns.
     Metrics: Total Sales Amount, Units Sold, Discounts Given, Return Rate

  2. Inventory Management: to help the business understand the current inventory level and improve stock management, what suppliers we have, and how much is being purchased.
     Monitoring stock levels, replenishments, and stockouts.Capturing costs of goods sold (COGS).
     Metrics: Stock Levels (on-hand quantity), Reorder Frequency, Shrinkage (losses due to theft/damage), Stock Turnover Rate

  3. Customer Engagement: to allow customer care and marketing teams to understand customer purchase orders, how much and when they are buying, assisting them to make better data-driven decisions. Tracking customer purchases and interactions.
     Measuring customer retention and loyalty.
     Metrics: Total Purchases per Customer, Average Order Value(AOV), Customer Lifetime Value (CLV), Loyalty Program Points Earned

  4. Marketing Campaigns: Measuring campaign performance (e.g., ROI, click-through rates). Analyzing the impact of promotions on sales.
     Metrics: Click-Through Rate (CTR), Conversion Rate, ROI of Campaigns, Incremental Revenue from Promotions

- Define data pipeline type: ELT or ETL
  ELT

- Decide on what to do with PII data
  We don't have to do anything about it.

- Overview of data type and sources(unstructured, structured)

  OLTP Databases:

  - Sales System: Tracks all online and in-store transactions.
  - Inventory System: Manages stock levels across warehouses and stores.
  - Customer Relationship Management (CRM): Stores customer profiles, interactions, and purchase history.

  Document DB Database:

  - Product Catalog: A NoSQL database containing unstructured product metadata, specifications, and images. It includes customer preferences, reviews, and engagement data.

- Ask for or create an ERD of current data
  Available in the GitHub repository under the ERD folder.

- Gather user stories

b. Data Profiling in Jupyter and dbt

We perform high-level checks of our data sources using a sample and ERD like:

- Permission level required to read data
- Meeting with source system experts to understand data flow
- Relationship between entities(dbt and querying in data warehouse)
- Missing data or data source(python, dbt and querying in data warehouse)
- Duplicates especially with PKs(dbt and querying in data warehouse)
- Quality of data to determine clean-up time
- Volume of data(scalability)
- Identify if data will be batch or streaming
- Check for any data redundancies or any extra steps
- Additional checks in python/jupyter
  - Import data, take a sample, check info, shape, head
  - Use describe() to check for outliers
  - Check for whitespaces in rows and column names
  - Check the cardinality of categorical columns with unique, nunique, value_counts, countplot to identify wrong values and incorrect data types
  - Distribution of numerical columns to check for outliers or weird values

The result of this process will be stored here and also under the data_profiling folder.

c. High-level bus matrix
   Link

d. Define ERD notation type with business
   We will use the information engineering style. It is the most widely used and was created in the early 1980s. It uses lines and graphical symbols to indicate relationships and crow's feet to represent relationship cardinality.

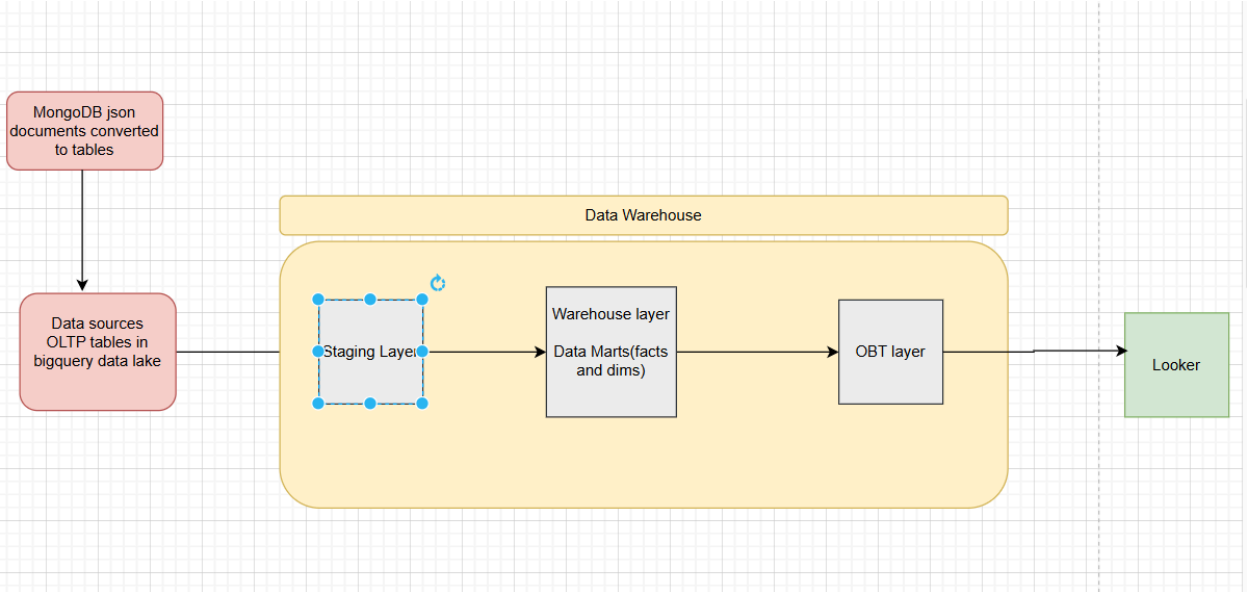
e. Create conceptual model
   A conceptual model is a high-level overview of our entities and what our system contains. The focus is to identify what data is used in a business. It is the first version of the data model and can't be used to build a database.
   Available under the ERD folder on GitHub and created with draw.io

3. Define data modeling methodology

We are going to use a hybrid of Kimball and OBT.
Kimball methodology was created by Ralph Kimball who defined a data warehouse as a copy of OLTP specifically for analysis. It is a bottom-up approach that starts by identifying business processes and requirements, data marts are built before the enterprise data warehouse. The purpose of this approach is to enable fast BI reporting. Data marts are built with the star schema, are denormalized and multiple star schemas can exist in the same model using conformed dimensions.

One Big Table(OBT) is one big denormalized table where all joins are removed and pre-aggregated. It is usually used as the reporting layer as it provides faster query performance compared to dimensional modeling.

4. Define technical architecture



Also available on GitHub under the ERD folder

5. Dimensional Modelling

a. Select high-priority business processes

All

b. Declare the grain

Fact_sales: one row per transaction
Fact_inventory_management: one row per movement
Fact_customer_engagement: one row per customer
Fact_marketing_campaigns: one row per campaign

c. Identify facts and dimensions

d. Build a detailed bus matrix

e. Source to target mapping(iterative process)

f. Create a logical model ERD

Will skip and create the physical model directly.

g. Identify facts and dimension type(additive, conformed, etc)

6. Set up Git/Github, dbt project, and database connection

Create a directory in your local environment and a repository on GitHub
Run the following in git bash:

- git init
- git remote add origin <url.git>
- git pull origin main

- git checkout -b dev

Create a dbt project and connect to bigQuery by running the following:

- dbt init <project_name>
- Select bigquery as adapter and select oauth as connection method
- Login to google cloud> gcloud init>create new configuration>pick project>
- gcloud auth application-default login
- Create a profiles.yml file under dbt project and test the connection with dbt debug(cd to dbt project folder)
- Delete example/ unders models/ and create three folders: staging, warehouse, obt
- In dbt_project.yml, add configuration for each model directory under models, enable schema and name schema

7. Build Physical Model

Available on github under ERD folder

8. ELT/ETL design and development

9. dbt modeling and layering

a. Staging and Testing

Data is extracted from the source layer in dbt and we use dbt testing(generic and singular to test data).
Singular tests: unique, not_null, relationships, accepted values
Generic tests: check for negative or zero values, for string values with elements greater than 30 or less than 3.
Run dbt run –select staging and dbt test
Results available here

b. Transformation

After investigating the failed tests, we proceed to fix the errors we found from data profiling, querying, and dbt testing. Also, add an ingestion_timestamp.

c. Warehousing

Here, we split into facts and dims using the physical model.

d. BI/OBT

We denormalize all tables, pre join facts and dims for faster reporting.

10. Validate results(sanity check, dbt test, etc)

We validate our results in two ways:

- Sanity check from domain experts by reviewing aggregated data
- Running extra dbt to flag any wrong data

11. Error Handling(pipeline failure notification)

Here, we will set warnings in several layers to ensure the pipeline remains healthy. We set warnings for:

- Failed dbt tests immediately after ingesting in the staging layer
- Any errors in the pipeline
- Failed dbt tests in aggregated/OBT layer

12. BI tool connection

This will be taken of by data analysts but the preferred tool is Looker. Any reports will be read directly from the OBT layer. Data analysts are also free to create tables from the warehouse layer.

13. Orchestration

To do: Orchestrate workflow after receiving approval to run

14. Deploy to Production

15. Maintenance

Fix any errors that are flagged and make the pipeline robust