**Big-data-warehouse design, Develop (ETL), Optimize and test (under Linux environment)**

Introduction (tools we use in this project):

**OLAP** (Online Analytical Processing) is a powerful data analysis technology used in business intelligence systems. It allows for the quick querying and analysis of multidimensional datasets, making it ideal for complex reporting tasks. OLAP organizes data into multidimensional cubes with dimensions (e.g., time, region, product) and measures (e.g., sales, revenue). Users can perform operations like slicing (focusing on specific data), dicing (viewing data from different angles), and drill-down (going into finer details). There are three types of OLAP systems: MOLAP (Multidimensional OLAP), ROLAP (Relational OLAP), and HOLAP (Hybrid OLAP). These systems support tasks such as financial reporting, sales analysis, and operational forecasting.

**Linux** is a free and open-source operating system based on Unix. Known for its stability and security, Linux powers a wide range of devices, from servers and desktops to smartphones and embedded systems. Its open-source nature allows users to view, modify, and distribute the source code. Popular for its multi-tasking and multi-user capabilities, Linux comes in various distributions like Ubuntu, **Red Hat**, and Debian, each tailored for different use cases. Linux is particularly valued in server environments, software development, and cloud infrastructure due to its flexibility, high performance, and strong security features.

**MySQL** is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and manipulate data. It is widely used for web applications, data warehousing, and enterprise software due to its scalability, speed, and reliability. MySQL supports ACID transactions and is known for its ease of use, flexibility, and strong community support. It powers popular platforms like WordPress and Facebook.

**Hive** is a data warehousing tool built on top of Apache Hadoop. It facilitates querying and managing large datasets stored in distributed storage using SQL-like language, HiveQL. Hive abstracts the complexity of Hadoop’s MapReduce framework, allowing users familiar with SQL to perform big data analytics. It’s commonly used in data processing environments for ETL tasks, reporting, and data aggregation across massive datasets.

**Shell** refers to the command-line interface used to interact with an operating system like Linux or Unix. Shell scripting enables automation of tasks through a series of commands written in a script file. It is useful for managing system processes, file manipulation, task automation, and executing programs. Popular shells include Bash and Zsh, which are extensively used by system administrators and developers for system automation and control.

**Azkaban** is an open-source workflow scheduler designed for managing and executing Hadoop jobs. It helps in creating, scheduling, and monitoring workflows in complex big data pipelines. Azkaban simplifies job execution dependencies and retries.

(Alternatives such as Apache Airflow, Luigi, Oozie, Prefect ,Argo Workflows)

Preparation before the project:

A request in building a data warehouse usually arises in a bank, IT companies or fast growing companies, where there is significant amount of data inflows to the system in daily basis (millions of them in fact tables, and a bit less on dimension tables.)

In this case, to ensure the company can operate without too much manual work and bias, operational staff have to work with technical staff to figure out a way to work efficiently, which is through data automation (data cleaning, data governance etc.) , so that operational staff can work with reasonable amount of cleaned data in daily basis.

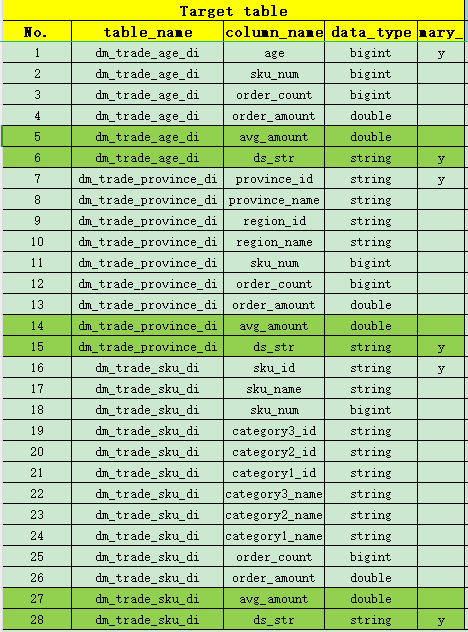
In practice, completing work within a single day is rarely feasible. Each stakeholder operates with their own perspective, and communication processes are often lengthy and repetitive, primarily to mitigate information asymmetry. This requires operational and technical teams to repeatedly confirm details to ensure alignment. When building a data warehouse, for instance, it's critical to ensure that the solution meets the users' requirements. Much like constructing a physical building, any errors in the foundational stages can lead to significant costs and setbacks later on. Thus, thorough communication and validation are essential to avoid costly mistakes.

Project stage 1:

Design: (see E-commerce\_OLAP\_project\_mapping)

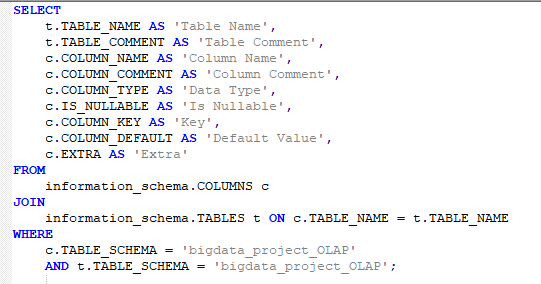
First stage of confirmation with operation staff (client) is through word documents (users explain their needs in wording formats), then designers turn them into excel mapping format for technical staff(developers/data engineers) to work on when the design stage is finished. (some designers uses visualize tools such as ‘Power-Designer’ to show the relationship between layers and tables.)

See below DM layer request: (see sales performance KPIs base on age, locations and products --3 tables.)



Then, from the original layer (OLAP) (see OLAP\_datasource\_mysql.sql) of the company’s database, designers needs to identify the relevant tables and columns which will be used to import to ods layer, where later will be import to DWD/DIM layers depending on the character of the tables (dimension or fact tables: whether full or incremental update when ETL.)

See below to extract OLAP mapping data from mysql (assuming mysql has all data for company.)



Designers also have to consider how these data will turn into the information that users requires by figuring out calculations to the KPIs.

Project stage 2&4:

Developing and optimizing stage:

For developers, given the mapping, this is the stage where it becomes technical in an offline environment; developers have to ETL the relevant data source from OLAP to ODS with tools such as azkaban, Sqoop, Shell and HiveSQL. (see sql\_hiveods.txt)

For data to be successfully extract, transform and load, developers have to consider factors such as data types, terminated fields (default \001 in mysql and \t in hive), therefore in sqoop needs to be specified.

Also, to prevent loss of data (shows null when importing to hive), table structure--data types in hive ods layers are all set to ’int’ or ‘string’, as string can contain all data types such as numbers, date etc.

Furthermore, because there are Facts and dimension tables in ods layer, we need to set algorithms using shell codes (type table names into 2 arrays: facts and dimension, and use for loop to read table names in the arrays, if table is facts, do incremental update base on the max primary key id column which is auto-increment, if they are dimensions, do full updates.)

For optimization in the ods layer, you can manually set map and reduce to 2:1 (in hadoop, more advanced technology such as spark, flink, Tez, Beam) to make sure that resources are allocated efficiently when **‘parallel computing’.**

Now that all the relevant data are moved to hive in linux, we can do the rest of ETL in Hive using create table if not exist ‘table name’ as select ‘relevant’ columns from ‘relevant tables’.

Moving onto the DWD/DIM layer of the data warehouse, for DWD (see ods\_dwd.txt) and DIM (see ods\_dim.txt), in this stage, we need to start joining the tables, so make sure do not forget any joining columns to import from ODS to DWD/DIM, also, bear in mind to include the joining columns for DIM and DWD to connect them on DWS.

For optimization in DWD, because they are fact tables and do incremental updates, which means they have millions of rows each day, therefore it make sense to make a dynamic partition for each new date, therefore when DQL it will be more efficient and takes up less cost. Also, it helps to set

Hadoop to dynamic partition restricted mode, and put a cap on every 1000 rows to make sure that each node manager (in yarn resource navigator in hadoop) is allocating the resources evenly.

For optimization in DIM, because they are dimension tables, which means it does not change in a daily basis, therefore it make sense to create a zipper table in the same layer, to keep historic records such as old sku\_no and update\_time.

In the DWS (see ods\_dim\_dwd\_dws.txt), the main point is connecting all the facts and dimension tables, so if in previous foundation layers everything is done correctly, this part becomes easy.

The optimization in hadoop is to set optimize skew joins, since there is a lot of joining tables going on; if data loading amount becomes too big, data-skew might happen( when parallel computing, yarn allocated resources unevenly, leading to one of the reduce is taking on too many data, the ETL process will get stuck at 99% for a long time.)

Finally in DM layers (see dws\_dm.txt), because the requested kpis all requires ‘group by’, therefore when making the 3 DM tables, make sure to optimize hive by setting group by skew in data =true, as it also prevents data-skew problems.

After finish writing all the scripts: use azkaban to auto-ETL process (once a day set in different times.), back in the old days technicians also uses crontab in linux to set auto-update tasks. (see azkaban.zip)

Project stage 3:

Testing:

To test the scripts in different systems (windows,Linux, Macos) in the testing environment, we need to include some meaningful sample data which will show all scenarios of the data (ideally), and check the result if it is accurate.

After testing, there will be a UAT (user acceptance test) where technical and operational staff work together to cross check the results.

Project stage 5:

Online:

After the team leader and user approves, it can go online (live) for real-life data, in this case the technical team still needs to work with operational team to ask for feedback, and it is the most inflexible stage as everyone has to standby 24/7 in case of any emergencies that could cost time and money (will be huge loss for company if the data warehouse shuts down itself for couple of hours.)