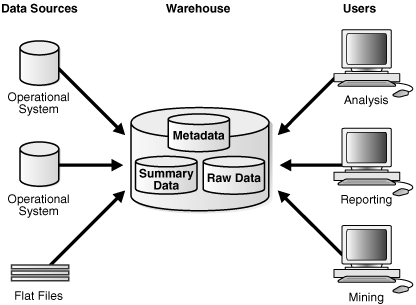
# Data Warehouse

## **What Is a Data Warehouse?**

A **data** **warehouse** (1) is a **database** designed to **enable business intelligence activities, that is,** **designed for analysis (read operations) rather than for transaction processing (write, update and delete operations)**, and (2) typically **has** data from different sources. They usually include **historical data derived from transaction data** and in this case data warehouses (3) **separate analysis workload from transaction workload**. This separation helps improve the performance of business intelligence and transaction databases.

A data warehouse as a master database may contain multiple relational databases. Within each database, schemas are defined for SQL query performance, tables can be organized inside of schemas, and data is organized into tables. When data is ingested, it is stored in various tables described by the schema. Query tools use the schema to determine which data tables to access and analyze.

In addition to relational databases as mentioned above, a data warehouse environment can include an extraction, transportation, transformation, and loading (ETL) solution, and data services such as statistical analysis, reporting and data mining. Thanks to ETL, data in a warehouse is cleaned, enriched and transformed.



*Architecture with a data warehouse*

In pipeline graphs, ELT may be represented as a seperate part outside a data warehouse for emphasis of its importance. It is important to note that defining the ETL process is a very large part of the design effort of a data warehouse. The speed and reliability of ETL operations are the foundation of the data warehouse.

## **Integrating Heterogeneous Databases**

To integrate heterogeneous databases, we have two approaches −

* Query-driven Approach
* Update-driven Approach

## Query-Driven Approach

This is the traditional approach to integrate heterogeneous databases. This approach was used to build wrappers and integrators on top of multiple heterogeneous databases. These integrators are also known as mediators.

## Process of Query-Driven Approach

* When a query is issued to a client side, a metadata dictionary translates the query into an appropriate form for individual heterogeneous sites involved.
* Now these queries are mapped and sent to the local query processor.
* The results from heterogeneous sites are integrated into a global answer set.

## Disadvantages

* Query-driven approach needs complex integration and filtering processes.
* This approach is very inefficient.
* It is very expensive for frequent queries.
* This approach is also very expensive for queries that require aggregations.

## Update-Driven Approach

This is an alternative to the traditional approach. Today's data warehouse systems follow update-driven approach rather than the traditional approach discussed earlier. In update-driven approach, the information from multiple heterogeneous sources are integrated in advance and are stored in a warehouse. This information is available for direct querying and analysis.

## Advantages

This approach has the following advantages −

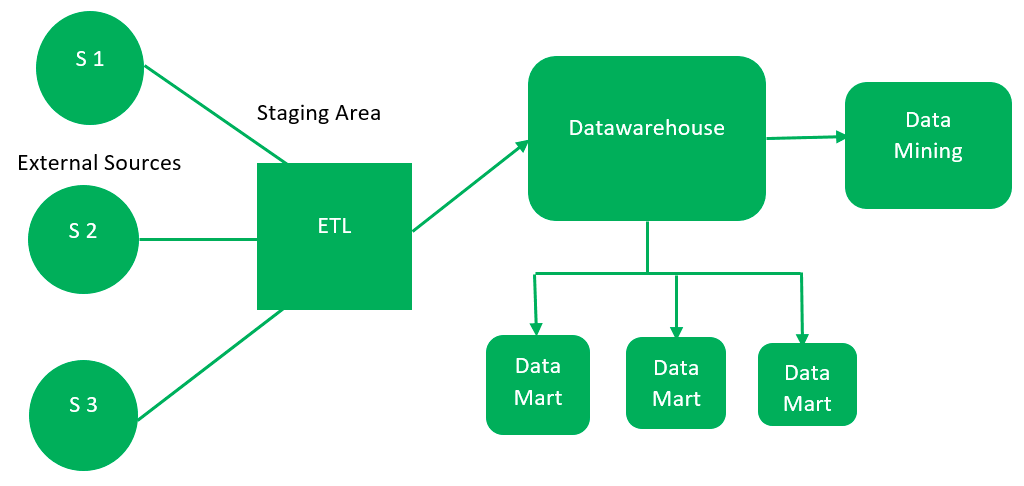
* This approach provide high performance.
* The data is copied, processed, integrated, annotated, summarized and restructured in semantic data store in advance.
* Query processing does not require an interface to process data at local sources.

Data warehouses contain relational data for downstream business applications but not all applications require this format. Data lakes come to enable diverse queries to nonrelational data.

## **Data Warehouse Architecture**

A Data Warehouse is a heterogeneous collection of data organized under a meaningful business schema. It is an information system containing historical and commutative data from multiple sources. Data Warehouses enhance data utilization in the organization towards decision support, analytics, prediction, etc… Data Warehouse Architecture is a method of defining the overall design of data communication processing and presentation for users within the Enterprise. A Data Warehouse could be architected with one of two approaches

1. Top-down approach (defined by Inmon):

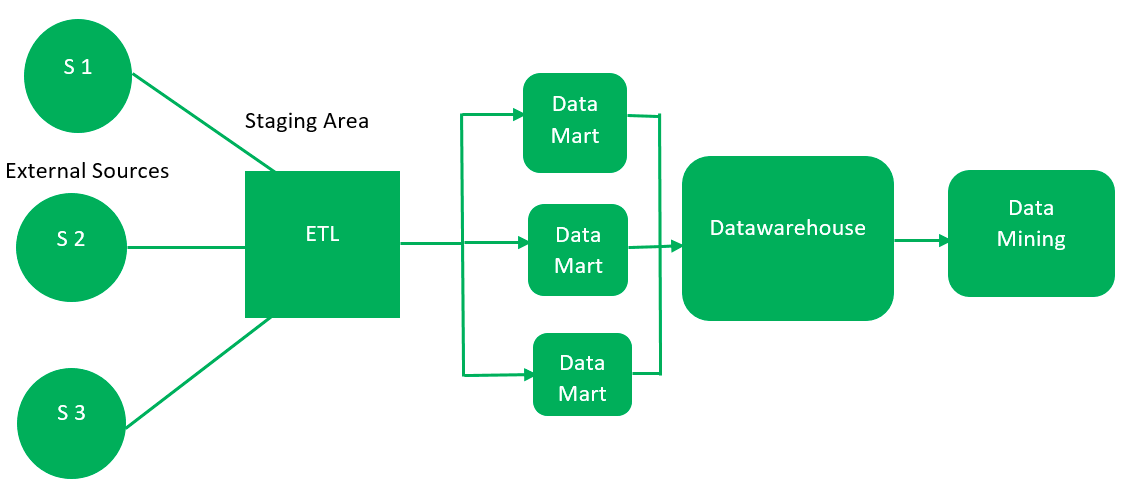


* External Sources can be structure, semi-structure and unstructured as well.
* Staging Area is where the data is ingested, validated, transformed and loaded
* Data Warehouse is where the storage happens for data and meta-data.
* Data Marts segregates organization data by function (for e.g. sales, finance)
* Data Mining is where the data is analyzed, patterns identified in the data.

Merits & de-merits of this approach

* Consistent dimensional view of Data Marts
* Resilient towards changes in business
* Cost and time are impacted designing this flexible approach

1. Bottom-up approach (defined by Kinball):



* External Sources can be structure, semi-structure and unstructured as well.
* Staging Area is where the data is ingested, validated, transformed and loaded
* In this approach Data Marts take precedence over the warehouse, are created first
* Data Marts are then integrated to form the Data Warehouse and then data is mined

Merits & de-merits of this approach

* Provides smaller, quicker user deliverables
* More open to Data Warehouse extensions compared to the first approach
* Cost and time consumed is considered better as users get to see results much faster
* Dimensional view of Data Marts lack consistency resulting in weaker Data Warehouse

Further to considering the approach the Data Warehouse design is generally three-tired as explained below

* Bottom Tier:

This tier consists of storage which usually is a relational database system. The cleansed and transformed data is loaded into this layer of the architecture. This tier acts as the staging area where data in a very raw form from different sources are pulled together for further processing.

* Middle Tier:

This tier is generally an Online Analytical Processing (OLAP) server. OLAP enables faster query and better performance for Data Warehouse operation. The Middle Tier can either be a Relation OLAP or a Multi-dimensional OLAP implementation. This provides abstract view of the stored, transformed data.

* Top Tier:

This tier is the front-end the end user interacts with. It is generally tools and API that connect and get data out from the Data Warehouse. It can run interactive user queries, report on transformed data, analyze and mine data.

## **Date Warehouse Architecture considerations**

Architecting a Data Warehouse should involve the following considerations

* Separation : Analytical processing data structures are totally different from transactional processing. Separating these layers is better to have good performance on the Data Warehouse.
* Scalability : An enterprise data ware house can scale from Giga Bytes to Peta Bytes based on functions being addressed, data sources included and desired outcome. All of these can evolve over time, a scalable architecture ensures minimal design changes.
* Extensibility : Technologies adopted by an enterprise evolve with time and as the business grows. An Extensible architecture should be able to incorporate these over the life time of the Data Warehouse without a complete redesign.
* Security : Data Warehouse provides the single version of the truth for the Enterprise. All the Enterprise data needs to be carefully secured to ensure the right users have access to the right information. This would be a critical consideration for the Data Warehouse architecture.

* Administration : In general a complicated system has a very short life span. This holds true to Administration of any system, simpler administration ensure longevity of the Data Warehouse in an Enterprise.

## **Data Warehouse Components**

ERP

External

CRM

POS

ETL Tools

Meta Data

Data Mart

Data Mart

Data Mart

Data Warehouse

Reporting / Analysis / Mining / Prediction Tools

Ad-hoc Query

Business Reports

Analytics

Models / Predictions

The Data store on a Data Warehouse forms the central repository providing single version of the truth to the organization. This store is supported by some key Data Warehouse components to make the entire environment functional, manageable and accessible.

1. Data Warehouse Datastore

The Data store is one of the critical components of the Data Warehouse environment. This can be implemented utilizing RDBMS. The implementation would be constrained by the fact that traditional RDBMS is optimized for transactional processing. For instance ad-hoc query, multi-table joins, aggregation are resource intensive and slow down performance. Alternate approaches could be considered as follows

* Deploy relational databases in parallel to allow for scalability. Parallel relational databased allow shared memory or shared nothing model on various multiprocessor configurations or massively parallel processors.
* Deploy index structures that can be used to bypass relational table scans, improve speed and overall performance.
* Deploy multidimensional databases (MDDBs) to overcome limitations which are placed because of the relational database. For instance consider deployment of OLAP as either ROLAP or MOLAP

1. ETL Tools (Acquisition, Clean-up and Transformation of Data)

ETL Tools are deployed for acquisition, clean-up, conversions, summarizations and any other changes needed to transform source data into meaningful information in the Data Warehouse. ETL is an acronym for Extract, Transform and Load.

The process generally includes:

* Anonymization of data as per regulatory stipulations
* Eliminating unwanted data from the operational data that do not add value
* Search and replace common data and definitions for data arriving from different source
* Calculating summaries and derived values
* Populate defaults in case of missing data
* Remove duplicates in data arriving from multiple datasource

ETL Tools utilize scheduled background jobs, scripts, programs, etc… to regularly update data in the Data Warehouse. The tools also value add in maintenance of Metadata while dealing with heterogeneity of external databases and data.

1. Metadata

Metadata is definition about the data being stored in the Data Warehouse. It assists in building, maintaining and managing the Data Warehouse. Metadata is critical as it specifies the source, usage, values and features of Data Warehouse data. It even defines how the data can be changed and processed. It forms and integral part of the Data Warehouse datastore.

Metadata helps to answer the following questions:

* What tables, attributes and keys the Data Warehouse contains?
* Where did the data come from?
* How many times does the data get reloaded?
* What transformations were applied with cleansing?

Metadata can be either

1. Technical Metadata

Technical Metadata contains information about the data store of the Data Warehouse used by designers and administrators in Data Warehouse operations.

1. Business Metadata

Business Metadata contains details that give end-users an easy way to understand information stored in the Data Warehouse.

1. Query Tools

Query tools allow business users to interact with the data in the Data Warehouse interactively. Ad-hoc queries enhances inputs for strategic decisions dynamically. Categories of Query Tools are as follows

* Managed Queries help business users run ad-hoc queries on the data store interactively in a very user friendly manner.
* Reporting Tools allow organization to generate regular operational reports. It also supports high volume batch jobs with printing and summarization.
* Application Development Tools are used by power users to run customized reports / analysis to satisfy organizations ad-hoc requirements dynamically.
* Data Mining Tools enable the process of discovering meaningful new correlation, patterns and trends among all the organizations data stored in the Data Warehouse. These tools can enable automation and can feed into AI / ML
* OLAP Tools enable multi-dimensional views of data allowing business users answer complex business scenario questions and also elaborate parameters involved

1. Data Marts

Data Marts are access layers specific to a business function. It is tweaked for performance and addresses specific needs of business functions like Sales, Marketing, Finance, etc… Modular Data Marts feed into the overall Data Warehouse created for the organization. Considerations for Data Mart design varies from organization to organization. Data Marts could exist in the same data store as the Data Warehouse or could exist in their own independent physical data stores.

1. Data Warehouse Bus

Data Warehouse Bus determines the flow of data in the Data Warehouse. This flow can be categories as Inflow, Upflow, Downflow, Outflow and Metaflow. While designing the Data Warehouse Bus, shared dimensions and facts across Data Marts need to be considered.

## **Data Warehousing Principles**

1. Load Performance

Data warehouses require increase loading of new data periodically basis within narrow time windows; performance on the load process should be measured in hundreds of millions of rows and gigabytes per hour and must not artificially constrain the volume of data business.

1. Load Processing

Many phases must be taken to load new or update data into the data warehouse, including data conversion, filtering, reformatting, indexing, and metadata update.

1. Data Quality Management

Fact-based management demands the highest data quality. The warehouse ensures local consistency, global consistency, and referential integrity despite "dirty" sources and massive database size.

1. Query Performance

Fact-based management must not be slowed by the performance of the data warehouse RDBMS; large, complex queries must be complete in seconds, not days.

1. Terabyte Scalability

Data warehouse sizes are growing at astonishing rates. Today these size from a few to hundreds of gigabytes and terabyte-sized data warehouses.

## **Data Warehouse Architecture Best Practices**

The following best practices can be considered while designing a Data Warehouse Architecture

1. Optimize Data Warehouse models for information retrieval utilizing dimensional, de-normalized or hybrid approach that suit the needs at hand
2. Choose the appropriate design approach as top down or bottom up suitable to the organizations business needs
3. Data need to bee processed quickly and accurately. Data needs to be consolidated into a single version of the truth
4. Design data acquisition and cleansing process such that the Metadata is shared between components of the Data Warehouse
5. Consider implementing ODS model when information retrieval need is near bottom of data abstraction pyramid or when there are multiple operational sources
6. Data model should integrated further to consolidation with consideration for 3NF data model. This would be ideal for acquiring ETL and Data cleansing tools

<place holder>

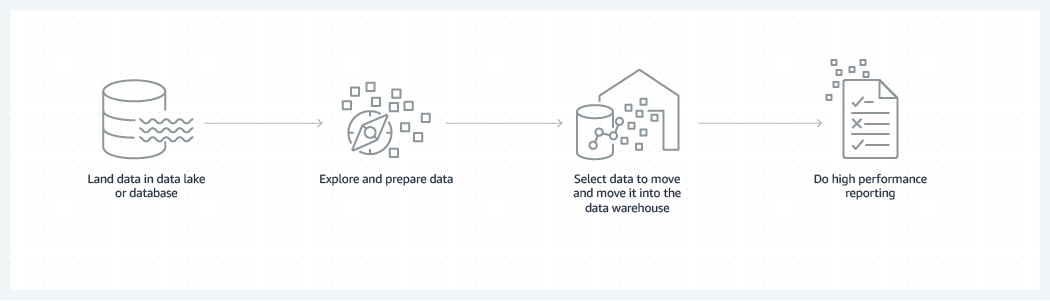
1. <place holder>
2. <place holder>
3. <place holder>
4. <place holder>

## **What is a data lake?**

A data lake is a centralized repository that stores all **structured and unstructured** data at any scale. You can store your data without having to first structure the data, and can run **different types of analytics—**from dashboards and visualizations to big data processing, real-time analytics, and machine learning and full-text search.

The structure of the data or **schema is not defined when data is captured.** This means you can store all of your data without careful design or the need to know what questions you might answer in the future. Data Lakes allow you to import any amount of data that can come in **real-time** in its original format, **saving time of defining schema**.

A data lake can be an upstream database for a data warehouse, as is seen below.

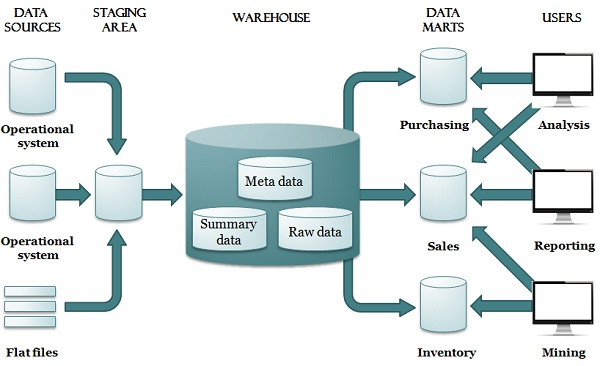


Data warehouse vs Data lake

|  |  |  |
| --- | --- | --- |
| **Characteristics** | **Data Warehouse** | **Data Lake** |
| **Data** | **Relational** from transactional systems, operational databases, and line of business applications | **Non-relational and relational** from IoT devices, websites, mobile apps, social media, and corporate applications |
| **Schema** | Designed prior to the data warehousing implementation (schema-on-write) | Written at the time of analysis (schema-on-read) |
| **Data Quality** | Transformed data | Transformed or raw data |
| **Users** | **Business analysts** (with SQL queries) | Data scientists, Data developers, and Business analysts (using transformed data) with various tools such as Apache Hadoop, Presto, and Apache Spark |
| **Analytics** | Batch reporting, BI and visualizations | Machine Learning, Predictive analytics, data discovery and profiling |

## **What is a data mart?**

A data mart serves the same role as a data warehouse, but it is intentionally limited in scope. It may serve one **particular department or business unit** like marketing and sales. Data marts **may be** a subset of a data warehouse that is highly curated for a specific end user. The following graph illustrates this possible case.



*A possible architecture with a data warehouse and data marts*

### Data warehouse vs data mart

|  |  |  |
| --- | --- | --- |
| **Characteristics** | **Data Warehouse** | **Data Mart** |
| Scope | Centralized, multiple subject areas integrated together | Decentralized, specific subject area |
| Users | Organization-wide | A single community or department |
| Data source | Many sources | A **single or a few sources**, or a portion of data **already** collected in a data warehouse |

### Data warehouse vs database

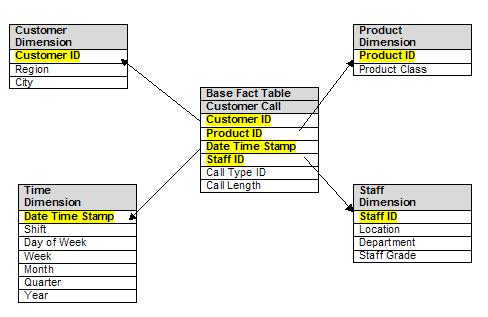
|  |  |  |
| --- | --- | --- |
| **Characteristics** | **Data Warehouse** | **Transactional Database** |
| Suitable workloads | Analytics, reporting, big data | **Transaction** processing |
| Data source | Data collected and **transformed from many sources** | Data captured **as-is from a single source**, such as a transactional system |
| Data capture | **Bulk write** operations typically on a predetermined batch schedule | Optimized for **continuous write** operations as new data is available |
| Data storage | Optimized for **high-speed query performance** | Optimized for high throughout **write operations** |

SCHEMAS

Schema is a logical description of the entire database. It includes the name and description of records of all record types including all associated data-items and aggregates. Much like a database, a data warehouse also requires to maintain a schema. A database uses relational model, while a data warehouse uses Star, Snowflake, and Fact Constellation schema. In this chapter, we will discuss the schemas used in a data warehouse.

**Star schema**

A star schema is a database organizational structure optimized for use in a data warehouse or business intelligence that uses a single large fact table and one or more smaller dimensional tables. It is called a star schema because the fact table sits at the center of the logical diagram, and the small dimensional tables branch off to form the points of the star.



A fact table sits at the center of a star schema database, and **each star schema database only has a single fact table**. The fact table *contains* the specific **quantifiable** data to be analyzed, such as sales figures.

The fact table stores two types of information: numeric values and dimension attribute values. Using a sales database as an example:

* **Numeric value cells** are unique to each row or data point and do not correlate or relate to data stored in other rows. These might be facts about a transaction, such as an order ID, total amount, net profit, order quantity or exact time.
* **The dimension attribute values** do not directly store data, but they store the foreign key value for a row in a related dimensional table. Many rows in the fact table will reference this type of information. So, for example, it might store the sales employee ID, a date value, a product ID or a branch office ID.

Dimension tables **store supporting information** to the fact table. Each star schema database has at least one dimension table. Each dimension table will relate to a column in the fact table with a dimension value, and will store **additional information about that value**.

* The **employee dimension table** may use the employee ID as a key value and can contain information such as the employee's name, gender, address or phone number.
* A **product dimension table** may store information such as the product name, manufacture cost, color or first date on market.

### Benefits of the Star Schema

* It is extremely simple to understand and build.
* No need for complex joins when querying data.
* Accessing data is faster (because the engine doesn’t have to join various tables to generate results).
* Simpler to derive business insights.
* Works well with certain tools for analytics, in particular, with OLAP systems that can create OLAP cubes from data stored using star schema.

### Disadvantages of the Star Schema

* Denormalized data can cause integrity issues. This means some data can turn out to be inconsistent at times.
* Maintenance may appear simple at the beginning, but the larger data warehouse you need to maintain, the harder it becomes (due to data redundancy).
* It requires a lot more disk space than snowflake schema to store the same amount of data.
* Many-to-many relationships are not supported.
* Limited possibilities for complex queries development.

**Snowflake schema**

The snowflake schema is an extension of a star schema. The main difference is that in this architecture, each reference table can be linked to one or more reference tables as well. The aim is to normalize the data.

|  |  |
| --- | --- |
| Star schema | Snowflake schema |
| Start Schema | Snowflake Schema |
|  | |

### Benefits of the Snowflake Schema

* Uses less disk space because data is normalized and there is minimal data redundancy.
* Offers protection from data integrity issues.
* Maintenance is simple due to a smaller risk of data integrity violations and low level of data redundancy.
* It is possible to use complex queries that don’t work with a star schema. This means more space for powerful analytics.
* Supports many-to-many relationships.

### Disadvantages of the Snowflake Schema

* Harder to design compared to a star schema.
* Maintenance can be more complex due to a large number of different tables in the data warehouse.
* Queries can be very complex, including many levels of joins between many tables.
* Queries can be slower in some cases because many joins should be done to produce final output.
* More specific skills are needed for working with data stored using snowflake schema.

## **Fact Constellation Schema**

A fact constellation (or galaxy schema) has multiple fact tables.

* The following diagram shows one more fact table, namely shipping, in addition to the sales table.

|  |  |
| --- | --- |
| Star schema | Fact constellation schema |
| Start Schema | Fact Constellation Schema |

**Partitioning**

Partitioning is done to enhance performance and facilitate easy management of data.

## **Why is it Necessary to Partition?**

### **For Easy Management**

The fact table in a data warehouse can grow up to hundreds of gigabytes in size. This huge size of fact table is very hard to manage as a single entity. Therefore it needs partitioning.

### **To Assist Backup/Recovery**

If we do not partition the fact table, then we have to load the complete fact table with all the data. Partitioning allows us to load only as much data as is required on a regular basis. It reduces the time to load and also enhances the performance of the system.

### **To Enhance Performance**

By partitioning the fact table into sets of data, the query procedures can be enhanced. Query performance is enhanced because now the query scans only those partitions that are relevant. It does not have to scan the whole data.

**Partitioning Strategies**

### **Partitioning by Time into Equal Segments**

In this partitioning strategy, the fact table is partitioned on the basis of time period. Here each time period represents a significant retention period within the business. For example, if the user queries for **month to date data** then it is appropriate to partition the data into monthly segments. We can reuse the partitioned tables by removing the data in them.

### **Partition by Time into Different-sized Segments**

This kind of partition is done where the aged data is accessed infrequently. It is implemented as a set of small partitions for relatively current data, larger partition for inactive data.

### **Partition on a Different Dimension**

The fact table can also be partitioned on the basis of dimensions other than time such as product group, region, supplier, or any other dimension. Let's have an example.

Suppose a market function has been structured into distinct regional departments like on a **state by state** basis. If each region wants to query on information captured within its region, it would prove to be more effective to partition the fact table into regional partitions. This will cause the queries to speed up because it does not require to scan information that is not relevant.

### **Partition by Size of Table**

When there are no clear basis for partitioning the fact table on any dimension, then we should **partition the fact table on the basis of their size.** We can set the predetermined size as a critical point. When the table exceeds the predetermined size, a new table partition is created.

### **Partition by normalization**

Normalization is the standard relational method of database organization. In this method, the rows are collapsed into a single row, hence it reduces space. Take a look at the following tables that show how normalization is performed.

Table before Normalization

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Product\_id** | **Qty** | **Value** | **sales\_date** | **Store\_id** | **Store\_name** | **Location** | **Region** |
| 30 | 5 | 3.67 | 3-Aug-13 | 16 | sunny | Bangalore | S |
| 35 | 4 | 5.33 | 3-Sep-13 | 16 | sunny | Bangalore | S |
| 40 | 5 | 2.50 | 3-Sep-13 | 64 | san | Mumbai | W |
| 45 | 7 | 5.66 | 3-Sep-13 | 16 | sunny | Bangalore | S |

Table after Normalization

|  |  |  |  |
| --- | --- | --- | --- |
| **Store\_id** | **Store\_name** | **Location** | **Region** |
| 16 | sunny | Bangalore | W |
| 64 | san | Mumbai | S |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product\_id** | **Quantity** | **Value** | **sales\_date** | **Store\_id** |
| 30 | 5 | 3.67 | 3-Aug-13 | 16 |
| 35 | 4 | 5.33 | 3-Sep-13 | 16 |
| 40 | 5 | 2.50 | 3-Sep-13 | 64 |
| 45 | 7 | 5.66 | 3-Sep-13 | 16 |

**Metadata**

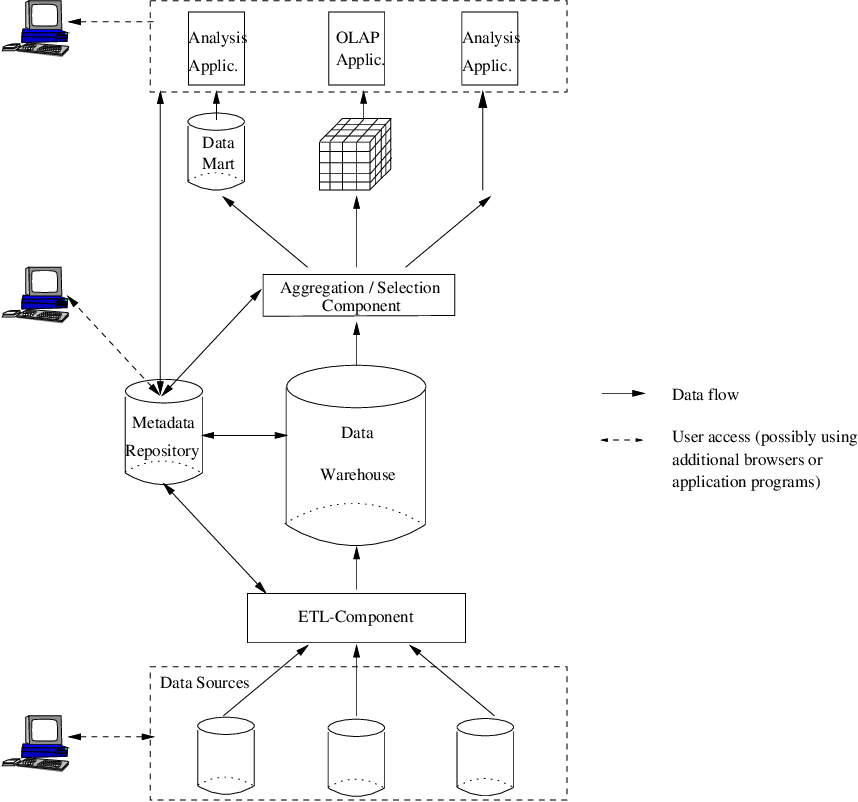
Metadata is data about data; or the **description** of the structure, content, keys, indexes, etc., of data.

Here are several examples.

* Metadata for a document may contain the document created date, last modified date, it’s size, author, description, etc.
* Metadata for ETL includes the job name, source tables/files, target tables/files, and frequency.
* Metadata associated with data management defines the data store in the Data Warehouse. Every object in the database needs to be described including the data in each table, index, and view, and any associated constraints.

The Role of Metadata in the Data Warehouse

* *Consistency of definitions*: One department refers to “revenues,” another to “sales.” Are they talking about the same activity? One unit talks about “customers,” another about “users” or “clients.” Are these different classifications or different terms for the same classification? Effective metadata management can ensure that **the same data “language” applies throughout the organization**. Business users can easily understand the full meaning of data. With understanding comes eased communication, and an overall improved process. *The business metadata primarily supports business end users who do not have a technical background, and cannot use the technical metadata to determine what information is stored inside the Data Warehouse. Technical metadata primarily supports technical staff that must implement and deploy the Data Warehouse.*
* *Clarity of relationships*: Meta data management illuminates the associations among all components of the warehouse: business rules, tables, columns, transformations, and user views of the data. By clarifying relationships throughout the Data Warehouse environment, managed Meta data enables warehouse managers to **see the bigger picture—to fully understand the meanings of the data assets, and to accurately predict and manage the impact of changes to the data warehouse**.
* *Availability of information*: Meta data exists “behind the scenes,” revealing the **origin of data, who defined it, when it was modified**, and much more.
* Resource discovery: Metadata serves the same functions in resource discovery as good cataloging does by: 1. Allowing resources to be found by relevant criteria; 2. Identifying resources; 3. Bringing similar resources together; 4. Giving location information.

*A data warehouse system at buildtime and usetime*

Metadata has been identied as a key success factor in data warehouse projects. It captures all kinds of information necessary to extract, transform and load data from source systems into the data warehouse, and afterwards to use and interpret the data warehouse contents.

The generation and management of metadata serves two purposes: (1) to minimize the efforts for development and administration of a data warehouse and (2) to improve the extraction of information from it.

**Online Analytical Processing (OLAP)**

OLAP is a category of software that allows users to analyze information from multiple database systems at the same time. Ienables end-users to perform ad hoc analysis of data in multiple dimensions, thereby providing the insight and understanding they need for better decision making.

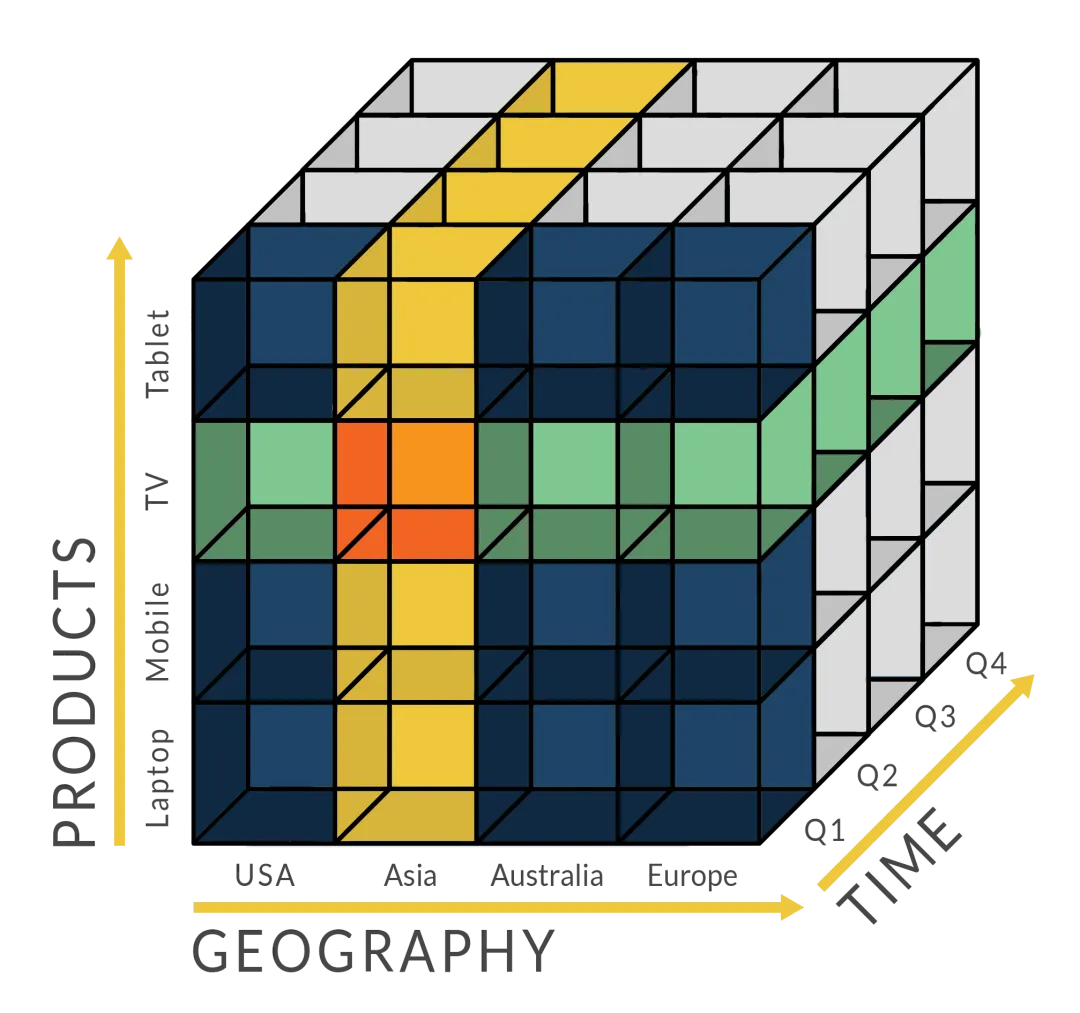
Analysts frequently need to group, aggregate and join data. These operations in relational databases are resource intensive. With OLAP data can be pre-calculated and pre-aggregated, making analysis faster.

Business is a multidimensional activity and businesses are run on decisions based on multiple dimensions. Businesses track their activities by considering many variables. When these variables are tracked on a spreadsheet, they are set on axes (x and y) where each axis represents a logical grouping of variables in a category.

For example, sales in units or dollars may be tracked over one year’s time, by month, where the sales measures might logically be displayed on the y axis and the months might occupy the x axis (i.e., sales measures are rows and months are columns).

To analyze and report on the health of a business and plan future activity, many variable groups or parameters must be tracked on a continuous basis—which is beyond the scope of any number of linked spreadsheets. These variable groups or parameters are called Dimensions.

Unlike relational databases, OLAP tools do not store individual transaction records in two-dimensional, row-by-column format, like a worksheet, but instead use multidimensional database structures—known as *Cubes* —to store arrays of consolidated information.  The OLAP cube is a data structure optimized for very quick data analysis. The data and formulas are stored in an optimized multidimensional database, while views of the data are created on demand.



Analysts can take any view, or Slice, of a Cube to produce a worksheet-like view of points of interest. Rather than simply working with two dimensions (standard spreadsheet) or three dimensions (for example, a workbook with tabs of the same report, by one variables), companies have many dimensions to track—-for example, a business that distributes goods from more than a single facility will have at least the following dimensions to consider: Accounts, Locations, Periods, Salespeople and Products. These dimensions comprise a base for the company’s planning, analysis and reporting activities. Together they represent the “whole” business picture, providing the foundation for all business planning, analysis and reporting activities.

**Types of OLAP**

OLAP can be categorized into specific types based on how they function as described below

1. Relational OLAP

ROLAP stands for Relational Online Analytical Processing. ROLAP stores data in columns and rows (also known as relational tables) and retrieves the information on demand through user submitted queries. A ROLAP database can be accessed through complex SQL queries to calculate information. ROLAP can handle large data volumes, but the larger the data, the slower the processing times.

Because queries are made on-demand, ROLAP does not require the storage and pre-computation of information. However, the disadvantage of ROLAP implementations are the potential performance constraints and scalability limitations that result from large and inefficient join operations between large tables. Examples of popular ROLAP products include Metacube by Stanford Technology Group, Red Brick Warehouse by Red Brick Systems, and AXSYS Suite by Information Advantage.

2. Multidimensional OLAP

MOLAP stands for Multidimensional Online Analytical Processing. MOLAP uses a multidimensional cube that accesses stored data through various combinations. Data is pre-computed, pre-summarized, and stored (a difference from ROLAP, where queries are served on-demand).

A multicube approach has proved successful in MOLAP products. In this approach, a series of dense, small, precalculated cubes make up a hypercube. Tools that incorporate MOLAP include Oracle Essbase, IBM Cognos, and Apache Kylin.

Its simple interface makes MOLAP easy to use, even for inexperienced users. Its speedy data retrieval makes it the best for “slicing and dicing” operations. One major disadvantage of MOLAP is that it is less scalable than ROLAP, as it can handle a limited amount of data.

3. Hybrid OLAP

HOLAP stands for Hybrid Online Analytical Processing. As the name suggests, the HOLAP storage mode connects attributes of both MOLAP and ROLAP. Since HOLAP involves storing part of your data in a ROLAP store and another part in a MOLAP store, developers get the benefits of both.

With this use of the two OLAPs, the data is stored in both multidimensional databases and relational databases. The decision to access one of the databases depends on which is most appropriate for the requested processing application or type. This setup allows much more flexibility for handling data. For theoretical processing, the data is stored in a multidimensional database. For heavy processing, the data is stored in a relational database.

Microsoft Analysis Services and SAP AG BI Accelerator are products that run off HOLAP.

How does it work?

A Data warehouse would extract information from multiple data sources and formats like text files, excel sheet, multimedia files, etc.

The extracted data is cleaned and transformed. Data is loaded into an OLAP server (or OLAP cube) where information is pre-calculated in advance for further analysis.

# Data Warehouse Pipeline: Basic Concepts & Roadmap

## Description

Building a data warehouse pipeline can be complex sometimes. If you are starting in this world, you will soon realize there is no *right or wrong way*to do it. **It always depends on your needs**.

Yet, there a couple of basic processes you should put in place when building a data pipeline to improve its operability and performance.

Intend to share you with a roadmap that can help as a guide when building a data warehouse pipeline.

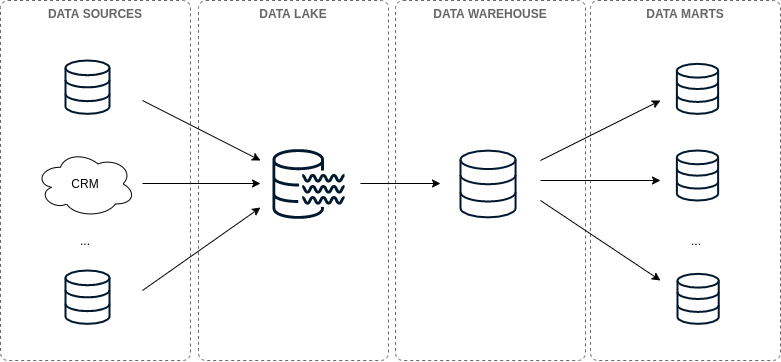
This roadmap is intended to help people to implement [DataOps](https://medium.com/data-ops/what-is-dataops-ten-most-common-questions-ffc09c09c921) when building a data warehouse pipeline through a set of processes.

*In the roadmap section we talk about five processes that should be implemented to improve your data pipeline operability and performance — Orchestration, Monitoring, Version Control, CI/CD, and Configuration Management.*

*Some of the data warehousing terminology — e.g., data lakes, data warehouses, batching, streaming, ETL, ELT, and so on.*

## Basic concepts of Architecture

The basic architecture of a data warehouse pipeline can be split into four parts: ***data* *sources, data lake, data warehouse,*and*data marts***.



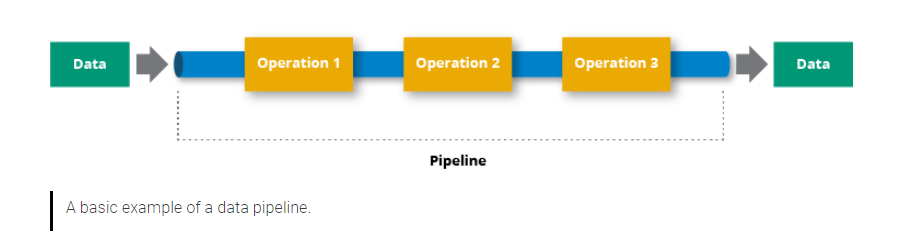
### Data Warehouse Pipeline Architecture

* **Data sources:**The data coming from the business operation. This data comes from production databases, CRMs, APIs, Flat file and so on.
* **Data Lake:** “*A Data Lake is a storage repository of* ***multiple sources of raw data*** *in a single location.”*The data can be found in several formats. Usually, the data can be usually unstructured and a little bit messy at this stage of the data pipeline.
* **Data Warehouse:** “*A Data Warehouse (also commonly called a single source of truth) is a* ***clean, organized****, single representation of your data. Sometimes it’s a completely different data source, but increasingly it’s structured virtually, as a schema of views on top of an existing lake.*”
* **Data Marts:***“A Data Mart is a* ***filtered*** *(and sometimes aggregated) subsection of a Data Warehouse to make it easier for a particular group to query data. It provides a* ***smaller*** *schema with only the relevant tables for the group.”*

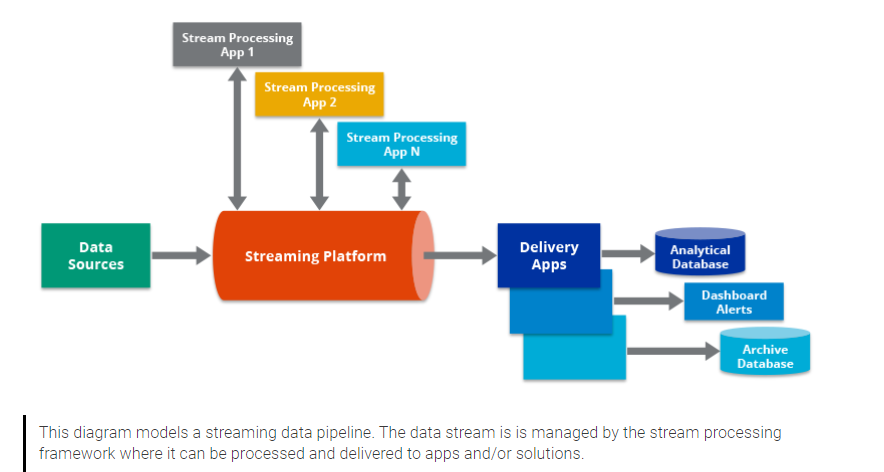
Together, these four parts represent the basic architecture of a data pipeline. The data is moved from data sources down to the data warehouse. This can be done in performing via *batch*or *stream*processing.

### Batch vs Streaming vs Lambda

Batch processing is based on loading the data in batches. This means, your data is loaded once per day, hour, and so on.

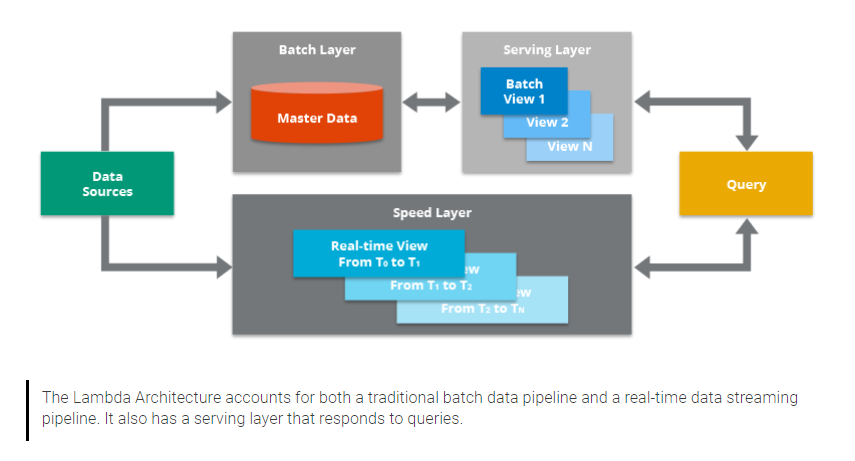


Stream processing is based on loading the data as it arrives. This is usually done using a [Pub/Sub](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) system. So, in this way, you can load your data to the data warehouse *nearly in real-time.*



These two types of processing are not mutually exclusive. They may coexist in a data pipeline — see [Lambda and Kappa architectures](https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb) for more info. Particularly,we’ll focus on the**batch approach**in this post.

Lambda Architecture combines batch and streaming pipeline into one architecture.



### ETL vs ELT processes

Batch processing implies moving the data from point A to point B. Processes allowing for doing such tasks are known as **ETL** processes — Extract, Load, and Transform.

These processes are based on extracting data from sources, transforming, and loading it to a data lake or data warehouse.

Although, in recent years, another approach has been introduced: the **ELT**approach.

**ETL** is the legacy way, where transformations of your data happen on the way to the lake.

**ELT** is the modern approach, where the transformation step is saved until after the data is in the lake. The transformations really happen when moving from the Data Lake to the Data Warehouse.

*ETL was developed when there were no data lakes; the staging area for the data that was being transformed acted as a virtual data lake. Now that storage and compute is relatively cheap, we can have an actual data lake and a virtual data warehouse built on top of it.*

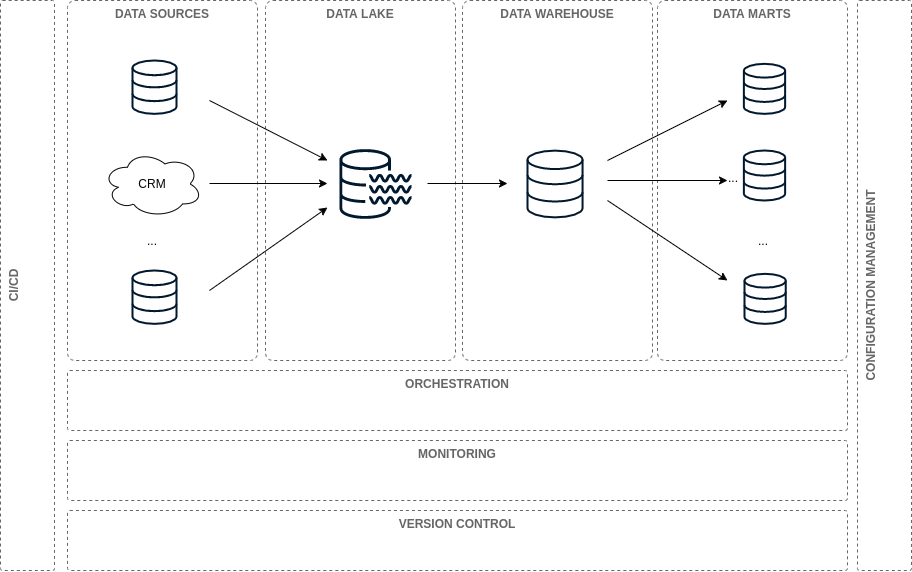
**ELT** approach is preferred over **ETL**since it fosters best practices making easier data warehousing processes — *e.g.,*highly reproducible processes,simplification of the data pipeline architecture, and so on.

## Data Pipeline Roadmap

There are **five processes** we recommend you should put in place to improve your data pipeline operability and performance.

These processes are *Orchestration*, *Monitoring*, *Version Control*,*CI/CD*, and *Configuration Management.*

Such processes are defined based on the [DataOps philosophy](https://medium.com/data-ops/what-is-dataops-ten-most-common-questions-ffc09c09c921), which “*is a collection of technical practices, workflows, cultural norms, and architectural patterns*” enabling to reduce [technical debt](https://medium.com/data-ops/a-great-model-is-not-enough-deploying-ai-without-technical-debt-70e3d5fecfd3) in the data pipeline — among other things.



We all have written CRON jobs for orchestrating data processes at some point in our lives.

When data is in the right place and it arrives at the expected time, everything runs smoothly. But, there is a problem. Things always go wrong at some point. When it happens everything is chaos.

Adopting better practices for handling data orchestration is necessary — e.g., retry policies, [data orchestration process generalization](https://towardsdatascience.com/generalizing-data-load-processes-with-airflow-a4931788a61f), process automation, task dependency management, and so on.

As your pipeline grows, so does the complexity of your processes. CRON jobs fall short for orchestrating a whole data pipeline. This is where [Workflow management systems](https://en.wikipedia.org/wiki/Workflow_management_system)(WMS) step in. They are systems oriented [to support robust operations](https://medium.com/tech-quizlet/going-with-the-flow-how-quizlet-uses-apache-airflow-to-execute-complex-data-processing-pipelines-1ca546f8cc68) allowing for orchestrating your data pipeline.

Some of the WMS used in the data industry are[Apache Airflow](https://airflow.apache.org/), [Apache Luigi](https://github.com/spotify/luigi), and [Azkaban](https://azkaban.github.io/).

### Data Pipeline (Question before going to Data Pipeline)

A data pipeline is a series of data processing steps. If the data is not currently loaded into the data platform, then it is ingested at the beginning of the pipeline. Then there are a series of steps in which each step delivers an output that is the input to the next step. To suitable design and implement and follow [5Vs](https://www.geeksforgeeks.org/5-vs-of-big-data/) (Volume, Velocity, Variety, Veracity, Value) in Big data Design, there are some consideration need to be addressed first:

1. Which of data processing type need to be handle (streaming or batching)
2. What rate of data do you expect (daily, hourly data ? 90% or 95% captured data)
3. What is the data structure you need to handle as well as data sources type ?
4. Is the data being generated in Cloud or On-premises ?
5. Where to you implement data pipeline
6. What is specific technologies in which your team needs to have ?
7. Choosing the technical suite stack ?
8. Etc etc

### Monitoring

Have you been in that position where all dashboards are down and business users come looking after for you to fix them? or maybe your DW is down you don’t know? That’s why you should always monitor your data pipeline!

Monitoring should be a [proactive process, not just reactive](https://www.infoworld.com/article/3231666/how-devops-changes-monitoring.html). So, if your dashboard or DW is down, you should know it before business users come looking out for you.

To do so, you should put in place monitoring systems. They run continuously to give you realtime insights about the health of your data pipeline.

Some tools used for monitoring are [Grafana](https://grafana.com/), [Datadog](https://www.datadoghq.com/), and [Prometheus](https://prometheus.io/).

### CI/CD

Does updating changes in your data pipeline involve a lot of manual and error-prone processes to deploy them to production? If so, CI/CD is a solution for you.

CI/CD stands for *Continuous Integration*and *Continous Deployment*. The goal of CI is “*to establish a consistent and automated way to build, package, and test applications*”. On the other hand, CD “*picks up where continuous integration ends. CD automates the delivery of applications to selected infrastructure environments.*” — more info [here](https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html).

CI/CD allows you to push changes to your data pipeline in an automated way. Also, it will reduce manual and error-prone work.

Some tools used for CI/CD are [Jenkins](https://www.jenkins.io/), [GitlabCI](https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/), [Codeship](https://codeship.com/), and [Travis](https://travis-ci.org/).

### Configuration Management

So…Imagine your data pipeline infrastructure breaks down for any reason. For example, you need to deploy again the whole orchestration pipeline infrastructure. How you do it?

That’s where configuration management comes in. [Configuration management](https://www.netapp.com/us/info/what-is-configuration-management.aspx) “*deals with the state of any given infrastructure or software system at any given time.*” It fosters practices like [Infrastructure as Code](https://en.wikipedia.org/wiki/Infrastructure_as_code). Additionally, it deals with the whole configuration of the infrastructure — more info [here](https://medium.com/formcept/configuration-management-and-continuous-deployment-cd0892dce998).

Some tools used for Configuration Management are [Ansible](https://www.ansible.com/), [Puppet](https://puppet.com/), and [Terraform](https://www.terraform.io/).

### Version control

Finally, one of the most known processes in the software industry: version control. We all have had problems when version control practices are not in place.

Version control manages changes in artifacts. It is an [essential process](https://www.perforce.com/blog/vcs/what-is-version-control) for tracking changes in the code, iterative development, and team collaboration.

Some tools used for Version Control are [Github](https://github.com/), [GitLab](https://about.gitlab.com/), [Docker Hub](https://hub.docker.com/), and [DVC](https://dvc.org/).

## **Summary**

1. Data Warehouse is an information system that contains historical and commutative data from single or multiple sources.
2. A Data Warehouse is subject oriented as it offers information regarding subject instead of organization's ongoing operations.
3. In Data Warehouse, integration means the establishment of a common unit of measure for all similar data from the different Databases
4. Data Warehouse is non-volatile in the sense that the previous data is not erased when new data is entered in it.
5. A Data Warehouse is Time-variant as the data in a Data Warehouse has high shelf life.
6. There are 5 components of Data Warehouse Architecture namely Datastore, ETL Tools, Metadata, Query Tools, Data Marts
7. The Query tools can be categorized as Query and reporting, tools, Application Development tools, Data mining tools, OLAP tools
8. The data sourcing, transformation, and migration tools are used for performing all the conversions and summarizations.
9. In the Data Warehouse Architecture, meta-data plays an important role as it specifies the source, usage, values, and features of data warehouse data.

## **Data Warehousing Application Examples**

1. Airlines use Data Warehousing for management operation like crew assignment, route analysis, frequent flyer programs, discount schemes
2. Banking sector use Data Warehousing to manage resources, financial portfolios, mitigate risks, predict financial performance
3. Healthcare sector use Data Warehouse to strategize and predict outcomes, create patient medical condition profiles, identify patterns for better medical services, design financial support based on demography
4. Retail sector use Data Warehouse to identify customer buying patterns, spending habits, demography related marketing effectiveness, pricing structure
5. Telecommunication sector use Data Warehouse for product promotions, planning future product road maps, sales decision, distribution patterns