Enterprise Decision Support System Architectures

Joshua Cook

**Bill Inmon's Theoretical Data Warehouse Architecture**

Bill Inmon is considered by many to be the father of the data warehouse and the EDSS.

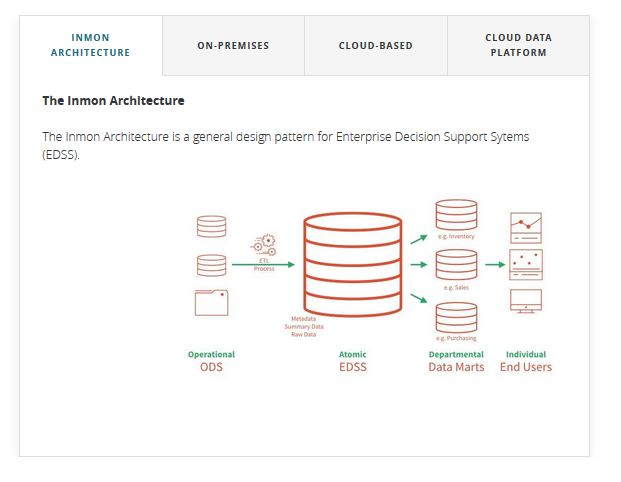
Inmon developed a theoretical architecture for the EDSS including all four layers, from operational to individual:

* An ETL process loads **operational** data into an EDSS (**atomic** data)
* Another process loads **atomic** data into data marts at the **departmental** level
* End users and other applications access **departmental** data at the **individual** level

**Reviewing EDSS Architectures**

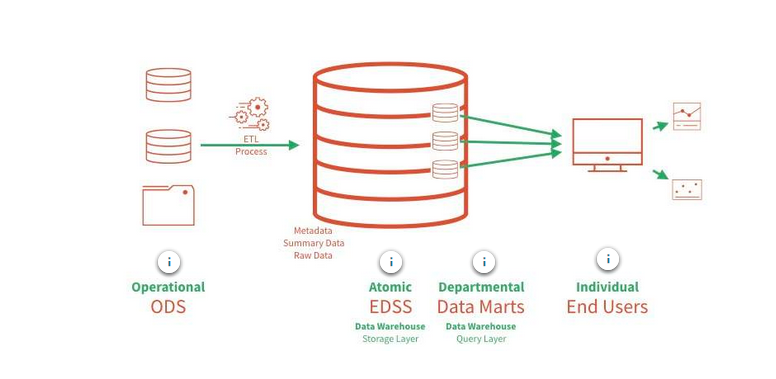
In this lesson, we review the most common architecture patterns used for the modern EDSS.

Delta Lake is the next-generation implementation of Bill Inmon’s vision and iterates on his and others’ work in data warehousing.



**The on-premises data warehouse system**

The on-premises data warehouse (ODW) system was the original design pattern used in building an EDSS.



**Benefits of on-premises data warehouses**

Highly-optimized

–

ODWs typically involve highly optimized and tuned hardware and software.

ODW system architecture tightly couples hardware and software.

This coupling provides for best-in-class concurrent query performance.

Massively parallel processing systems

–

ODWs typically leverage massively parallel processing (MPP) systems which allow for extremely fast ODW reads and writes.

Well-established technology

–

ODWs are also tried and true and are supported by decades of research and development, ensuring their reliability.

**Challenges with on-premises data warehouses**

Not cloud-native

–

Since ODW's are not cloud-native, leveraging data for use anywhere other than within the ODW system requires potentially long-running queries or ETL processes.

Not elastic

–

ODWs are not elastic. This means that resources cannot easily be scaled up or down to meet expanding or contracting workload demands.

In an ODW, scaling up often requires months of expensive planning and development by an engineering team.

Data duplication

–

Performance benefits often require data replication/duplication, which goes against the principle of having a single source of truth.

High data gravity

–

Data gravity refers to how difficult it is to move data from a given location.

Data kept in an ODW has high data gravity, meaning that it is difficult to move.

Data is kept in closed, proprietary formats on physical media.

If an organization is using an ODW managed by a vendor, their data is effectively locked in to that particular vendor.

Structured data only

–

An ODW is, by nature, a structured data store, meaning that it can only ingest structured data.

An ODW does not provide the flexibility to take a NoSQL approach to data processing.  
  
In recent years, a NoSQL (Not Only SQL) approach to data processing has been found to be reliable and valuable.

Expense

–

An ODW is very expensive to build and maintain.

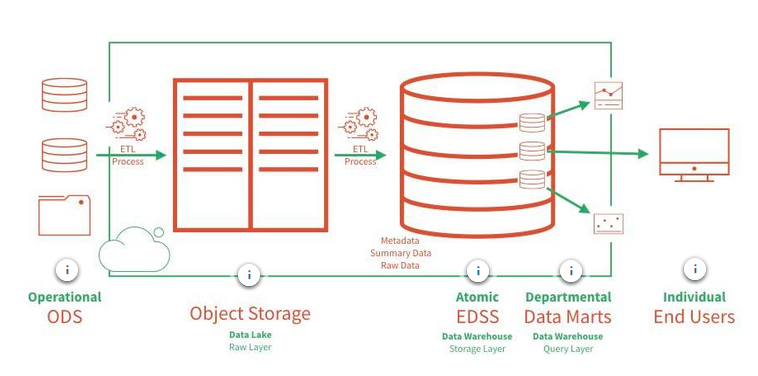
Expenses accrue in several areas:

* Initial hardware investment
* Teams of engineers required for both scaling systems and ongoing management
* Time lost due to required downtime for system upgrades
* Costs of having a dedicated database administrator (DBA) required to oversee an ODW
* ETL processes required for leveraging downstream applications like cloud-based analytics systems

**The Cloud-based data warehouse system**

Another approach to implementing an EDSS is the cloud-based data warehouse (CDW). This architecture often includes a low-cost raw layer as part of the ETL process.

Such an approach provides an optimized cloud-native system for reading data stored in the cloud.



**Benefits of cloud-based data warehouses**

Separation of compute and storage

–

A CDW is typically built separating compute and storage.

Compare this to the inelastic ODW.

The separation of computation and storage makes it easier to allocate resources elastically.

Elastic resource allocation

–

Cloud vendors allocate CDW resources elastically.

Elastic resource allocation means that resources can dynamically be scaled up or down to meet expanding or contracting workload demands.

Optimized for query throughput

–

By design, CDWs handle a large volume of concurrent queries, meaning that many users can simultaneously query the system.

CDWs are very quick in reading data and returning results, excelling in low latency queries.

Cloud-based backup of historical Data

–

Because data is stored in the data lake before being passed to the data warehouse, an historical record of the data exists in these raw initial files.

No database administrator required

–

Typically, no database administrator is required to maintain a CDW because resource maintenance and allocation is handled by a cloud service provider.

**Challenges with cloud-based data warehouses**

Medium data gravity

–

One significant problem for organizations using a CDW is that data must be stored in a proprietary format within the cloud object storage belonging to the CDW provider.

Organizations are locked in and typically pay more per volume for this storage then they would pay storing the data in their cloud. CDW providers typically bill for storage at a premium.

Although data is in the cloud, moving the data requires expensive ETL jobs to successfully transfer it from the CDW provider's cloud object storage to an organization's cloud storage, where it can be used in another analytic application (like Tableau or Databricks).

Structured data only

–

A CDW is by nature a structured data store. It can only ingest structured data.

In recent years, a NoSQL (not only SQL) approach has been found to be reliable and valuable in terms of the flexibility it provides.

A CDW cannot do this.

Black-box processing of queries

–

CDWs typically offer little opportunity for optimizing data queries, often lacking the ability to index data or provide transparency into the query execution plan.

Data Lake not queryable

–

Because data in the data lake only exists as raw data, it is not queryable.

In other words, the data housed in a CDW data lake would only be accessible by running an ETL process to load it into a data warehouse.

Expense

–

A CDW is expensive to build and maintain. Expenses accrue in several areas:

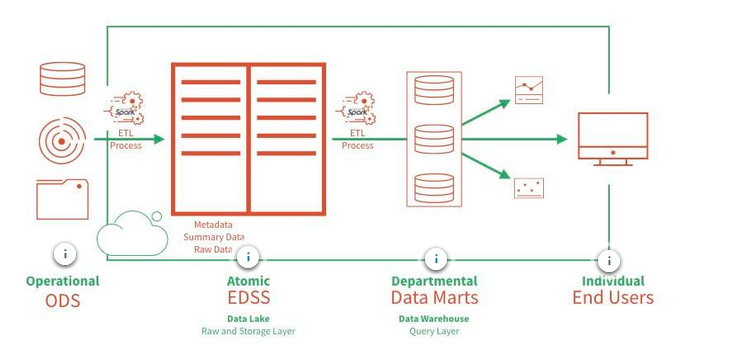
* ETL processes required for leveraging other cloud-based analytics systems
* Storage in vendor object-stores

**The Cloud Data Platform**

Another approach to implementing an EDSS is to designate a data lake built on cloud-based object storage as the SSOT.

This is unique from other data warehouse architectures, because **the designated EDSS is the data lake**, instead of a data warehouse.

In subsequent lessons, we will explore this architecture as implemented with Delta Lake and why it is the best option for building an EDSS.



**Benefits of the Cloud Data Platform**

Separation of compute and storage

–

A data lake is foundationally built by separating compute and storage.

Compare this to the inelastic ODW.

The separation of computation and storage makes it easier to allocate resources elastically.

Infinite storage capacity

–

With the unlimited availability of cloud-based object storage, organizations do not need to worry about running out of space.

Leverage best aspects of a data warehouse

–

Using a data warehouse as a query layer means that all of the advantages of a data warehouse can be leveraged for reads:

* DBA-free resource allocation
* High query throughput and concurrent reads

Expense

–

A cloud data platform is the least expensive option for building an SSOT. Organizations pay for compute only when needed.

Low data gravity

–

The SSOT is data in the data lake.

Cloud object stores can use any data format. Raw data can be kept in the format deemed best by the engineering team.

This arrangement has **the lowest data gravity**, meaning it is easiest to move from here to any other location or format.

High data throughput

–

High data throughput refers to this architecture's ability to handle a much higher volume of data per unit time.

Apache Spark supports high data throughput by design.

Since the cloud data platform leverages Apache Spark for moving data, this architecture also supports high data throughput.

No limitations on data structure

–

Using a data lake as the SSOT means that there are no limits to the kinds of data that can be ingested.

A data lake is a NoSQL system meaning that it can support structured as well as semi-structured and unstructured data.

There are many advantages to a NoSQL approach. For one, it can significantly reduce the resources organizations must spend to clean up data.

Mix batch and stream workloads

–

Building a system with Apache Spark means that the system benefits from the Structured Streaming capabilities of Apache Spark.

This means that the system can work with both stream and batch data sources.

**Challenges with the Cloud Data Platform**

Not designed for high concurrency

–

While this architecture is designed for high data throughput, it is not designed for high concurrency.

A cloud data platform can be less efficient running many simultaneous queries.

Poor interactive query experience

–

Issuing queries over a data lake can have a great deal of overhead for each query, as Apache Spark is designed for data throughput. In other words, Apache Spark handles large data loads well but can struggle to handle many concurrent queries.

Querying batch and stream data requires lambda architecture

–

While Apache Spark makes it possible to query data in a Data Lake, a complex lambda architecture will be required to validate all of the data at read time to ensure that the most accurate state of data is queried.

**Using Delta Lake to build your cloud data platform means that a lambda architecture is not required.**

Knowledge of the Spark environment required

–

At a minimum, support from data engineers who can configure the Spark runtime for the system is needed.