6 Things to Get Right for the Logical Data Warehouse

Published 15 February 2021 - ID G00736619 - 18 min read
By Analysts Henry Cook

Initiatives: Data Management Solutions

The logical data warehouse is now well established as a best practice for which there are many potential starting points and ways to expand. This research helps data and analytics leaders and their lead architects to ensure that their LDW projects are set up to succeed.

Overview

Key Findings

■ Because of its multiple capabilities and components the logical data warehouse is more complex, but also more comprehensive, than traditional, single-purpose analytical systems.

■ To be fully effective the correct collection of logical data warehouse components must be systematically chosen and integrated.

■ The logical data warehouse can begin from a number of starting points and then be expanded in different directions, sometimes simultaneously.

Recommendations

Data and analytics leaders and their data management solution strategists and designers should:

■ Familiarize themselves with the architecture of multiengine, multicomponent LDW solutions.

■ Provide an integrated view of the data and analytical techniques of the LDW through metadata standardization and data virtualization.

■ Future-proof their system by matching their analytical requirements to the workloads and service-level agreements both now and anticipating future needs. Delegate analytical work to the right processing components inside and outside the LDW.

■ Ensure the delivery of a complete integrated system by thinking in terms of the key deliverables; requirements, data, data models, architectures and phased project plans. Enable agile development through zoning, DataOps and data orchestration.
Introduction

Data warehousing has existed since the 1980s. It has evolved into its modern incarnation, the logical data warehouse (LDW) which integrates multiple analytical systems. In particular, it incorporates the traditional enterprise data warehouse and the data lake. There are six key areas that must receive attention, as seen in Figure 1.

Figure 1. Six Things to Get Right for the Logical Data Warehouse

Six Things to Get Right for the Logical Data Warehouse

1. Familiarize with LDW components
2. Provide an integrated view with metadata and data virtualization
3. Future proof your LDW, balance work, requirements and SLAs
4. Ensure delivery through key deliverables
5. Delegate analytical work to the right processing components
6. Enable agility: Zoning, DataOps, data orchestration

The context for this evolution is the changing requirement for analytics in large enterprises. The volume, complexity, variety and applicability of analytics has grown rapidly.

In the early days of analytics, there was a very restricted range of data types and processing that could be performed. This has now changed radically, as shown in Figure 2.
Figure 2: The Challenge — Modern Analytical Workloads

The Challenge — Modern Analytical Workloads

If individual systems are developed to work with specific combinations of data, processing technique, delivery method and development style this leads to an explosion in the number of systems. This makes them hard to coordinate, synchronize and reconcile — slowing development and thus delivery of benefits.

The goal of the LDW design is to maximize return on investment

Instead, what is needed is an approach that provides the maximum return on investment while still meeting requirements and delivering benefits. The modern system should allow any combination of data and processing while minimising cost — both in terms of development and runtime. The modern
approach should also maximize the number and speed of requirements being met in order to maximize the benefits returned.

The current best practice to meet these challenges is the logical data warehouse.

**Analysis**

**Familiarize Yourself With the Architectural Components of a Logical Data Warehouse**

It is important for data and analytics leaders and their lead architects to have a clear picture of the type of system that they will deliver. This is the target architecture to meet their data and analytics requirements (see Figure 3).

Previous Gartner research has described how to classify and plan for the wide variety of data and analytics that the typical large organization has to cope with (see *The Practical Logical Data Warehouse*). This research forms a bridge to the more detailed implementation oriented research from Gartner for Technical Professionals, such as *Solution Path for Planning and Implementing the Logical Data Warehouse*, and *Building Data Lakes Successfully — Part 1 — Architecture, Ingestion, Storage and Processing*. These best practices will enable data and analytics leaders to ensure a correct overall approach before committing to detailed design and implementation.

The aim is to have the minimum number of instances and the maximum amount of reuse for the components and their data. Sometimes it will be unavoidable to have multiple components (different data lakes, for example) for different purposes, such as IoT, social media or genomics. Also, more than one real-time component may be needed for performance or availability — the aim is to keep this to an absolute minimum.
Figure 3: Practical LDW Architecture

Multiengine Architecture

The LDW is an integration of multiple components, with each component acting as an analytical system in its own right. Typically, these components are the data warehouse, data lake, data marts, and real-time data warehouse systems to provide real time operational intelligence. In addition there may be additional analytical engines for graph processing and machine learning, either contained within existing components, or being placed in separate component engines alongside them. Modern analytical workloads are varied and diverse.

No one component can perform all of the workload, but between all of them, all requirements can be accommodated. Sufficient variety of components are used to cover all the bases — components are kept to a minimum and each is reused many times. Regard the components of the LDW not as competing solutions but as collaborating engines within an overall solution.

The LDW provides a portfolio of different data engines, storage and processing techniques to share the work, as well as well-defined integration techniques between them. The architecture is very flexible,
components can be added, enhanced, replaced and removed so that the system can evolve.

Some components may need to grow to hold more data, or to do more processing, but this is not a problem as most are linearly expandable. The overall architecture is very stable. The requirements find a natural home for their data and processing within the architecture. It is not necessary to constantly redesign the architecture as new requirements arrive — this greatly assists agility.

In addition, the data and processing for a requirement can be moved around the architecture as demand changes, or as service-level agreements (SLAs) evolve. If common data definitions and metadata is used, together with data transport mechanisms such as ETL, this is relatively straightforward.

**Multiple Starting Points and Expansion Paths**

Developers can build out the architecture from a variety of starting points. This is both a strength and potential source of confusion. It is common that the data warehouse is established and a lake needs to be added, or vice versa. Likewise, specialist data marts may be added, or real-time operational data stores to provide operational intelligence. However, the number of types of components and instances implemented is kept to a minimum — unlike the proliferation of data marts seen in older systems.

The interfaces between components can also be systematically designed — specifying the method of transmitting data and metadata along each path between components. Data virtualization provides a convenient way of accessing all components, but performance needs to be taken into account.

The LDW is accessed via a variety of interfaces including SQL, batch, ETL, REST, streaming or API. To the outside world the LDW usually appears as an integrated whole. In some circumstances, analytical consumers may access a component directly if necessary, usually for performance reasons.

**Provide an Integrated View with Metadata and Data Virtualization**

Data and analytics leaders and their lead architects need to consider how the LDW will sit among their existing systems and the overall approach to metadata. Metadata can be used by data virtualization tools to provide a single interface to the LDW (see Figure 4). The different metadata stores are shown in different colors, the bottom line; expect to use multiple stores and to have to integrate them.
The system is tied together using data virtualization (federation), metadata and data transports. The data virtualization may be a third-party tool or, increasingly, it is built into one or more of the major components. Most modern data warehouse DBMSs have some federation capability. Increasingly, data may be easily shared between the data lake and the data warehouse. The data transport may simply be a standard ETL tool, replication, streaming or other data methods like analytical query accelerators (see Market Guide for Analytics Query Accelerators). These may include data hubs, which orchestrate data transmission from sources to targets.

Remember, existing business intelligence and application systems can be incorporated into the LDW. For example, retail product trend analysis can be done within the data warehouse or lake, and then real-time stock holdings looked up from the physical retail distribution depot systems.

Data virtualization provides a unified semantic layer for data consumption. It can provide a single interface into all of the underlying components, thus presenting a single face to data consumers. It can also help with performance by using techniques such as data caching and cost-based optimization.

Remote systems can be accessed by data virtualization. To join data from remote sources needs confidence in the data compatibility between the systems — both in structure and quality. Network and processing performance must be adequate — however, this can often support the required SLAs and be an effective and agile implementation method.
Developers can use data virtualization to make prototypes of analyses and move them into production — if the SLAs are acceptable. If the SLAs of the requirement cannot be met placed where they are, then the data can be moved to other components of the LDW. This can be done with minimal disruption to the consumers of the analyses by using common and standard interfaces. The interfaces between the data virtualizer or BI analytical tools and the different data systems will determine how smooth these changes can be. These need to be checked in advance, as will the subject of proof-of-concept exercises. In many cases, the main difference the consumer would see is a change in performance. Common metadata standards and standard interfaces will make this easier.

While proliferation of components is to be avoided where possible, dedicated components are sometimes justified. This would be the case for sandboxes and data marts that could be created and destroyed on demand. There may be special processing requirements — for example the need to work remotely and offline — that mean a dedicated data mart makes sense.

Future-Proof Your LDW by Balancing Workloads, Requirements and SLAs

Data and analytics leaders need to ensure that their LDW will be responsive to business needs, and requirements are correctly prioritized in order to maximize return on investment. Requirements can easily be mapped to an existing part of the architecture. If needed, a new component can be added. Once added, a component will be shared, reused and expanded as other requirements arrive that need the same capabilities. Adding a graph engine would be an example. The LDW can be thought of as a “composable system,” that combines many different capabilities into an integrated whole.

Note that existing components and instances can also be used, and the LDW architecture can be used to pull these together to gain value from reuse and avoid reinventing the wheel.

Also note that the placement of data and processing is a continual exercise. Data and demands (and therefore workload characteristics) can change over time. One of the advantages of the LDW is that it is relatively easy to move data and workloads — if the right standards and mechanisms have been put in place.

Making decisions about data placement should be done on a regular basis, because data and requirements will evolve — it is not a one-off exercise.

In fact, this can be used proactively to take advantage of the different components. It is easy to take on new data and begin experimenting with it in the data lake, which means that prototyping can occur there. If regular reporting with more stringent service levels is needed, the data and analysis can be moved elsewhere. Agile development techniques are used throughout but some parts can be developed at a faster pace than others.
Using the right components for each requirement optimises development costs since development is easier and quicker. Runtime processing will be more efficient too.

Because each requirement request has an estimate of cost and benefits, it can be prioritized, as much as is possible, to deliver the most benefit at the least cost. The LDW architecture facilitates servicing more requirements faster and at lower cost, thus maximizing return on investment over time.

Note that moving data and processing between components may entail costs due to changes in governance. That is, the governance policies may be the same for all components but the physical implementation can be different. The governance tooling may be in place for each component but the policy relating to particular sets of data may need to be translated as it moves around the system.

As part of the requirements, service levels have also to be met. This includes nonfunctional requirements such as performance and availability. There is no point providing the correct information if the system is frequently unavailable or too slow to be usable.

Assigning data and workloads to the LDW is typically driven by two questions, as shown in Figure 5.

**Figure 5: Aligning Requirements to LDW Components**

### Aligning Requirements to LDW Components

<table>
<thead>
<tr>
<th>Cost</th>
<th>Requirement</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>Xxxx Finance xxx x xxxx xxxx</td>
<td>10</td>
</tr>
<tr>
<td>0.1</td>
<td>Xx Internet of Things xx x x xx xx xxxxxxx</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>Xxxx xx high-volume business process API to xxx x xxxxx x</td>
<td>12</td>
</tr>
<tr>
<td>1.0</td>
<td>Xxx roving buyers xxx xx xx xxxxx</td>
<td>25</td>
</tr>
</tbody>
</table>

**Key Questions:**
- What kind of data?
- What kind of service level?

1. What type of data is being used?. This suggests what type of storage is suitable.
2. What are the service levels for the analysis being performed on the data? This determines what type of server or processing technique is needed.

Placement of data and processing is illustrated in Figure 5.

- The Finance requirement will be using structured data that has to be quality assured. In addition it is likely to be consumed by many users who require a fast response time for their predefined and ad hoc requests. It belongs in the data warehouse.

- The next requirement mentions the Internet of Things. This is likely to be very high volumes of semistructured data. It belongs in the data lake. Most of the processing is likely to be batch-oriented, for example, the production of predictive maintenance models by data scientists.

- Operational business processes need to invoke high volumes of short requests and expect low response times via an API. Examples would be serving operational dashboards, mobile apps and automated processes. These also most likely need to provide high availability, so mainstream processes can continue despite any problems with the other components. Therefore, they need to be placed on a high-performance server hardened for high availability in production — however, not all the components need this level of availability.

- Remote uses who frequently travel or work off-site and cannot guarantee network connectivity may need physical and portable servers — specialist data marts. These would be kept to a minimum.

There is also the issue of the balance between predefined queries and agile, exploratory and self-service. This is discussed further on in this document.

For further discussion on connecting business aims and the technology architecture, see How to Optimize Business Value From Data and Analytics Investments ... Finally and Use Gartner’s Value Pyramid to Connect Data and Analytics to Business Value.

Delegate Workloads Inside and Outside the LDW

Data and analytics leaders and their senior design staff will find there are choices as to where different types of processing can be run.

It is common for queries to span both data warehouses and data lakes via federation and inbuilt data virtualization within the DBMS platform. In this case, parts of a query will be “pushed down” into the other component, and the partial intermediate results returned to be combined with data in another component. This may be done within the database layer, or at the data virtualization layer (see Figure 6).

This particularly applies to machine learning, where the user of the data science tool may specify the algorithm to be run, and see the result in the front-end tool. However, there is a choice as to where the algorithms themselves may run. They might not run in a local data science tool or service. Instead, they may be invoked to run within a warehouse, mart or lake, thus avoiding data movement.
This issue is a further refinement of the step that decides where in general the data and processing for a requirement should be placed. Clearly a requirement needing data science analysis needs to be run using a data science tool — but there is a further choice as to where the data science algorithms actually run. Data and analytics leaders and their design leads should consciously think about where the processing will be done to optimize cost, performance and availability.

Ensure Delivery by Thinking of the Key Deliverables

Data and analytics leaders can assure themselves that the LDW project is heading in the right direction by asking questions about the key deliverables of an LDW project. While during the project there will be a large number of detailed deliverables that project managers are responsible for tracking, there are five key high-level deliverables that must be managed. Preparing and discussing progress on these, initially at a high level, will ensure that the project has a good foundation (see Figure 7).
The data model allows the data sources, and the data required to meet requirements to be connected, and held with minimum duplication. Knowing the data and the processing needed to meet the requirements allows a system configuration to be designed, sized and cost estimated.

As part of this process, the data quality, requirement and architecture must be evaluated in terms of suitability and in meeting the service levels required.

The phased project plan defines how the system is to grow. In each phase, each of the deliverables are incrementally implemented, so a big bang implementation can be avoided. This will often be determined by the use cases being covered. This can be thought of as cutting a slice horizontally through the end-to-end architecture and implementing corresponding parts of the data sourcing layer, data model, system configuration and delivered requirements. In practice, this will be done through continuous agile sprints.

**Enable Agile LDW Development for Zoning, DataOps and Data Orchestration**

Data and analytics leaders can use the LDW to resolve the tension between more agile and less agile development methods.

The LDW architecture allows data and analytics leaders to accommodate multiple development environments as well as different processing types.
Different rhythms of development can be kept separate through appropriate zoning within the system (see Figure 8). Developers can allocate data mart (physical and virtual) or sandbox areas to support prototyping, experimentation and agile development. This allows experimentation and prototyping with fast delivery times, without endangering the integrity of the other main components. Change control can be applied to safely move work artifacts between the different areas of the LDW.

The principles of DataOps are also relevant. Agile development is the norm for analytics, but requirements can differ significantly in their acceptance criteria. For some, the criteria will be strict and include high standards for data quality, data lineage, completeness and accuracy. These requirements will also need sets of dependencies on ETL jobs, schedules, machine capacities and others before they can be released into production.

However, other analytics do not have these same stringent dependencies. It is sufficient in these cases to simply make data available in a form that enables fast analysis in order to get an answer quickly. It is not that these requirements are being treated in a lax manner. Rather, it is simply recognizing that the higher standards of acceptance criteria that would apply to, say, regulatory and financial reporting, are simply not applicable.

These less stringent requirements can be treated differently. In many cases this is enabled by a different ringfenced development environment that does not enforce unnecessary constraints. These more experimental environments are subsystems within the LDW. Note that, while some constraints can be legitimately be removed, any analysis must conform to regulatory and privacy legislation.
Different degrees of agile development can be separated from each other. The more experimental types of development can be isolated from the rest of the system from a security and workload management point of view. They should not be able to impact the work of the rest of the system, nor circumvent security and privacy laws. They can share data from the main system easily — provided they adhere to the appropriate regulations.

Data orchestration can be used to move data around the system. When data moves from the more stringent requirements of the data warehouse and data lake into the more agile sandbox and mart areas, controls can be relaxed. It may no longer be necessary to update the data with daily feeds that have guaranteed update times and assured data quality. Manipulation of the data may be allowed to recast it into different forms to make it suitable for new analysis. New data descriptions may be created.
If agile development simply provides a fast (if sometimes approximate) answer to an urgent question, then at the end of the analysis the resources are given up and returned for reuse by others. If the agile development provides something of lasting value — such as a new report or predictive model — then the prototype is moved back through change control. Moving back through change control is the trigger to reapply the necessary production and governance controls.

A common problem with business intelligence and analytics is that there may be 10 requirements presented, but only three will provide benefit. Therefore, the objective is to identify as fast as possible the three requirements that are worth pursuing, and then put the resources into getting them into full production. This needs agility.

Data and workloads can move between components over time. A requirement prototyped in the data lake can be moved to the data warehouse if it needs to support large numbers of users with tight SLAs. Likewise, reporting and batch work may move to the data lake to save processing and storage costs if the SLAs allow.

Evidence

This research is based on published Gartner research and discussions with clients during Gartner inquiries over the past two years.

Recommended by the Author

The Practical Logical Data Warehouse
Best Practices for Designing Your Data Lake
How to Avoid Data Lake Failures
How to Optimize Business Value From Data and Analytics Investments ... Finally
Use Gartner's Value Pyramid to Connect Data and Analytics to Business Value
Solution Path for Planning and Implementing the Logical Data Warehouse
Building Data Lakes Successfully — Part 1 — Architecture, Ingestion, Storage and Processing
Building Data Lakes Successfully — Part 2 — Consumption, Governance and Operationalization
Market Guide for Analytics Query Accelerators