International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I7P111
Volume 72 Issue 7, 87-92, July 2024
© 2024 Seventh Sense Research Group®

Original Article

Analyze and Optimize Data Pipelines with Effective Data Models

Piyush Pandey

Independent Researcher, NC, USA.

[1]*Corresponding Author : piyush.lohaghat@gmail.com*

Received: 17 May 2024 Revised: 28 June 2024 Accepted: 17 July 2024 Published: 31 July 2024

Abstract *- In the current data-driven world, all organizations rely on data warehousing solutions to conduct their daily operations and decision-making. Refreshing the data in analytical data warehouses in a timely manner is one of the critical goals of the data operations team. Technology has advanced a lot over the last couple of decades with the evolution of innovative and powerful processing engines, e.g. Spark, Hadoop, advanced databases, etc. But new challenges like increasing data volume, integration of additional sources, complex transformations, datasets for new use cases and unforeseen issues keep the operation teams on their toes. Generally, teams tend to add more resources (CPU, RAM, etc.), which is an easy way out for temporary respite. However, nothing comes for free – more resources mean increased infra costs. Hence, there is a need to dig deeper and analyze the ETL [1] processes to identify the bottlenecks and suggest corrective actions/ design changes. While doing a deeper analysis, the run history of ETL jobs is crucial for ensuring data integrity, optimizing performance, and maintaining overall system health. There should be enough buffer time to meet SLAs [2] in case of abends or unforeseen issues. Most of the research on ETL performance is focused on the "how" to optimize data refresh times, but there is less research done to identify "what" to optimize. Moreover, analysis and optimization of ETL require not only the technical skillset, but also a functional understanding of the nature of data. This article talks about approaches to analyze an ETL run and identify what are the problematic ETL steps. This article also talks about processes and ways to improve pipeline performance based on appropriate data models [3] and actions, with a knowledge of domain data.*

Keywords *- Data Pipeline Analysis, Effective data models, ETL Extract Transform and Load, Optimize Datawarehouse.*

1. Introduction

Data pipelines facilitate the extraction, transformation, and loading (ETL) of data from disparate sources into centralized repositories, such as data warehouses or data lakes. With the increasing volume, variety, and velocity of data, optimizing these pipelines has become imperative. Effective data models play a critical role in ensuring data pipelines are both performant and scalable.

As organizations expand, they encounter a surge in data volume. Properly optimized ETL (Extract, Transform, Load) processes are crucial for efficiently managing larger datasets without experiencing significant performance drops. These streamlined ETL systems can easily adapt to various data sources, formats, and evolving business needs without necessitating major overhauls. Efficient ETL processes also help mitigate the burden on both source and target systems, preventing performance bottlenecks that could negatively impact other applications, concurrent processes, and users. Organizations that can swiftly process and analyze data gain a competitive edge by obtaining insights faster and making timely business decisions ahead of their competitors. Streamlined ETL systems enable businesses to experiment with new data sources and analytics techniques without being limited by performance concerns.

For data analysts, data scientists, and business users, reliable ETL processes are essential for accessing the data they need. Slow or unreliable ETL systems can cause frustration and hinder these professionals' ability to perform their tasks effectively. Efficient ETL processes support self-service analytics by providing accurate and timely data, empowering users to explore and analyze information independently.

1. 2. Literature Review

Streamline and optimization of ETL pipelines can be addressed by various means. Eg. Following ETL best practices, implementing ETL and data-warehouse on best tech stacks available, increasing storage and computing power, etc. However, it is critical to understand gaps and problems in current implementation before ETL can be optimized. ETL performance problems should be addressed and reviewed from a data model and functional standpoint as well. A comprehensive research and study have already been done on optimizing ETL and best practices for ETL development.

A study by Dhamotharan Seenivasan [4] outlines the ETL best practices for performance optimization and ETL design. Another study for ETL performance improvement gives insights into the query cache approach. [5]

An article by Lina Dinesh and Gayathri Devi [6] proposes algorithms for handling and optimizing large volumes of data on ETL processes based on cloud computing. optimizing ETL processes, companies can lower operational costs by reducing the need for extensive computational and storage resources, which is particularly beneficial in cloud environments where resource utilization directly affects expenses.

The literature review by Mozamel M. Saeed1, Zaher Al Aghbari, and Mohammed Alsharidah [7] discusses spark optimization techniques from a spark data clustering perspective.

An article by Xiang Wu and Yueshun He [8] addresses some of the ETL performance concerns on Spark with respect to join optimizations. The article also discusses how a table profile relates to spark optimization schemes.

A review of existing open literature suggests that not enough research has been done to optimize ETL using application domain knowledge and data models. Although ample studies are addressing ETL performance concerns, they are lacking in an approach to analyze and find the root cause of ETL performance issues.

3. ETL Job Summary

During the ETL process, the total run time comprises the durations of extraction, data transformation, and loading to the target warehouse. The end users have access to refreshed data after all the steps are completed successfully. The total duration of the ETL process could be the sum of each step, or there might be overlaps among the steps. This article will consider that each step is being executed in series. Additionally, the scenario considered in this article has both the source and target relational databases, and the transformation is performed using Spark in a file system.

The first step in analyzing any ETL run is to review the high-level job summary, focusing on the time taken and the number of records processed. The statistics for a given pipeline run should be compared with historical stats and trends. By examining these data points, one can identify which ETL steps require further analysis. For example, as shown in Table 1, the transform step is the longest-running step, with an average duration of 96 minutes. In the current run, the transformation step took 110 minutes, which is longer than the historical 60-day average. Therefore, to optimize this pipeline, one should start by diving deeper into the transformation run time details. While discussing all three steps, the primary focus will be on the transformation step.

Table 1. Job summary for an ETL run

Job Summary	**Current Run Summary**		**Historical Average (60 Days)**	
	Duration (Mins)	# Records ('000)	Duration (Mins)	# Records ('000)
Total	164		136	
Extract	23	3,479	21	3,189
Transform	110	7,409	96	7,118
Load	21	6,918	19	6,598

When reviewing the job summary and setting benchmarks for performance or SLAs, it is also essential to consider the resources and workload involved during a specific data pipeline run.

Table 2. Key KPIs description

# Executors [9]	A process launched for an application on a worker node. The higher the number of executors, the higher the number of worker nodes.
CPUs	Number of CPUs
data Read	During transformation, data from tables will be read from storage. Data read depends on transformation logic and table size. In some cases, transformations written to scan full tables will have high data reads compared to transformations written in an optimized manner with incremental delta processing.
Data Written	Transformed data written back to storage.
#Tables Transformed	Number of transformed objects, including stage and final objects.
# Load Plans	An incremental pipeline can bring data from different systems and functionalities to different areas. Sales data from OLTP, warranty data from flat files, etc.
Throughput	# of Records process/ Time taken
Error Rates	Frequency or count of error records (duplicates, schema mismatch, etc.)

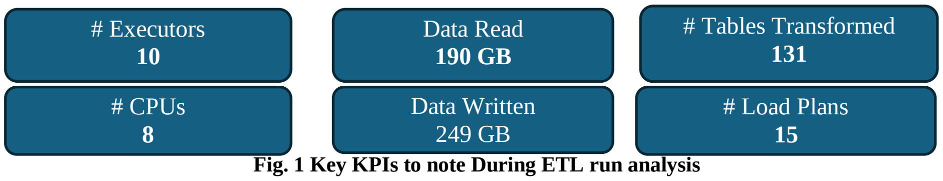

Fig. 1 Key KPIs to note During ETL run analysis

4. Extraction Step

The first step to analyzing extraction is to get a list of extraction objects/ queries which are regularly running for a longer duration and impacting downstream jobs. Incremental Delta Extraction [10]: Extract the data that has changed since the last ETL run. This reduces the volume of data being processed.

Sometimes, there are cases when source data sets do not have a way to identify updates (last updated date not available), or datasets can go through hard deletions. In such cases, one alternative could be to check if any triggers or logs are available in the source which captures deleted transactions. Also, instead of extracting such datasets in full for all the columns, check for the possibility of extracting only the key columns and determine deleted records on the transformation side.

Source Filtering: Apply filters at the source database level to reduce the amount of data extracted. Extract only the necessary data columns and rows required for processing. Instead of issuing complex queries to source systems, extract and stage individual datasets locally. Do the joins and heavy-duty operations post extracting the base data instead of overloading source systems.

5. Transform Step

During the transform step, the extracted data is converted into a format required for analysis. During this step, the data is cleaned, filtered, joined with other datasets, aggregated, etc., to achieve business and functional objectives.

To analyze the performance bottlenecks during the transformation step, the first step is to determine the transformation which is holding the pipeline. List down the top n (5) steps which are running for the longest duration.

In the table below (Table 3), Revenue_Fact is running for the maximum duration of 22 minutes, followed by Sales_Agg_Fact, taking 20 minutes. At first glance, it makes sense to analyze Revenue_Fact transformation first. However, Revenue_Fact is running Parallel to Sales_Fact. So, unless both Revenue_Fact and Sales_Fact are analyzed and tuned, the overall pipeline time will not improve much. On the other side, Sales_Agg_Fact is running in parallel to Sales_Backlog_Fact but Sales_Backlog_Fact finishes in 5 minutes. This means Sales_Agg_Fact is running in standalone mode for the remaining 12 minutes, thus holding the pipeline. If Sales_Agg_Fact performance can be analyzed and improved, there is a better chance of bringing down the run time of the pipeline.

Once the consistently long-running transformations are identified, further analysis is required to identify the long-running cause. The time spent on the transformation step depends on a lot of factors, such as data volume processed during an incremental run, complexities involved in transformation logic, the underlying data model design, resources available for processing, etc. While there could be numerous reasons behind the above factors, a few scenarios will be explained with examples.

Next, lets focus on Sales_Stage, which is taking 18 minutes. In the given example, Sales_Order_Header and Sales_Order_Lines datasets are joined together to populate Sales_Stage. During incremental, a changed or new record could flow in either Sales_Order_Header or Sales_Order_Lines. Hence, incremental changes on both input objects should be considered, which means Sales_Order_Header (500 M records) will be joined with Sales_Order_Lines (2 B records) before incremental data filters can be applied. Cases like this could be very costly and resource-intensive operations. Joining large datasets requires careful consideration of indexing, query optimization, and resource management to ensure efficient performance. On the other hand, joining small datasets is generally straightforward, focusing on minimizing overhead and ensuring in-memory operations.

Table 3. Top 5 Long-Running transformations

Object Name	Duration (Mins)	# Records ('000)	Start Time	End Time
Revenue_Fact	22	57	14:00	14:22
Sales_Agg_Fact	20	20	14:22	14:42
Sales_Stage	18	37	13:42	14:00
Sales_Fact	17	10	14:00	14:17
Sales_Backlog_Fact	5	5	14:22	14:27

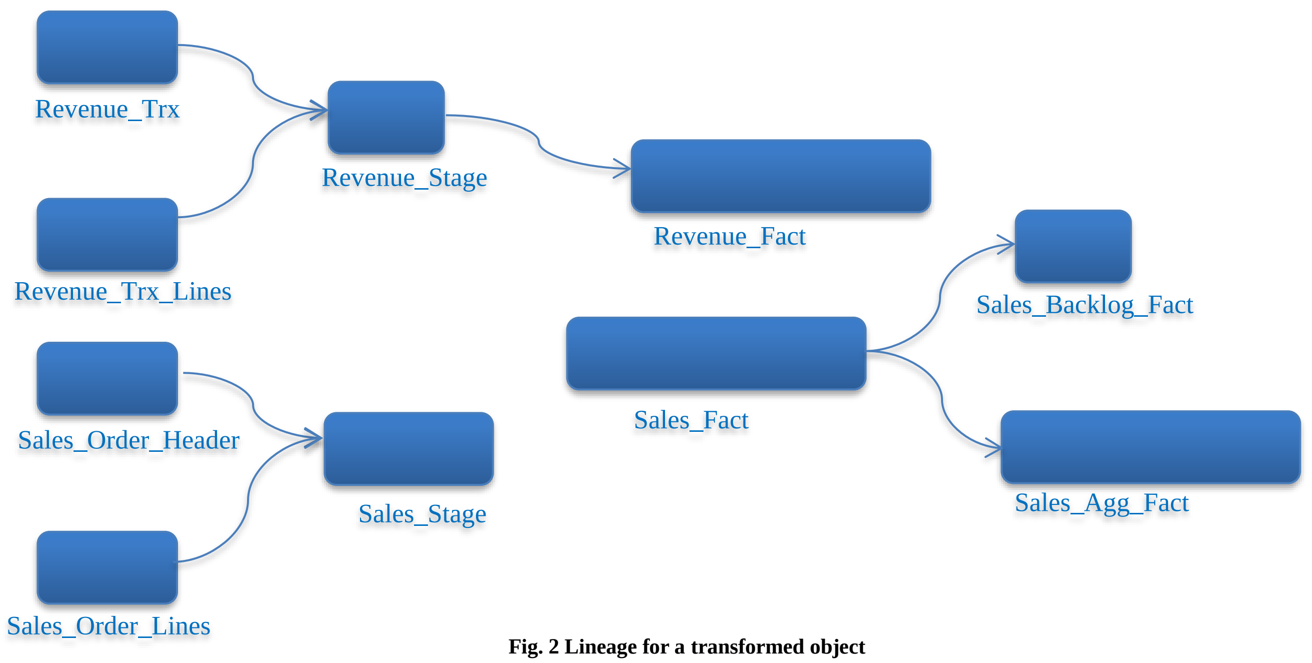

Fig. 2 Lineage for a transformed object

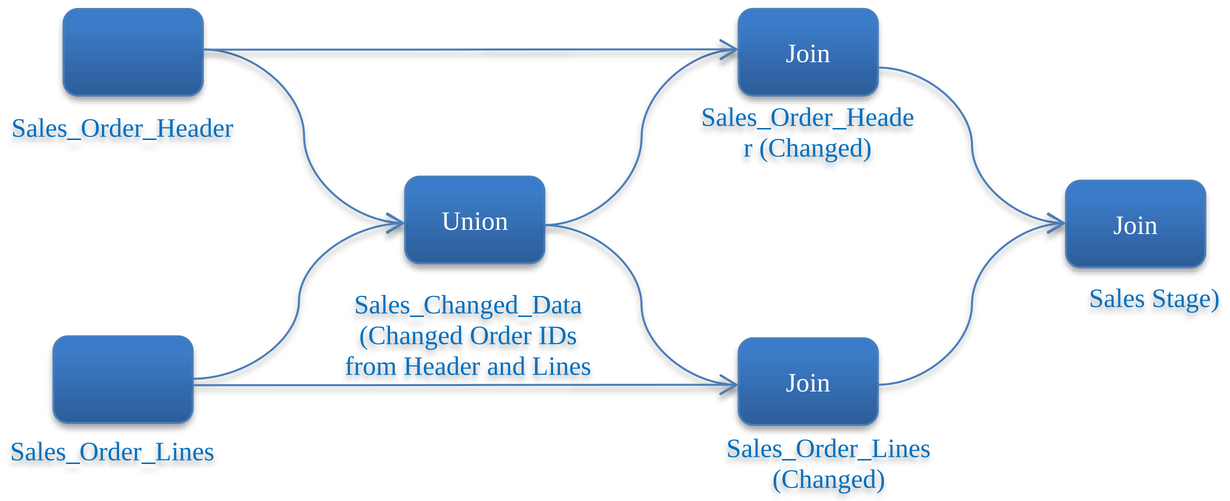

Fig. 3 Modified Lineage to Load Stage (Sales_Stage) table

One possible way to eliminate such joins between high-volume tables could be to stage the necessary data before the join. Identify changed Sales_Order_Ids from Sales_Order_Header and Sales_Order_Lines into a stage table Sales_Changed_Orders. Join the list of changed orders Sales_Changed_Orders with both Sales_Order_Header and Sales_Order_Lines to create another set of stage tables Sales_Order_Header_Stage and Sales_Order_Lines_Stage. These stage tables will only have the data for today's changed data. The volume stage tables Sales_Order_Header_Stage and Sales_Order_Lines_Stage would be much less compared to Sales_Order_Header_Stage and Sales_Order_Lines_Stage. Hence, the Sales_Stage transform, which earlier was taking 18 minutes, could finish much faster. In this example, with the modified lineage (Figure 3), the run time for Sales_Stage was reduced from 18 minutes to 2 minutes. Although there were additional intermediate steps introduced, but join of two large tables was avoided, thus providing a net benefit on the run times.[8] The impact could be greater if such datasets (eg Sales_Stage) are used as an input dataset for multiple transformations.

For significant data volume transformation, the primary check is to ensure that appropriate filters are applied either on the extract step itself or in the transformation logic. The filters should be pushed to the initial stages of transformations. Additionally, the use of operators like distinct, sort-by, and non-equi joins should be carefully reviewed.

Table 4. Sales_Stage run time comparison with modified approach

Object Name	State	Duration (Mins)	# New Records (000)	Total #Records (000)
Sales_Order_Header	Earlier	2	2	500,000
Sales_Order_Lines	Earlier	4	18	2,000,000
Sales_Stage	Earlier	18	37	37
Sales_Order_Header (Changed)	Modified	2	2.5	2.5
Sales_Order_Lines (Changed)	Modified	3	24	24
Sales_Stage	Modified	2	37	37

There could be scenarios where a transactional dataset (e.g. Sales) is joined to a dimensional dataset (e.g. Customer). During the design phase, it should be carefully evaluated if changes to the dimensional dataset should be listened to or not. Based on the nature of data and cardinality, an update on a few customer records can cause huge updates for the final dataset.

All the above considerations can be applied only when analysis is done with a functional background with a clear understanding of data transformation logic.

6. Load Step

The load phase is the final step of the ETL process, where transformed data is exported and loaded into a data warehouse. The time spent on this step depends on various factors, such as the volume of data, table schema, table size, data warehouse capacity (in terms of CPU and other resources), throttling, and concurrent processes or queries using data warehouse resources during the pipeline load.

During the load step, records can be inserted, updated (deleted and reinserted), or deleted. The larger the number of records processed during an incremental run, the longer the load times will be. Additionally, the larger the size of the warehouse table, the more costly delete and update operations are.

Some basic data modeling techniques can help improve load times, including table partitioning [11] and managing table indexes. For example, consider a scenario where the Sales_agg_fact table aggregates sales data by month. Records for the current month's aggregate will receive daily updates, while older months' data remain stable. Updating (or deleting and inserting) the current month's records in a 100-million-row table can be a costly operation. However, partitioning the table by month and then updating only the current month's partition simplifies the operation significantly.

Index management [12] also plays a crucial role during the load process. Indexes are generally defined on all warehouse tables to optimize reporting performance. However, indexes can slow down the load (write) process. Therefore, it may be beneficial to disable indexes during the load operation and re-enable them once the load is complete.

3.

7. Conclusion

In conclusion, ETL processes are critical for ensuring reliable and scalable data management in today's data-driven business world. Various business operations continually modify and enter data into source systems, causing ETL run times to vary proportionally with these changes. Events such as month-end closings or bulk updates in source systems can significantly extend pipeline run times.

To maintain optimal ETL performance, continuous analysis of pipeline run statistics, trend analysis, proactive monitoring, and ongoing improvements are essential. Leveraging advancements in technology, particularly the integration of Machine Learning (ML) and Artificial Intelligence (AI) capabilities, can further enhance monitoring and proactive measures.

ML algorithms can predict the need for additional capacity to handle high data volumes based on incremental updates in the source systems and can automatically provision this capacity for a given run. Additionally, automation can provide real-time insights and predict ETL completion times, making the entire process more efficient and responsive to changing demands.

However, the underlying data model for the pipeline remains a vital driver of pipeline performance. A top-down approach of analyzing pipelines, and key KPIs [13], taking a balanced approach in terms of tech stack enhancement and data model optimization is a must to ensure a healthy pipeline and ever-changing data warehousing solutions.

References

[1] Ralph Kimball, and Joe Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*, Wiley, 2013. [Google Scholar] [Publisher Link]

[2] CIO, What is an SLA? Best Practices for Service-Level Agreements. [Online]. Available: https://www.cio.com/article/274740/outsourcing-sla-definitions-and-solutions.html

[3] IBM, What is data Modeling?. [Online]. Available: https://www.ibm.com/topics/data-modeling

[4] Dhamotharan Seenivasan, "Improving the Performance of the ETL Jobs," *International Journal of Computer Trends and Technology*, vol. 71, no. 3, pp. 27-33, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Vishal Gour et al., "Improve Performance of Extract, Transform and Load (ETL) in Data Warehouse," *International Journal of Computer Science and Engineering*, vol. 2, no. 3, pp. 786-789, 2010. [Google Scholar] [Publisher Link]

[6] Lina Dinesh, and K. Gayathri Devi, "An Efficient Hybrid Optimization of ETL Process in Data Warehouse of Cloud Architecture," *Journal of Cloud Computing*, vol. 13, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] Mozamel M. Saeed, Zaher Al Aghbari, and Mohammed Alsharidah, "Big Data Clustering Techniques Based on Spark: A Literature Review," *Peer Journal of Computer Science*, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Xiang Wu, and Yueshun He, "Optimization of the Join between Large Tables in the Spark Distributed Framework," *Applied Sciences*, vol. 13, no. 10, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Apache Spark, Cluster Mode Review. [Online]. Available: https://spark.apache.org/docs/latest/cluster-overview.html

[10] Fahb Sabry Esmail Ali, "A Survey of Real-Time Data Warehouse and ETL," *International Scientific Journal of Management Information Systems*, vol. 9, no. 3, pp. 03-09, 2014. [Google Scholar] [Publisher Link]

[11] VLDB and Partitioning Guide. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/21/vldbg/

[12] Database Administrator's Guide. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/23/admin/managing-indexes.html

[13] KPI/Metrics. [Online]. Available: https://docs.oracle.com/cd/E99929_01/html/sm_41_omuser/omg_kpi_metrics.htm

[14] Purnima Bindal, and Purnima Khurana, "ETL Life Cycle," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1787-1791, 2015. [Google Scholar] [Publisher Link]

[15] Swapnil Gohre, ETL in Near-Real Time Environment: Challenges and Opportunities, 2020. [Google Scholar]