TABLE OF CONTENTS

Introduction

1.0 The Rise and Fall of ETL

1.0

The Rise and Fall of ETL

The first ETL solutions were built hand-in-hand with their data warehouse platforms. Relational databases were the new hot technology. Large companies, mostly in finance and retail, began acquiring on-premise databases, building out large data center estates managed and run by large IT teams with specialist knowledge. While these systems were far from perfect, they were revolutionary and fundamentally changed the way we would come to understand data.

How did companies get data into their new systems? By **extracting** it from the source system, **transforming** it into the destination format and then **loading** it into their relational database. This process grew in importance as the number of source systems a company used grew. For example, data could originate in a payments system, Excel document, CRM and ERP systems. Thus, ETL was the traditional method for 'extracting' data from numerous source platforms, 'transforming' the data on an ETL server and then 'loading' all the transformed data into a data warehouse ready for analytics and reporting.

As companies realized the potential data warehouses offered in terms of reporting and analytics, clean, structured and quality data became fundamental to Business Intelligence (BI).

The ETL engine was therefore a compute resource, and as such needed to be powerful enough to handle growing amounts of data to be transformed, used, and reused. To do ETL well you needed fast, expensive disk space to store large data sets sometimes temporarily; fast, expensive processors to perform calculations on the data; and lots of fast, expensive memory to perform data operations, such as aggregates and joins efficiently. Therefore "powerful" also meant expensive! This made databases and ETL capabilities of the time extremely costly and inflexible. This capability was, therefore, only afforded by large enterprises.

Furthermore, these large companies were made even bigger with the need to onboard highly skilled and specialized IT staff. Finding resource that could use and develop on-premise databases were few and far between and came at a high price. Once embedded, a few people could hold all the knowledge. Any changes in the team could adversely impact smooth day-to-day operations.

HISTORICAL ETL

Data is extracted from the source(s), and transformed by an ETL engine en route to its permanent home in the Data Store, which is usually a relational database.

E — extracting data from the source(s), and usually also implying that:

- There are multiple sources
- Data is staged into files or another relational database

T — transforming (i.e. converting) the raw data into a format that's suitable for reporting and analytics. This typically includes:

- Enforcing consistency (currencies, timezones, units of measurement)
- Applying business rules
- Enriching and validating (dealing with missing values, duplicates)
- Denormalizing to a simpler data model (usually a star schema)
- Joining disparate data sources

L — loading data into a target platform (e.g. a relational database)

As for the technology being developed, high levels of customization and bespoke solutions were fragile. Any changes in the technology stack would necessitate more bespoke configurations and technology and often highly specialized staff to carry out this work. This model was not ideal for inevitable business change.

There were consequently a few disadvantages to ETL, even at its heyday, which have persisted in modern times. The target schema needs to be known at the time of loading since the data will be transformed to match the schema when in transit. Furthermore, in the ETL process, in order to gain performance the granularity of raw data is often sacrificed because of the limitations in the technology's ability to scale or business' appetite for increased costs to protect granularity. Regardless of opting for performance or granularity, ETL was and still is expensive and therefore primarily adopted by large companies. With only big enterprise needs in mind, ETL can be considered inflexible for data-driven, agile SMEs. These limitations and challenges are now being further exacerbated by technological advancements and changes in the nature of data (sources, quantity and value).

Thanks to these technological advancements in data warehousing, there is a steady migration from on-premise to cloud-based data warehouses occuring. This wave of warehousing is challenging the relevance of ETL. Legacy environments running ETL software weren't built to scale in the same way as cloud data warehouses now are. Therefore, as data volumes increase and workloads become more complex these environments consume more IT resources, creating bottlenecks in the data chain and negatively impact reports and analytics. The worst outcome – bad business decisions, being made slowly, resulting in missed opportunities and ultimately losses.

Today when companies talk about data they are referring to 'big data' - and that is not just big businesses. Smaller and medium sized companies are recognizing the value of data, pushing for more investment in capturing, storing, and analysing 'big data'. Furthermore, our understanding of 'big data' is expanding to seemingly infinite scales with the proliferation of data sources. These paradigmatic changes in data technology are finding traditional data warehouses to be inflexible, too costly and painfully slow for the modern tech savvy, data-driven company. All of these are playing a critical role in challenging the relevancy of ETL in cloud-context.

However, it is not necessarily all doom and gloom. If you're still using an on-premise infrastructure and your data is predictable, coming from only a small number of sources requiring only minimal transformations – ETL could still be a legitimate cost-effective strategy. We suspect, however, that is not the case for most modern companies.

2.0 The Modern Cloud Data Warehouse

2.0

The Modern Cloud Data Warehouse

The migration from on-premise to cloud-based data warehouses is fundamentally changing the way we view and understand data. Data warehouses are not only data repositories but potential data gold mines that need to be secure but accessible, cost efficient but scalable, and capable of importing and exporting data from seemingly endless sources. The underlying architectures of modern data warehouses means ETL and legacy solutions are no longer fit for purpose in light of modern data challenges and needs.

2.1 Differences between On-Premise and the Cloud

The most obvious difference between cloud-based data warehouses and their traditional on-premise counterparts is their "serverless" design. Of course, there are plenty of virtual machines, networks and disks behind the scenes making it work, but this is all orchestrated and managed by the cloud vendor on your behalf. The fully managed nature of the service also offers security, caching, back ups, encryption, disaster recovery and other safeguards to ensure your data is as safe as technologically possible. These activities are carried out by specialists and experts, meaning you don't have to train or acquire these resources, often resulting in savings on overheads.

Instead you are left with the business focused task of gaining value and insights from your data without the headaches of hardware and software provisioning. Historically, with on-premise systems you would be burdened with laboriously extrapolating your data needs 1, 3, or 5 years down the line. It would be impossible for any company to exactly forecast and predict data needs and scale accordingly. This would put companies in a 'Goldilocks Dilemma'. You may overestimate, paying up-front for dedicated hardware and software that might not be fully utilized for years; or underestimate and over utilize within the first 6 months sending you back to your CTO to go through the painful procurement process all over again.

To break the 'Goldilocks Dilemma', most cloud databases are pay on-demand for data storage and/or query execution. This means you pay for your storage and usage by the hour, month, or query, for example, with no upfront costs simplifying the procurement process since there are no contracts. This model can also reduce your risk of vendor lock-in. In the event of a growth spurt, you can take advantage of the Massively Parallel Processing (MPP) architecture, which is a scale out approach opposed to the SMP scale up approach. Since scaling up is not a linear relationship, you could end up suffering unbalanced economies of scale. Scaling out is also essentially infinite offering few growth limitations. This is a much more cost-efficient strategy for building, maintaining and growing a modern data warehouse, which is just right.

Lastly, cloud data warehouses tend to operate columnar data storage. This means that all (or a large block) values for a particular column are stored sequentially on disk. This has a number of advantages, including (i) terrific compression, since values in a single column are likely to have similar values which compression can take advantage of; (ii) Columns not included in a query are never read from disk, compared to row-based data stored which read pages or rows regardless of which columns are required; (iii) The need for extra layers of complexity such as views, keys and indexes to be maintained in order to optimize query performance is negated, therefore, meaning that non-technical users can be allowed to explore data sets without hitting technical limits.

All of these advantages come at the cost of highly expensive operations that traditional relational databases are great at - for example single-row updates/deletes. However these operations aren't common in data warehouse queries.

What does this mean for ETL? There are some key fundamental differences between on-premise and cloud-based data warehouses. These changes have impacted the supporting processes, applications and technologies. ETL is no exception.

3.0 ELT for Data Integration Success

3.0

ELT for Data Integration Success

With increased frequency, articles and blogs on the death of ETL are being published.[1] While we don't disagree with the argument, we understand, first there are some use cases for which ETL is still a legitimate and necessary tool, and second, if ETL is dead, that doesn't mean there isn't a need for data integration tools.

The obstacle posed by data integration, or more accurately the lack thereof, is only increasing. Bespoke technology platforms, CRM, ERP, finance, marketing, email, and hundreds of other data generators need to speak to your data warehouse to push data in and pull information and insights out.

So what options do you have if you have a lot of data from numerous sources and you want to shape, clean, filter, join and transform that data? ELT is the next generation of data integration success to overcome the siloed data epidemic.

3.1 What is ELT?

'ELT' means you extract data from the source, load it unchanged into a target platform (which is often a cloud data warehouse), and then transform it afterwards, to make it ready for use. There's also an important implied assertion that:

- ETL is highly targeted (always driven by known requirements), and so only extracts certain very specific data from sources
- ELT in contrast, can be less selective. If you are building out a data warehouse you can use a targeted approached as above. However, if you are building a data lake you have the option to be less selective with the preference being to extract all the source data, and then allow the user to decide later what's needed for reporting

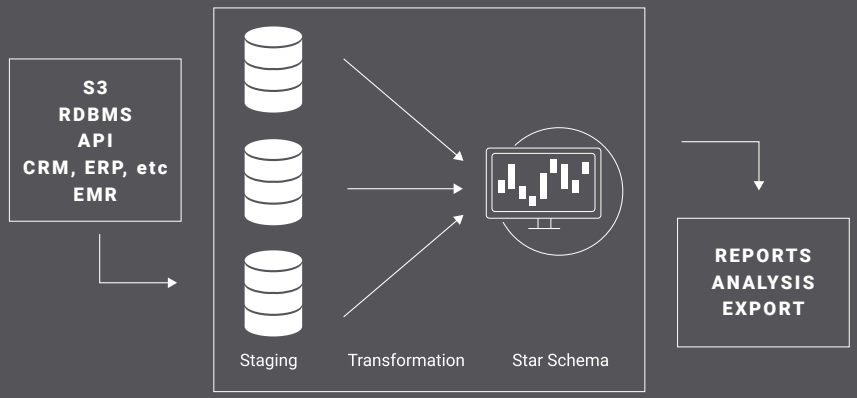
[1] For example; Mintz, Daniel. "ETL Is Dead." InfoWorld, 13 Oct. 2017, www.infoworld.com/article/3231652/analytics/etl-is-dead.html.

The Kimball Model vs. The Inmon Model

When designing a data warehouse, these are the two common patterns you could follow.

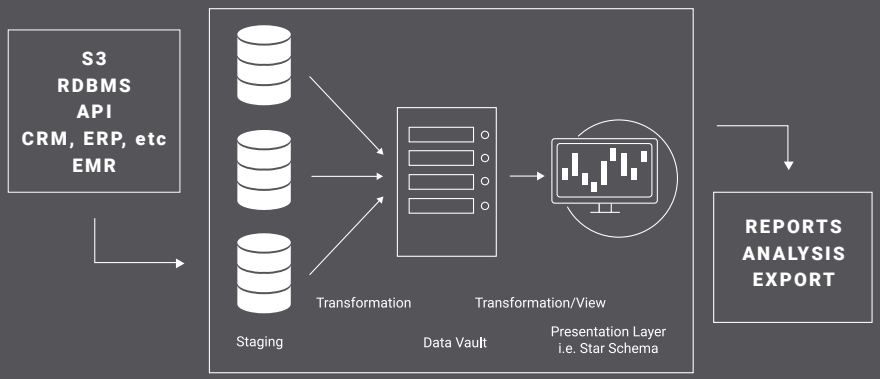
KIMBALL MODEL

The idea is that you 'extract' data from various sources and 'load' that data into the target database. There is no data transformation at this stage. Following these two steps, your data then becomes available in the database (staging area) or separate schema for raw data ready to do something with downstream. That leaves the 'T', for 'transformation'. The final stage sends transformations into the star schema. ELT transformations typically use SQL to do everything within the target database. Once you push your data you can share access, for example, with the reporting team to build reports, or denormalize structures for more sophisticated analytics downstream.



INMON MODEL

The benefit of the Inmon Model is in its attempt to address situations in which you are having to repeatedly perform complicated transformations or are applying complex business rules on the same set of data. With the Inmon approach you only need to cleanse the data once. This solves consistency problems you might experience as your data warehouse expands. For example, when loading and transforming customer data you would consider both sales and marketing needs. This keeps it relational, but has time stamps to differentiate loads. This solves consistency problems you might experience as your data warehouse expands. It is important to note that the Inmon Model assumes you have control over the source.



ELT APPLIED

While ELT is less selective, if you're not in control of your source data formats, they change frequently and without notice, or perhaps your company has very informal and unstructured procedures for storing data with ad hoc data models, you will need to relationalize the data. Find out more about the ETL/ELT lie and data relationalization:

www.matillion.com/matillion-etl/the-elt-lie/

E-L-T, as opposed to E-T-L, 'Extracts' data from source systems, 'Loads' it in its raw form into a target platform, and then allows you to 'Transform' it in-database. It immediately becomes convenient to access, won't disappear, and is easy to audit. ELT then re-uses the power of the MPP columnar data warehouse to do the transformations, which gets the data ready for presentation, reporting, analysis and modelling. With ELT tools such as those offered by Matillion, this involves running push-down SQL, and means that you only need one powerful piece of infrastructure - the data warehouse.

ELT leverages the power of the data warehousing platform itself to perform transformations, and get the data into an analytics-ready format. Further advancements on data warehouse platforms, such as Amazon Redshift's new Spectrum functionality, makes transforming your data in-database easier, faster and more cost efficient. This design results in savings on infrastructure, better performing workloads, and shorter development cycles. Your data is quickly migrated and immediately available for transformations and analysis based on current business questions and needs.

Also it means you don't need to know how you are going to use that data from the start. You have the freedom to apply transformations at a later stage once its use case becomes more clear. This ability is increasingly appealing given the changing nature of development with the rise of iterative Agile methodologies. Thus ELT, may be more aligned with the current development best practices.

That being said, ELT has some obstacles that you shouldn't ignore.

Since your transformations are being done in-database, you will need available space and compute power to perform the desired transformations. Without this performance, queries will suffer. Cloud-based platforms such as, Amazon Redshift, Google BigQuery and Snowflake, however, facilitate scalability in a cost efficient manner that helps address this challenge. As previously mentioned, continuous advancements such as Spectrum for Amazon Redshift, make loading your data and then transforming it even easier and faster!

Another problem we often hear about is the timely and labor intensive process of scripting, which can seem like a "quick win" early on in a project, but becomes more and more difficult to maintain as the project grows in scope. This is not just for loading data, but transforming it too. It applies especially when dealing with unusual or unstructured data types, or where access is not simply file-based. The more data, data sources and needed outputs can make these jobs increasingly complex, subjecting them to human error. Where mistakes are made with hand coding, it could take days or weeks to script, run, rollback and start again. Luckily, there are a number of tools on the market that make this process quicker and easier, such as Matillion ETL.

4.0 Exactly Why is ELT Better?

4.0

Exactly Why is ELT Better?

ELT is better than ETL because it is faster. Why is it faster? To answer this question we first need to break down and compare the component parts of the 'E', 'T' and 'L'.

Let's start with the '**E**', the **extract**. In both the **E**-T-L and **E**-L-T scenario the extract performance is very often outside of the control of the the overall process. Data extraction speed is instead a factor of **source** system performance, and the network speed between either the source and the ETL infrastructure or the target data warehouse in the case of ELT. Implementation choices such as, incremental or even real time data extraction can help with the overall performance here but come at the cost of complexity in either case. In summary, the **extract** part of ETL probably does have a massive bearing on overall performance, but fortunately if this part of the process is a bottleneck there are many ways to help (more network bandwidth, use of database read replicas, change-data-capture and so on).

Next there is the '**L**' - **loading** data. In E-T-**L** this is done last, post transformation. In E-**L**-T it's done earlier. The loading of data is also a function of the performance of the target system, however, modern MPP columnar databases often have excellent bulk data loading capabilities when compared to generalist databases. So, while loading **could** be a factor if the target is a generalist database, it doesn't need to be.

Transformation '**T**', which happens last in E-L-**T**, is where the vast majority of the performance comes from. This improves speed because:

Transformations can be broken down and executed in parallel across multiple hardware nodes.

Transformations are heavily optimized by the analytic data warehouse platform. In analytic data warehouses such as Amazon Redshift, Google BigQuery or Snowflake, the data warehouse and the storage format have been designed and optimized for each other. In addition the data warehouse platform has highly detailed knowledge of how the data is stored, distributed, the data types, lengths and ranges, the compression used, the context of the query and much more. Other big data platforms such as Hadoop/Spark do without this deep knowledge of the source data and so cannot optimize queries and transformation so heavily.

The data stays within the scope of the data warehouse platform system. It's not being transmitted between other heterogeneous systems. The internal networks joining the nodes of the data warehouse platform are extremely highly optimized.

5.0 ETL vs. ELT—Practical Differences

5.0

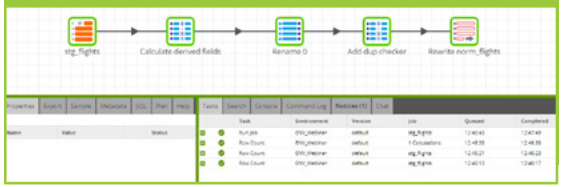
ETL vs. ELT—Practical Differences

We have talked about how modern cloud-based data warehouses are challenging ETL's performance and capability. ELT tools built for the cloud offer a superior alternative to ETL tools and traditional ETL tools that have been 'ported' to the cloud. The table below explains the practical differences in how ETL vs. ELT tackle calculations, lookups (joins) and aggregations.

Check out our **ETL vs. ELT webinar** to see these differences in action with Matillion ETL.

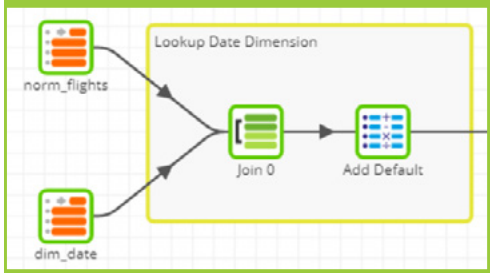
	ETL	ELT
CALCULATIONS	Calculations can be expressions or applied functions within the ETL server itself. Modern ETL tools are fairly quick with smaller datasets but performance may vary when dealing with large volumes. Calculations may result in overwriting existing columns or a new derived column appended to the data set and pushed to the target platform.	Bring raw data into the target database and then easily add a calculated/derived column to existing data. Some ELT tools, such as Matillion ETL have pre-built components to make calculations quick and easy to set-up and allow you to specify SQL expressions compatible with your target platform to drive your calculations.

What it looks like in Matillion:



LOOKUPS (JOINS)

What it looks like in Matillion:



ETL

To perform a lookup, ETL would go row by row to map a fact value to dimension key from another table or source. An API lookup or execute function would bring back a key which is then appended to the data and pushed to target. Usually this works okay, but there is a constraint of needing both facts and dimensions or whatever sources are helping you with this data to be available at that point in time. Another challenge is the amount of data you're working with when performing lookups. If the dimension table is really big you might have to partially or fully cache the data set. Performance is dependent on the capability of your ETL server and the options provided by your ETL tools.

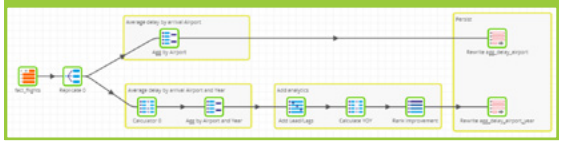
ELT

In an incremental scenario where a fact table is being added to new data on a regular basis, in all but the simplest scenarios it's necessary to join back to the complete fact or a dimension to perform valuable analytic business calculations such as ranking data or identifying outliers. In ETL this involves large and complex lookups that need to be done in memory and over the network. By contrast, in ELT flows this computationally heavy task can be delegated to an MPP database engine.

With ELT you can marry fact table records with appropriate dimension keys. Again this is implemented using SQL and typically a Left Join in order to find matches. All the data you need for your join is already present since it was extract and loaded previously. ELT differs from ETL in that MPP platforms are designed and optimized for handling large quantities of data by crunching individual transactions in parallel. This inevitably leads to faster processing times to populate your data warehouse.

AGGREGATIONS

What it looks like in Matillion:



These are very tricky in the ETL world, especially if you want to keep granular and aggregated data. You will end up with multiple stores of the same dataset with different granularity levels. Aggregations are further complicated by very large datasets. Performing aggregations on the ETL server can be very expensive and you may need sufficiently powerful ETL servers to handle large datasets. Some ETL tools allow you to perform pushdowns where possible, but require a lot of hand holding and manual coding and is a departure from how ETL usually works.

By loading the data first you can then use the capability and power of the target platform to apply transformations. You can easily multiplex to use the same input with different transformation-flows or use tables with transformed data from previous jobs to build complex workflows on large datasets. You can write the table(s) that result from the aggregation to storage platforms like S3 or Google Cloud Storage. The table(s) can then be imported into another database/ data lake if you so require.

	ETL	ELT
ANALYTICS FUNCTIONS	Analytic functions return an aggregate value that is somehow related to the current record. This implies that all such records relating to the current record are available for processing. Often however, an ETL process only sees a single batch of data, which means the analytic function needs to be done by the data warehouse (which has all records - apart from the ones in the current batch which haven't been loaded yet!). For example, a Year-to-date value, or Lifetime value metric may require a summary over thousands of records not in the current batch-window. This can be incredibly expensive and time-consuming processing. **We have seen ETL users tie themselves in knots trying to engineer solutions to analytic function problem in ETL without running out of memory, CPU or some other resource.**	Analytic functions are a major feature of all good MPP analytic databases and due to the parallel nature of the database engine and the columnar way the data is stored tend to be an extremely fast and powerful way to extract analytic meaning from raw data sets by placing each record into the context of its related records. Having all of the data immediately queryable meaning there are no limitation on the size of the context you can see the data in. For example, when ranking a datapoint within a category the database engine must first sort the entire category and apply the rank, as the size of the category is unknown. It's a huge advantage to be able to rapidly sort all or the data set.
UNSTRUCTURED INPUT DATA	Since ETL can essentially do anything, it's ideal for situations in which the input data is so unstructured it simply can't be loaded into a database without some initial transformation or parsing. The number of data sources that fit this category, however, are shrinking all the time, as technology shifts away from legacy binary formats, and is now standardizing around well-defined text formats such as JSON. JSON is semi-structured, and can indeed be loaded into many analytic databases directly and further analyzed/processed.	Because ELT relies on having the data in a relational form this makes dealing with proprietary formats or unstructured data more difficult. Modern data warehouses are making strides to deal with these kind of data sets. It can, however, still be necessary to pre-process certain formats to "relationalize" them prior to ingestion into a relational MPP database.

Conclusion

CONCLUSION

On-premise data warehouses are quickly being eclipsed by their cloud-based counterparts catching the attention of data-driven companies. The migration from on-premise to the cloud should also trigger a switch from ETL to ELT which is specifically designed for the advanced technology.

At Matillion, we believe that ELT is the best architecture for the majority of modern cloud data warehouses. Cloud-based data warehouses and ELT, hand-in-hand, offer superior scalability, improved performance and lower costs when compared to on-premise databases and ETL. When used in combination with data interpretation and relationalization services, this end-to-end approach enables you to make the most out of your data.

Try Matillion ETL out today with a free trial or test drive on Amazon Redshift, Google BigQuery or Snowflake.