5 REASONS ETL IS THE WRONG APPROACH FOR MAINFRAME DATA MIGRATION

ETL (extract, transform, and load) is the wrong approach for mainframe data migration because it was built for structured, database-to-database transfers—not the flexible, cloud-ready data movement that modern mainframe environments require. ETL's complexity, labor demands, processing costs, and inability to handle unstructured data make it a poor fit for organizations pursuing mainframe data migration today. ELT (extract, load, and transform), which moves raw data to its destination first and transforms it there, is better aligned with how mainframe modernization actually works.

Change is good—a familiar mantra, but one not always easy to practice. When it comes to moving toward a new way of handling data, mainframe organizations, which have earned their keep by delivering the IT equivalent of corporate-wide insurance policies (rugged, reliable, and risk-averse), naturally look with caution on new concepts like extract, load, and transform (ELT).

Positioned as a lighter and faster alternative to more traditional data handling procedures such as extract, transform, and load (ETL), ELT definitely invites scrutiny. And that scrutiny can be worthwhile.

Definitions provided by SearchDataManagement.com say that ELT is "a data integration process for transferring raw data from a source server to a data system (such as a data warehouse or data lake) on a target server and then preparing the information for downstream uses." In contrast, another source defines ETL as "three database functions that are combined into one tool to pull data out of

one database and place it into another database."

The crucial functional difference is that ETL focuses exclusively on database-to-database transfer, while ELT is open-ended and flexible. In the mainframe world, ETL is a tool with a more limited focus—ELT is focused on jump-starting the future.

Why does ETL fail for mainframe data migration?

ETL falls short across five key dimensions: complexity, labor intensity, processing bottlenecks, structural rigidity, and high processing costs. Here is a closer look at each.

1. ETL is too complex

ETL was not originally designed to handle all the tasks it is now being asked to do. In the early days, ETL was often applied to pull data from one relational structure and fit it into a different relational structure—including cleansing the data along the way.

For example, a traditional relational database management system (RDBMS) can get befuddled by numeric data where it is expecting alpha data, or by the presence of obsolete address abbreviations. ETL is optimized for that kind of painstaking, field-by-field data checking, "cleaning," and movement—but not for feeding a Hadoop database or modern data lake. ETL was not invented to take advantage of all the ways data originates and all the ways it can be used today.

2.ETL is labor-intensive

Because the ETL process is built around transformation, everything depends on the timely completion of that transformation step. With larger amounts of data in play—think Big Data—the required transformation times can become inconvenient or impractical, turning ETL into a functional and computational bottleneck.

3.ETL demands structure

Because the ETL process is built around transformation, everything depends on the timely completion of that transformation step. With larger amounts of data in play—think Big Data—the required transformation times can become inconvenient or impractical, turning ETL into a functional and computational bottleneck.

4.ETL demands structure

ETL is not designed for unstructured data and can add complexity rather than value when asked to handle it. ETL works best for traditional databases but does not help much with the massive waves of unstructured data that organizations need to process today.

5.ETL has high processing costs

ETL is especially challenging for mainframes because mainframe workloads generally incur MSU (million service unit) processing charges—burdening systems that need to be handling real-time workloads at the same time. ELT, by contrast, can be accomplished using mostly the capabilities of built-in zIIP engines, which cuts MSU costs, with additional processing handled at a chosen cloud

destination. In response to ETL's high processing costs, many organizations have already moved the data transformations stage to the cloud to support analytics workloads and data lake creation.

Why is ELT the better path forward for mainframe organizations?

It would be wrong to oversimplify a decision between ETL and ELT—there are too many moving parts and decision points to weigh. But the key insight is this: ELT speaks to the evolving IT paradigms that ETL was never built to address.

ELT is ideal for moving massive amounts of data to the cloud—typically to a data lake built to ingest any and all available data so that modern analytics can get to work. That is why ELT is growing and making inroads specifically in the mainframe environment. ELT represents perhaps the best path to accelerating mainframe data movement to the cloud at scale, making ELT a key tool for IT organizations aiming at modernization and at maximizing the value of their existing investments.

Frequently asked questions

What is ETL and why has it been used in mainframe environments?

ETL (extract, transform, and load) is a data integration method that pulls data from a source, transforms it to match a target schema, and loads it into a destination database. Mainframes have historically relied on ETL because it was well-suited to the structured, RDBMS-to-RDBMS data movement that dominated before cloud and big data workloads became standard.

What is the difference between ETL and ELT?

ETL transforms data before loading it into the destination system. ELT loads raw data to the destination first and transforms it there, leveraging the destination system's processing power. ELT is more flexible and better suited to modern cloud environments and data lakes than ETL.

Why does ETL become a bottleneck for mainframe data migration?

ETL requires all transformation to complete before data can move to its destination. At the scale of modern mainframe data volumes, this dependency on sequential transformation creates delays that make ETL impractical. ELT avoids the bottleneck by separating the load and transform steps entirely.

What is a zIIP engine and how does it reduce mainframe processing costs?

A zIIP (IBM Z Integrated Information Processor) is a specialty engine on IBM mainframes designed to offload eligible workloads from general-purpose processors, reducing MSU charges. ELT workloads are often eligible for zIIP processing, making ELT significantly more cost-efficient than ETL for mainframe data migration projects.

When does ETL still make sense over ELT?

ETL remains appropriate for structured, database-to-database migrations where schema alignment and data quality transformation must happen before the data reaches its destination—particularly when data volumes are manageable and the target system requires pre-transformed data. For large-scale, cloud-bound mainframe modernization, ELT is generally the better choice.

The views and opinions expressed in this post are those of the author and do not necessarily reflect the official position of BMC.