TECHNICAL WHITEPAPER

Migrating from Oracle to MemSQL

October 2017

MEMSQL

Migrating from Oracle to MemSQL

This is a no-nonsense guide to migrate from Oracle RDBMS to MemSQL.

The paper focuses on workloads well-suited for migration, and includes estimates on migration time and effort.

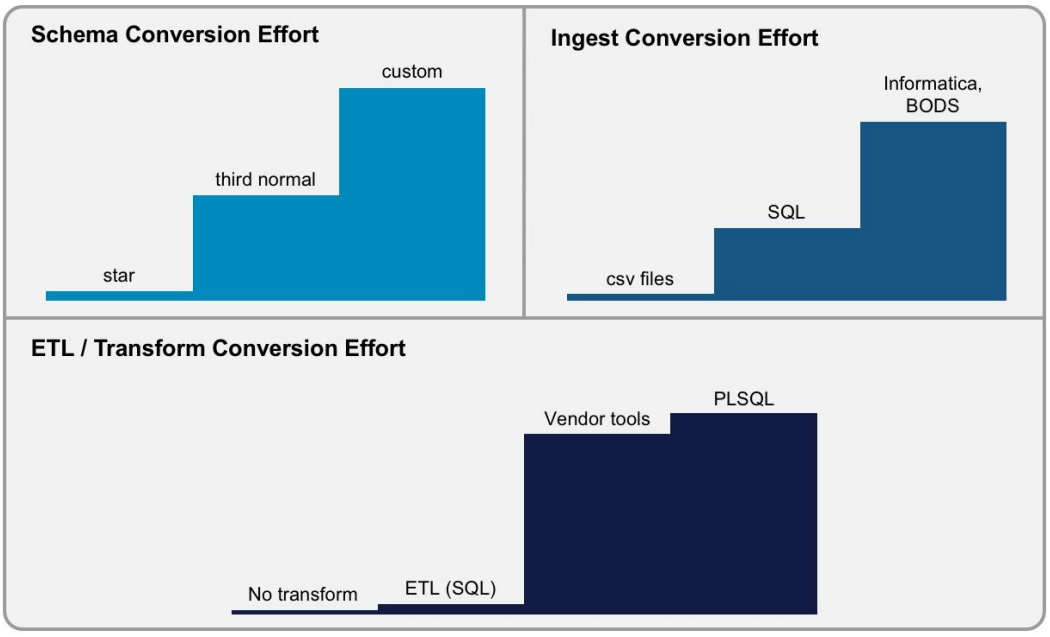
Identifying Workload Migration Candidates

The best candidates for migration are data analytics workloads. This does not include pre-packaged applications like ERP, Peoplesoft, and Siebel which are not recommended for migration. Ideal workloads are those custom developed within the overall Oracle ecosystem. This could include ingesting data from packaged applications, a custom stream of data ingest, or a combination of both.

MemSQL supports standard ANSI SQL with table creation and object creation compliant with the MySQL wire protocol. There are a few extensions to support distributed tables.

Analytical and Data Warehousing Workloads

The ETL methods for ingesting, transforming and structuring the data relate directly to the overall time for migrations. The charts below indicate the effort required to migrate various applications based on the schema, ingest method, and transformation method. This is a high level evaluation of the effort based on experience from existing MemSQL customers.



The more reliance on vendor tools, packaged application schemas, and PL/SQL, the more effort required to port the application.

Transactional Workloads

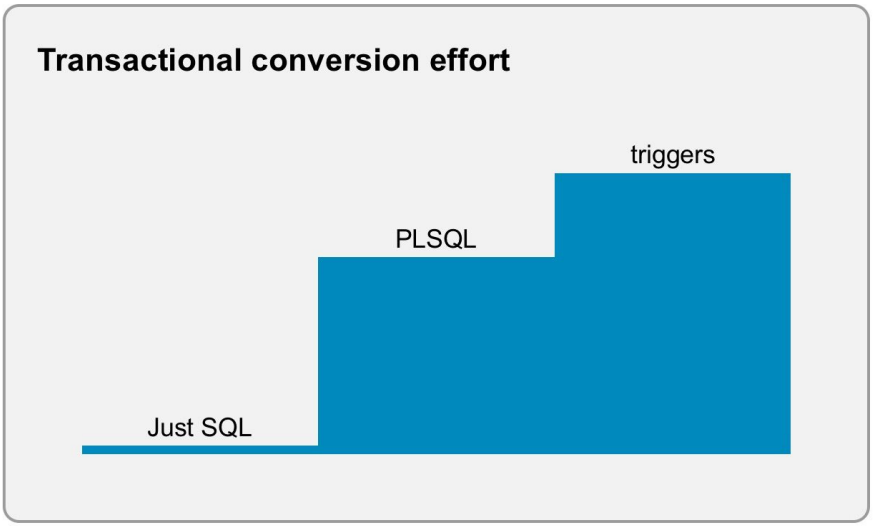
Vendor packaged applications are not ideal candidates for migration. However, for transactional workloads, the best fit are applications that stream sampled data in real time, such as IoT sensor readings financial trades, or asset tracking.

In particular when migrating from Oracle, the following situations require more time, planning and potential development work. They can be complex, but ultimately provide high reward. Be sure to dedicate enough of a planning cycle and focus on parallel deployments to ensure success.

Effort-intensive migrations

- Databases with heavy reliance on stored procedures. While they can be migrated, the custom code requires time and energy to move.
- Databases embedded deep within Oracle ERP systems (e.g. Peoplesoft) can require more effort as they can be part of a locked-down, proprietary solution.
- Databases connected to current critical path infrastructure should be migrated after a successful approach focused on adding analytics to an existing application.
- Databases connected to applications with significant trigger logic. This may require logic to be re-written outside of MemSQL.

The newest version of MemSQL includes extensibility across stored procedures, user-defined functions, and user-defined aggregates. In some cases, customers will be able to rework certain procedures into the MemSQL extensibility language MPSQL, which stands for Massively Parallel SQL.



Technical Approach for Database Migrations

This section addresses the steps required to port a database from Oracle to MemSQL. There are certainly numerous potential methods, and we will try illustrate simple options for a better understanding of requirements to use MemSQL.

Assess the Data Model

MemSQL flexibly handles all traditional data models. For data warehouse workloads MemSQL supports a model popularized by Ralph Kimball focused on star and snowflake schemas. Oracle also supports these data models. So, what really matters when porting the existing Oracle data model to MemSQL?

First, you must understand the usage as well as the data structure within Oracle. Oracle allows you to create various indexes and materialized views to improve the performance of aggregates. Additionally, Oracle allows you to create in-memory columnar and Hybrid Columnar Compression (HCC) with Exadata.

Finally, Oracle also allows you to partition data and avoid accessing certain partitions based on the partitioned schema. By far the most common partitioning is by DATE to eliminate scanning data based on certain date ranges. You should have a comprehensive understanding of the schema in place, and understand the reasons a certain path was chosen.

Review Data Types

Oracle supports a wide variety of data types. All of the typical integer, character, and date types have fairly straightforward mappings. If you use the Oracle SQL developer tool, you can translate data types from Oracle to MySQL and these should work directly with MemSQL. Please note this will not result universally in query compatibility. There are a few outliers that will require some changes, particularly around LOB, CLOB, NCLOB, and similar statements.

MemSQL has a LONGBLOB datatype that can store up to a 4GB object and Oracle will allow much larger LOBs up to 128TB. That said, these are very rarely used and unfortunately often mis-used. In all cases, review the data types used within your application for outliers. Both the MemSQL and Oracle documentation have plentiful details regarding data types and usage.

Porting the schemas

Oracle does not use databases in the exact same way that MemSQL does. For example, Oracle schemas were typically mapped to MemSQL databases. MemSQL 6.0 supports the joining of tables across Databases so the schema to database mapping is a reasonable approach. Earlier releases of MemSQL do not allow for cross database joins. In this case, tables that are to be joined, need to be consolidated in one database.

Porting the tables

Some migration tools are highlighted in the Migration Tools section of this paper. These tools are useful for porting over the data types, indexes, and views; but they do not address the distributed nature of MemSQL or the mapping of tables to in-memory rowstore tables or columnar tables. The table designation is the important to ensure the best query performance for your particular workload. So, how do you decide?

Row store tables

Rowstore tables are good for seeks and concurrent updates. It keeps all the data in memory and all the columns for a given row together, resulting in very fast performance when running queries that seek to specific rows. With rowstore, data is stored in lock-free indexes for great performance at high concurrency. Rowstore tables can use multiple indexes, allowing it to flexibly support many types of queries. Transactional style OLTP or IoT workloads where table data is frequently updated row-by-row are good candidates for row store.

Column store tables

MemSQL is able to efficiently load data via pipelines or CSV files into column store tables. It does so by loading batches in-memory and writing columnar segments to disk. Typically, Analytical workloads where data is loaded but not updated are a good fit for Column store tables.

If the application ETL transforms the data by updating certain columns, consider altering the data during import. Column store tables are not meant to be updated directly, so using ELT style ingest is considered the best approach if transformations are necessary. This process transforms data in temporary row store tables before storing the final result to a column store FACT table.

Picking the shard key

The shard key simply tells the MemSQL database how to distribute data in a particular table. The goal is to pick the key that has high cardinality and is most used for joins. For a transactional workload, this will essentially push the work to where the data resides. This provides the best performance and scalability by reducing the number of nodes in a cluster active for an individual transaction.

For an analytical workload, it also makes sense to use a high cardinality column that is also used for joins. MemSQL will push down the joining of table data to all of the active leaf node partitions. This is very efficient as all the join work is localized on each leaf node where the database partitions reside. If you join on a column that is not the shard key, MemSQL will perform a distributed join. This involves moving data across various leaf nodes as needed to satisfy the join criteria. This causes additional network load that must be processed for the join to complete.

Often, joins are performed on multiple columns and this cannot always be avoided. For these cases, it makes sense analyze the workload to find which queries run most frequently. Oracle's AWR and ASH reports can be used to find the frequency of execution over various periods.

In summary, shard keys are specified at the time of create table. Once created, the shard key for a table cannot be modified. You can always create a new table and migrated the data, but this may not be ideal given the situation. There are two main considerations when sharding data:

- distributing data evenly across partitions
- partitioning data on columns you most frequently join

Picking the columnstore key

Column store tables have an additional key known as the clustered columnstore key. When data is ingested into columnstore tables, segments for each column are sorted written in batches. These batches store the min/max values for rows within a segment. By doing this, we can scan metadata to eliminate rows that do not contain the required data. This technique can save a large amount of time as scanning and filtering data can be a resource intensive activity.

For example, consider a table that has 10 billion rows representing 10 years of data. Most of the queries however, analyze data on a yearly basis. If you pick a column that stores the date as the primary sort criteria, you can avoid looking at 90% of the data.

Translate queries to new database SQL syntax

MemSQL is ANSI SQL-92 compliant. Luckily, Oracle supports ANSI joins as well. The things you will need to watch out for is the proprietary Oracle syntax and functions that do not comply with ANSI SQL-92. Below are some examples of typical non-ANSI compliant Oracle SQL:

- LEFT OUTER JOINS can be made using a (+) syntax on the join criteria. The Oracle product manager discusses this translation in a blog entry.
- DECODE function is used, but this is essentially a CASE statement.
- MERGE statement in Oracle is not supported in MemSQL. This is essentially an UPSERT where you UPDATE the primary key values and INSERT new rows. This can be done in MemSQL by creating a table that automatically updates rows duplicate entries.

Tuning phase

After porting the queries and migrating the data, it is time for some tuning. This is typically an iterative process of analyzing performance and making various changes to the application, data model, tables, and queries. There are a number of best practices to follow when tuning MemSQL. The details of this are out-of-scope for this Whitepaper.

Migration tools

There are various tools to assist in the migration process. No one tool can handle the entire migration, but they can provide a head start down a successful path to MemSQL. Various tools and methods are discussed below with regards to schema migration, extraction methods, and change data capture.

Schema

Various migration tools exist to migrate to MySQL. MemSQL is 100% compatible with MySQL syntax which should make that an easy migration path. This helps you to quickly create most of the tables and just adjust the sharding and columnstore keys where needed.

Some of the tools for Schema migration include:

- Amazon Web Services Schema Conversion Tool : SCT
- Oracle Data Integrator : ODI
- Oracle SQL developer

Migrating the data

To extract data from Oracle to load into MemSQL, there are several tools and scripts you can write extract data to CSV files. Once you have CSV files, you can easily use the LOAD DATA command to load a CSV file. MemSQL also has the ability to create pipelines that point to a directory structure and load massive amounts of data in parallel. Once files arrive in this structure, they will be automatically loaded into the database. Creating CSV files from your traditional database is simple, but the main downside is you will require extra space to house the extracted data. The extra space can be an unmanageable solution if you are transferring upwards of 100TB of data.

To transfer data without the making a second copy, you can use tools that do change data capture (CDC), Informatica, GoldenGate, Shareplex, or Striim. These tools are commercially licenses, but if you already own and use them, then this path makes sense. If you do not own these tools, then you might want to consider using Spark to assist in the migration and transformation.

Conclusion

Tackling database migrations is an endeavor that with the right planning and execution can provide incredible technical and business benefits. In particular as systems move from traditional single node architectures to massively parallel distributed architectures MemSQL, with its full relational ANSI SQL capabilities, is an ideal candidate.

For more information, or to talk to a MemSQL specialist about your database migration, please contact us at info@memsql.com