BACK TO OFFICE /AFTER TRAINING REPORT

Title	**: RPTIS Data Migration Training**
Date	**:** On June 20-24, 2022
Venue	**:** One Central Hotel, Corner Leon Kilat, Sanciangko St, Streets, Cebu City, Cebu
Sponsored by	**:** RAMESES SYSTEM INC
Submitted by	:

FERDINAND MARCREY V. IRIG - Local Assessment Operations Officer IV – PASSO
NIÑO P. LANOY - Planning Officer II - BICTO
ERICKSON M. CRESCENCIO – Disbursing Officer II - PTO
PETER T. VALE – Assessment Clerk II - PASSO

I. INTRODUCTION/BACKGROUND

About

TRACS is a free software available to local government units to computerize tax and revenue assessments and collection. Designed,developed and maintained by Rameses Systems Inc, the project aims to seamlessly and fully integrate real property system, business licensing and all other revenue activities to treasury operations. Our mission is to provide an effective management tool to enforce discipline and best practice in LGU operations; one that can be easily availed especially for LGUs that cannot afford such systems. E-TRACS can be availed through grants, LGU to LGU assistance, or directly applying for membership in the community.

RPTIS or Real Property Information System a software used by the assessor's Office for logging Real Property holdings and transaction in the local government unit.

Database migration is the process of migrating data from one or more source databases to one or more target databases by using a database migration service. When a migration is finished, the dataset in the source databases resides fully, though possibly restructured, in the target databases. Clients that accessed the source databases are then switched over to the target databases, and the source databases are turned down.

Terminology

The most important data migration terms for these documents are defilned as follows:

source database: A database that contains data to be migrated to one or more target databases.

target database: A database that receives data migrated from one or more source databases.

database migration: A migration of data from source databases to target databases with the goal of turning down the source database systems after the migration completes. The entire dataset, or a subset, is migrated.

homogeneous migration: A migration from source databases to target databases where the source and target databases are of the same database management system from the same provider.

heterogeneous migration: A migration from source databases to target databases where the source and target databases are of different database management systems from different providers.

database migration system: A software system or service that connects to source databases and target databases and performs data migrations from source to target databases.

data migration process: A configured or implemented process executed by the data migration system to transfer data from source to target databases, possibly transforming the data during the transfer.

database replication: A continuous transfer of data from source databases to target databases without the goal of turning down the source databases. Database replication (sometimes called database streaming) is an ongoing process.

Partial versus complete migration

Database migration is understood to be a complete and consistent transfer of data. You define the initial dataset to be transferred as either a complete database or a partial database (a subset of the data in a database) plus every change committed on the source database system thereafter.

Heterogeneous migration versus homogeneous migration

A homogeneous database migration is a migration between the source and target databases of the same database technology, for example, migrating from a MySQL database to a MySQL database, or from an Oracle® database to an Oracle database. Homogeneous migrations also include migrations between a self-hosted database system such as PostgreSQL to a managed version of it such as Cloud SQL (a PostgreSQL variant).

In a homogenous database migration, the schemas for the source and target databases are likely identical. If the schemas are different, the data from the source databases must be transformed during migration.

Heterogeneous database migration is a migration between source and target databases of different database technologies, for example, from an Oracle database to Spanner. Heterogeneous database migration can be between the same data models (for example, from relational to relational), or between different data models (for example, from relational to key-value).

Migrating between different database technologies doesn't necessarily involve different data models. For example, Oracle, MySQL, PostgreSQL, and Spanner all support the relational data model. However, multi-model databases like Oracle, MySQL, or PostgreSQL support different data models. Data stored as JSON documents in a multi-model database can be migrated to MongoDB with little or no transformation necessary, as the data model is the same in the source and the target database.

II. TRAINING OBJECTIVES

At the end of this course, you will be able to:

- Develop strategy and scripts in migrating RPTIS Data to ETRACS
- Merge Municipal databases into a Single Provincial Database
- Design an implementation plan on how to operationalize ETRACS between Province and Municipalities

Migration Goals:

- Produce a UNIFIED master data and SMV in the province
- Merge a COMPLETE FAAS Data in the province

IMPORTANT:

- Master data and SMV must be maintained by the province and synchronized by the municipalities

III. COURSE OUTLINE

Day 1	Day 2	Day 3
8am TO 6pm	8am to 6pm	8am to 6pm
I. ETRACS Server Setup and Configuration II. Database Restoration	I. Master data and SMV Migration	I. Land and Plant/Tree FAAS Migration
Day 4	**Day 5**	
8am TO 6pm	8am to 6pm	
I. Building and Machinery FAAS Migration	I . Migration of Annotations, Restrictions, etc. Real Tax Ledger Migration	

IV. INSIGHTS AND LEARNINGS

Switching clients from the source databases to the target databases involves several processes:

To continue processing, clients must close existing connections to the source databases and create new connections to the target databases. Ideally, closing connections is graceful, meaning that you don't unnecessarily roll back ongoing transactions.

After closing connections on the source databases, you must migrate remaining changes from the source databases to the target databases (called draining) to ensure that all changes are captured.

You might need to test target databases to ensure that these databases are functional and that clients are functional and operate within their defined service level objectives (SLOs).

You can configure the amount of data being migrated (that is, in flight between the source and target databases) to be as small as possible when the switch over period approaches. This step reduces the time for draining because there are fewer differences between the source databases and the target databases.

While it's unrealistic to achieve zero downtime during a switch over, you can minimize the downtime by starting activities concurrently with the ongoing data migration when possible.

In some database migration scenarios, significant downtime is acceptable. Typically, this allowance is a result of business requirements. In such cases, you can simplify your approach.

For example, with a homogeneous database migration, you might not require data modification; export/import or backup/restore are perfect approaches. With heterogeneous migrations, the database migration system does not have to deal with updates of source database systems during the migration.

However, you need to establish that the acceptable downtime is long enough for the database migration and follow-up testing to occur. If this downtime cannot be clearly established or is unacceptably long, you need to plan a migration that involves minimal downtime.

Database migration cardinality

In many situations database migration takes place between a single source database and a single target database. In such situations, the cardinality is 1:1 (direct mapping). That is, a source database is migrated without changes to a target database.

A direct mapping, however, is not the only possibility. Other cardinalities include the following:

Consolidation (n:1). In a consolidation, you migrate data from several source databases to a smaller number of target databases (or even one target). You might use this approach to simplify database management or employ a target database that can scale.

Distribution (1:n). In a distribution, you migrate data from one source database to several target databases. For example, you might use this approach when you need to migrate a large centralized database containing regional data to several regional target databases.

Re-distribution (n:m). In a re-distribution, you migrate data from several source databases to several target databases. You might use this approach when you have sharded source databases with shards of very different sizes. The re-distribution evenly distributes the sharded data over several target databases that represent the shards.

Database migration provides an opportunity to redesign and implement your database architecture in addition to merely migrating data.

Migration consistency

The expectation is that a database migration is consistent. In the context of migration, consistent means the following:

Complete. All data that is specified to be migrated is actually migrated. The specified data could be all data in a source database or a subset of the data.

Duplicate free. Each piece of data is migrated once, and only once. No duplicate data is introduced into the target database.

Ordered. The data changes in the source database are applied to the target database in the same order as the changes occurred in the source database. This aspect is essential to ensure data consistency.

An alternative way to describe migration consistency is that after a migration completes, the data state between the source and the target databases is equivalent. For example, in a homogenous migration that involves the direct mapping of a relational database, the same tables and rows must exist in the source and the target databases.

This alternative way of describing migration consistency is important because not all data migrations are based on sequentially applying transactions in the source database to the target database. For example, you might back up the source database and use the backup to restore the source database content into the target database (when significant downtime is possible).

Data migration process

The core technical building block of a database migration system is the data migration process. The data migration process is specified by a developer and defines the source databases from which data is extracted, the target databases into which data is migrated, and any data modification logic applied to the data during the migration.

You can specify one or more data migration processes and execute them sequentially or concurrently depending on the needs of the migration. For example, if you migrate independent databases, the corresponding data migration processes can run in parallel.

Data extraction and insertion

You can detect changes (insertions, updates, deletions) in a database system in two ways: database-supported change data capture (CDC) based on a transaction log, and differential querying of data itself using the query interface of a database management system.

CDC based on a transaction log

Database-supported CDC is based on database management features that are separate from the query interface. One approach is based on transaction logs (for example the binary log in MySQL). A transaction log contains the changes made to data in the correct order. The transaction log is continuously read, and so every change can be observed. For database migration, this logging is extremely useful, as CDC ensures that each change is visible and is subsequently migrated to the target database without loss and in the correct order.

CDC is the preferred approach for capturing changes in a database management system. CDC is built into the database itself and has the least load impact on the system.

Data modification

In some use cases, data is migrated from source databases to target databases unmodified. These straight-through migrations are typically homogeneous.

Many use cases, however, require data to be modified during the migration process. Typically, modification is required when there are differences in schema, differences in data values, or opportunities to clean up data while it is in transition.

Data transformation

Data transformation transforms some or all data values from the source database. Some examples include the following:

Data type transformation. Sometimes data types between the source and target databases are not equivalent. In these cases, data type transformation casts the source value into the target value based on type transformation rules. For example, a timestamp type from the source might be transformed into a string in the target.

Data structure transformation. Data structure transformation modifies the structure in the same database model or between different database models. For example, in a relational system, one source table might be split into two target tables, or several source tables might be denormalized into one target table by using a join. A 1:n relationship in the source database might be transformed into a parent/child relationship in Spanner. Documents from a source document database system might be decomposed into a set of relational rows in a target system.

Data enrichment and correlation

Data transformation is applied on the existing data without reference to additional, related reference data. With data enrichment, additional data is queried to enrich source data before it's stored in the target database.

Data correlation. It is possible to correlate source data. For example, you can combine data from two tables in two source databases. In one target database, for instance, you might relate a customer to all open, fulfilled, and canceled orders whereby the customer data and the order data originate from two different source databases.

Data enrichment. Data enrichment adds reference data. For example, you might enrich records that only contain a zip code by adding the city name corresponding to the zip code. A reference table containing zip codes and the corresponding city names is a static dataset accessed for this use case. Reference data can be dynamic as well. For example, you might use a list of all known customers as reference data.

Data reduction and filtering

Another type of data transformation is reducing or filtering the source data before migrating it to a target database.

Data reduction. Data reduction removes attributes from a data item. For example, if a zip code is present in a data item, the corresponding city name might not be required and is dropped, because it can be recalculated or because it is not needed anymore. Sometimes this information is kept for historical reasons to record the name of the city as entered by the user, even if the city name changes in time.

Data filtering. Data filtering removes a data item altogether. For example, all canceled orders might be removed and not transferred to the target database.

Schema-based addressing. Schema-based addressing determines the target database based on the schema. For example, all data items of a customer collection or all rows of a customer table are migrated to the same target database storing customer information, even though this information was distributed in several source databases.

Content-based routing. Content-based routing (using a content-based router, for example) determines the target database based on data values. For example, all customers located in the Latin America region are migrated to a specific target database that represents that region.

You can use both types of addressing at the same time in a database migration. Regardless of the addressing type used, the target database must have the correct schema in place so that data items are stored.

Persistence of in-transit data

Database migration systems, or the environments on which they run, can fail during a migration, and in-transit data can be lost. When failures occur, you need to restart the database migration system and ensure that the data stored in the source database is consistently and completely migrated to the target databases.

As part of the recovery, the database migration system needs to identify the last successfully migrated data item to determine where to begin extracting from the source databases. To resume at the point of failure, the system needs to keep an internal state on the migration progress.

You can maintain state in several ways:

You can store all extracted data items within the database migration system before any database modification, and then remove the data item once its modified version is successfully stored in the target database. This approach ensures that the database migration system can exactly determine what is extracted and stored.

You can maintain a list of references to the data items in transit. One possibility is to store the primary keys or other unique identifiers of each data item together with a status attribute. After a failure, this state is the basis for recovering the system consistently.

You can query the source and target databases after a failure to determine the difference between the source and target database systems. The next data item to be extracted is determined based on the difference.

Completeness and consistency verification

You need to verify that your database migration is complete and consistent. This check ensures that each data item is migrated only once, and that the datasets in the source and target databases are identical and that the migration is complete.

Depending on the data modification rules, it is possible that a data item is extracted but not inserted into a target database. For this reason, directly comparing the source and target databases is not a solid approach for verifying completeness and consistency. However, if the database migration system tracks the items that are filtered out, you can then compare the source and target databases along with the filtered items.

Replication functionality of the database management system

A special use case in a homogeneous migration is where the target database is a copy of the source database. Specifically, the schemas in the source and target databases are the same, the data values are the same, and each source database is a direct mapping (1:1) to a target database.

In this case, you can use functionality within the database management system to replicate one database to another. Replication only creates an exact copy; it does not perform data modification. Examples are MySQL replication, PostgreSQL replication (see also pglogical), or Microsoft SQL Server replication.

However, if data modification is required, or you have any cardinality other than a direct mapping, a database migration system's functionality is needed to address such a use case.

TOPIC	SPEAKER
Server and Database setup Schema and Table Dissection Data Transfer from MSSQL to MysQL Database Scripting Plugging error from Scripts	Engr. Jessei Sato Zamora,CpE Instructor VP- Software Solutions

V. SUMMARY

Some reasons for building database migration functionality instead of using a database migration system or database management system functionality include the following:

You need full control over every detail.

You want to reuse functionality.

You want to reduce costs or simplify your technological footprint.

Building blocks for building migration functionality include the following:

Export/import. If downtime is not a factor, you can use database export and database import to migrate data in homogenous database migrations. Export/import, however, requires that you quiesce the source database to prevent updates before you export the data. Otherwise, changes might not be captured in the export, and the target database will not be an exact copy of the source database.

Backup/restore. Like in the case of export/import, backup/restore incurs downtime because you need to quiesce the source database so that the backup contains all data and the latest changes. The downtime continues until the restore is completed successfully on the target database.

Error handling

Failures during database migration must not cause data loss or the processing of database changes out of order. Data integrity must be preserved regardless of what caused the failure (such as a bug in the system, a network interruption, a VM crash, or a zone failure).

A data loss occurs when a migration system retrieves the data from the source databases and does not store it in the target databases because of some error. When data is lost, the target databases do not match the source databases and are thus inconsistent and incomplete. The completeness and consistency verification functionality flags this state (Completeness and consistency verification).

Scalability

In a database migration, time-to-migrate is an important metric. In a zero downtime migration (in the sense of minimal downtime), the migration of the data occurs while the source databases continue to change. To migrate in a reasonable timeframe, the rate of data transfer must be significantly faster than the rate of updates of the source database systems, especially when the source database system is large. The higher the transfer rate, the faster the database migration can be completed.

Order violation. If scalability of the migration system is achieved by scaling out, then several data transfer processes are running concurrently (in parallel). Changes in a source database system are ordered according to committed transactions. If changes are picked up from the transaction log, the order must be maintained throughout the migration. Parallel data transfer can change the order because of varying speed between the underlying processes. It is necessary to ensure that the data is inserted into the target databases in the same order as it is received from the source databases.

Missing or duplicate data. When a failover occurs, a careful recovery is required if some data is not replicated between the primary and the failover replica. For example, a source database fails over and not all data is replicated to the failover replica. At the same time, the data is already migrated to the target database before the failure. After failover, the newly promoted primary database is behind in terms of data changes to the target database (called flashback). A migration system needs to recognize this situation and recover from it in such a way that the target database and the source database get back into a consistent state.

In general, there are other pitfalls to watch out for. The best way to find problems that might lead to data inconsistency is to do a complete failure analysis that iterates through all possible failure scenarios. If concurrency is implemented in the database migration system, all possible data migration process execution orders must be examined to ensure that data consistency is preserved. If high availability or disaster recovery (or both) is implemented, all possible failure combinations must be examined.

VI. RE-ENTRY

ACTIVITY	Start	End	Person Responsible
Finalize Migration Script	June 27, 2022	August 26, 2022	Rameses, BICTO, Province and Municipal IT
Data Migration Testing	August 29, 2022	Sept 2, 2022	BICTO, Province and Municipal IT
Assessor's User Training	Sept 5, 2022	Sept 9, 2022	BICTO, PASSO, MASO, IT
Pre-production Preparation	Sept 12, 2022	Sept 23, 2022	BICTO, PASSO, MASO, IT
Production Migration	Sept 26, 2022	Sept 29, 2022	BICTO, PASSO, MASO, IT
Rollout and Implementation	Sept 30, 2022	Sept 30, 2022	BICTO, PASSO, MASO, IT
Correction and Process Transaction of erroneous Data	Sept 30, 2022	December 31, 2022	MASSO, PASSO

VII. RECOMMENDATION

The training module was delivered by an experienced trainer and developer of the Enhanced Tax Revenue Assessment and Collection System (ETRACS). The skills and understanding gained from the training proved mandatory for technical persons managing ETRACS and RPTIS databases. Moreover, the training provides new knowledge and skills and is highly recommended for database administrators and ETRACS users in data manipulation.

Thus, we, recommend or suggest that to be updated with the current technology the Provincial Government should invest for the sustainability of this project by capacitating its users that will administer the system, and pursue unified system among Treasury and Assessor's alike as well as the infrastructure that is needed for the system. In line with, the best interest of the institution in delivering effective government services to its stakeholders.

Submitted by:

FERDINAND MARCREY V. IRIG, CpE, REA
Local Assessment Operations Officer IV - PASSO

NIÑO P. LANOY, CpE , C[VA], CFOT, CFOSO
Planning Officer II - BICTO

ERICKSO M. CRESCENCIO
Disbursing Officer II - PTO

PETER T. VALE
Assessment Clerk II - PASSO

Approved by:

SENEN S. BOJOS, C [VA], CFOT
Head – BICTO

EUSTAQIOU A. SOCORIN, CPA
Provincial Treasurer

ROGELIO VILLARIAS, REA
Officer-In-Charge,
Provincial Assessor's Office

JOSEFINA J. RELAMPAGOS
HRMDO