International Journal of Modern Electronics and Communication Engineering (IJMECE) ISSN: 2321-2152
Volume No.-7, Issue No.-4, July, 2019

# Data Migration: Types and Challenges

Stuti Nareshkumar Garg
Computer Science and Engineering
Babaria Institute Of Tecnhnology
Vadodara, India
E-mail: gargstutee@gmail.com

Pratibha Bhahulkar
Supercomputing Facility, Computer Division
Bhabha Atomic Research Center
Mumbai, India
E-mail: pbahulkar@gmail.com

Prof. Ketan Rathod
Computer Science and Engineering
Babaria Institute Of Tecnhnology
Vadodara, India
E-mail:ketanrathod.ce@bitseducampus.ac.in

Prof. Amita Garg
Parul Institute of Management and Research
Parul University
Vadodara, India
E-mail:amita.garg@paruluniversity.ac.in

**Abstract:** Database migration has been an ongoing issue since data has been collected. Inevitably, new systems are designed which are intended to be more efficient, user friendly, inclusive, and current than older systems. In some cases, new systems are developed simply to replace older systems, and the justification may be new software trends, financial constraints or windfalls, new management initiatives, user complaints, or other reasons. In other some instances, new systems are developed to replace older ones that have become too complex or outdated and which resist further modification and evolution. This paper deals with the process of data migration in detail, along with the types and the challenges faced during migration and primarily on cross platform migration

**Keywords:** Data migration, Types of data migration, Challenges, Cross platform migration

## I. INTRODUCTION

In present days the area of data migration is very tropical. Data migration is defined as transferring data between computer storage types or file formats. It is the process of transferring data from one system to another while changing the storage, database or application. In reference to the ETL (Extract-Transform-Load) process, data migration always requires at least Extract and Load steps

It is usually performed programmatically to achieve an automated migration, freeing up human resources from tedious tasks. During data migration, software programs or scripts are used to map system data for automated migration. Data migration occurs for a variety of reasons, including server or storage equipment replacements, maintenance or upgrades, application migration, website consolidation and data canter relocation.[1]

Data migration is categorized as storage migration, database migration, application migration and business process migration.

## II. WHEN AND HOW IT OCCURS

Typically, data migration occurs during an upgrade of existing hardware or transfer to a completely new system. Examples include: migration to or from hardware platform; upgrading a database or migrating to new software; or company-mergers when the parallel systems in the two companies need to be merged into one. There are three main options to accomplish data migration:

1. Merge the systems from the two companies into a brand new one
2. Migrate one of the systems to the other one.
3. Leave the systems as they are but create a common view on top of them - a data warehouse. [2]

A typical data migration and merging project includes the following phases:

**1. Data and Schema Analysis** is understanding the environment where the data is stored and the schema used. It also involves studying the

**2. Data profiling** provides an indication of data quality, and includes information such as data type, length, value range, discrete values and their frequency, uniqueness, typical data pattern, etc.

**3. Data mining** helps discover integrity constraints and business rules. These can be used to correct illegal values, identify duplicate records and rebuild missing values across data sources.

**4. Schema review** through discussions with domain experts, or system support forums, to obtain insight into the definition, structure and identities within the source and target data. Where this is not possible, we rely on the data mining results.

**5. Define data formats, mapping rules and transformation workflows**: Results from Step 1 above are translated into a series of programmable instructions. Their purpose is to convert the data from its source to target format, plus resolve any associated data quality issues. These transformation steps may require user input where there is no applicable logic.

**6. Verification**: The accuracy and effectiveness of the transformation steps are evaluated on the source data and the transformational steps tweaked if required. Steps 2 & 3 may require multiple iterations since some errors may only become apparent after applying the initial transformations.

**7. Transformation**: The verified transformation steps are applied. Any errors requiring manual identification are identified and resolved.

**8. Export cleaned data and import into target system**: The cleaned and transformed data is exported and imported into the target system.

**9. Test**: Key reports from the target system are compared to those from the source system to ensure that the data was correctly transformed and uploaded. Significant errors may require a reiteration of Steps 1 to 6. [4]


Once the decision is made to perform data migration and before migration can begin the following analysis must be performed:

1. Analyze and define source structure (structure of data in the legacy system)

2. Analyze and define target structure (structure of data in the new system)

3. Perform field mapping (mapping between the source and target structure with data cleansing, if necessary)

4. Define the migration process (automated vs. manual)

To analyse and define source and target structures, analysis must be performed on the existing system as well as the new system to understand how it works, who uses it, and what they use it for. A good starting point for gathering this information is in the existing documentation for each system. This documentation could take the form of the original specifications for the application, as well as the systems design and documentation produced once the application was completed. Crucial information can also be found in other forms of documentation, including guides, manuals, tutorials, and training materials that end-users may have used. Most often this type of material will provide background information on the functionality exposed to end-users but may not provide details of how the underlying processes work. For this part of the analysis, you may actually need to review and document the existing code. This may be difficult, depending on the legacy platform. Undocumented code or fixes that are critical to the application and that have not been documented elsewhere may exist.

Another key area to examine is how the data in the system are stored (i.e., in flat files, files, or tables). What fields are included in those files/tables and what indexes are in use? Also, a detailed analysis of any server processes that are running that may be related to the data must be performed (e.g., if a mighty process runs across a file and updates it from another system).

Now that the source and target structures are defined, the mapping from the legacy to the target should fall into place fairly easily. Mapping should include documentation that specifically identifies fields from the legacy system mapped to fields in the target system and any necessary conversion or cleansing.

Once the analysis and mapping steps are completed, the process of importing the data into the new system must be defined. This process may be a combination of automation and manual processes or may be completely automated or may be completely manual. For example, a process may:

1. Create data extractions from the legacy system.
2. Cleanse the data extractions according to mapping guidelines
3. Import the data extractions into the new system using that system's import feature
4. Verify and test samplings of the data in the new system by comparing data reports to the old system. [5]

## III. TYPES OF DATA MIGRATION

The data migration process is performed on four levels.

1. **Storage migration** is justified through technology refreshes, and the process is used as an optimal time

to do data validation and reduction by identifying obsolete or corrupt data. The process involves moving blocks of storage and files from one storage system to another, whether it is on disk, tape or the cloud. There are numerous storage migration products and tools that help smooth the process Storage migration also offers the chance to remediate any orphaned storage or inefficiencies. [6]

2. **Database migration** is done when there is a need to change database vendors, upgrade the database software or move a database to the cloud. In this type of migration, the underlying data can change, which can affect the application layer when there is a change in protocol or data language. Data migrations in databases deal with modifying the data without changing the schema. Some key tasks include assessing the database size to determine how much storage is needed, testing applications and guaranteeing
data confidentiality. Compatibility problems can occur during the migration process, so it is important to test the process first. Following are the sub-categories of database migration: [6]

a) **Relational database migration** refers to the management of incremental, reversible changes to relational database schemas. A schema migration is performed on a database whenever it is necessary to update or revert that database's schema to some newer or older version.Migrations are performed programmatically by using a schema migration tool. When invoked with a specified desired schema version, the tool automates the successive application or reversal of an appropriate sequence of schema changes until it is brought to the desired state. [10]

3. **Cross Platform migration** is a variation of ordinary transportable tablespace. All of the restrictions that apply to transportable tablespaces apply here also, such as the need to ensure that all of the objects being transported are completely contained within the set of tablespaces being transported. Cross-platform transportable tablespace can be performed between platforms that have the same, or different, endian format (Endianness refers to the sequential order used to numerically interpret a range of bytes in computer memory as a larger, composed word value). [9]

a) **Heterogeneous and Homogeneous migration** If the target system doesn't exist and you are building a new system on the same o/s and database then this process is called as homogenous system copy. In case the o/s and database are different in your source and target SAP systems, then that process is called heterogeneous system copy or migration**.** [7]

b) **Application migration** can occur when switching to another vendor application or platform. This process has its own inherent layers of complexity because applications interact with other applications, and each one has its own data model. Applications are not designed to be portable. Management tools, operating systems and virtual machine configurations can all differ from those in the environment where the application was developed or deployed. Successful application migration may require the use of middleware products to bridge technology gaps. [6]

4. **Cloud migration** is a major technology trend, as the cloud provides on-demand flexibility, scalability and reduction of Capex for on-premises infrastructures. Public cloud providers offer a variety of services for storage, database and application migrations. [6]

## IV. CROSS PLATFORM MIGRATION

Cross platform data migration is the process of moving the tables in the databases of one platform to another. This consists of transferring static or pre-existing long-term retention data to a newer technology, and to describe it as complex may be an understatement. It is time consuming, labour intensive, logistically problematic and can incur substantial downtime if not done properly. The process behind cross platform data migration is converting the data in MS ACCESS, MS SQL Server, or Oracle to Extensible Markup Language (XML) and then transferring the XML file to Linux, where the data in XML form is converted to the form in which data can be stored on MYSQL Server using a parser form. The parsing part and extracting the data out of XML files results in exchange of data. This is possible since XML is platform independent.

The Cross-platform transportable database is not the same thing as transportable tablespace. In this case you are copying an entire database, including the SYSTEM tablespace from one platform to another. Containment checks are irrelevant, and because the SYSTEM tablespace is being copied, no export/import step is required. Cross-platform transportable database can only be performed between platforms that have the same endian format. [11][12]

You can transport tablespaces in a database that runs on one platform into a database that runs on a different platform.

Typical uses of cross-platform transportable tablespaces include the following:

- Publishing structured data as transportable tablespaces for distribution to customers, who can convert the tablespaces for integration into their existing databases regardless of platform
- Moving data from a large data warehouse server to data marts on smaller computers such as Linux-based workstations or servers
- Sharing read-only tablespaces across a heterogeneous cluster in which all hosts share the same endian format. [8]
- Migrating tablespaces across platforms with minimal application downtime.

Data migration is not something that occurs daily, and so the process often requires a temporary need for additional expertise, software or consulting and equipment. Administrators and application developers generally work with consultants to determine the criteria for migrating files, and a process is created which although is fairly straight forward can be quite involved.

A well-planned data migration process will involve the following:

- Research the current environment and the goals to determine the source-to-target data relationships that need to be migrated.
- Determine all pre-installation requirements.
- Audit all devices and CPUs to provide complete documentation of the network.
- Manage other venders who are crucial to the process.
- Develop a comprehensive project and test plan.
- Construction and deployment of all necessary equipment and software.
- Accurate refinement of the data migration process.

The process often requires additional refinements and the key to a successful data migration process is to make subsequent refinements more comprehensive than the last. Testing and validation to provide complete quality assurance of the data migrationprocess.

Although, as stated above, the process is complex, once it is completed a successful, data migration will provide numerous benefits including:

1. Reduction in cost.
2. Allows you to take advantage of newer technologies.
3. Provides effective utilization of primary storage disk space.
4. Automation of data life cycle management operations.
5. Allows you to virtualizes and consolidate storage resources
6. Reduces data management overhead.
7. Allows for comprehensive management of data across all departments and data center[11][12][13]

## V. CHALLENGES FACED WHILE MIGRATING

### 1. Poor Knowledge of Source Data

This knowledge gap includes not being aware of the problems that exist in your data, such as duplicates, missing information, misspellings and erroneous data. It can be all too easy to get complacent and assume that your data can easily be configured into the parameters of a new system however the reality could mean critical failures when it comes to user acceptance. So to ensure success, you need a good understanding of the source data.

### 2. Underestimating Data Analysis

Due to constraints in computer systems, information can be hidden in obscure places because often there aren't specific fields to hold all elements of the data or users may not be aware of the purpose of the available fields. This will result in incomplete, inaccurate and outdated information being transferred during the migration, often discovered very late in the day, even after the project has been completed. The outcome can mean not having enough time or the right resources to be able to identify and correct this data. Performing a thorough data analysis at the earliest possible occasion, usually when planning and designing your data migration can help you uncover these hidden errors.

### 3. Lack of Integrated Processes

Data migrations typically involve a disparate set of people using disparate technologies. The classic example is the use of spreadsheets to document data specifications, which is prone to human errors and cannot be easily translated when analysing data or performing data transformations. Using disparate technologies can lead to failure in the transfer of data and its design between the analysis, development, testing and implementation phases. Things can get lost in translation, resulting in increased costs and wasted time. Organizations must look to utilize a platform that successfully links the critical inputs and outputs from each of the stages to help reduce error and save time and money.

### 4. Inability to Validate a Specification

While you may well have an understanding of your source data, this will not necessarily result in a strong specification for migrating and modifying data into a target system. As this is early in the stage of the migration, critical misses can have repercussions later in the chain of activities. Validating your data transformation specifications early on with actual data,

rather than just documented aspirations can increase the confidence in executing the rest of the steps.

**5. Failure to Validate the Implementation**

Like before, where your knowledge of source data is evident, you can still hit a brick wall because of a lack of test cases. If you don't explore various scenarios, you run the risk of developing problems when it is often too late. Testing your migration using full volume data from the real world helps cover a wider range of possibilities and tests for the worst-case scenario, which could be missed when using more convenient samples of data.

**6. Late Evaluation of the Final Results.**

This problem can occur in the testing stage, where users only see the actual data that will be loaded into the new system at the end of the design and development. At this point, one of the worst outcomes can arise – incompatibility of the data in the new system. While an organization is capable of working without remedying the problem, this is not best practice. Time, money and the embarrassment of a delayed project can be avoided by introducing early and agile testing phases and getting your users involved in evolving the test cases as they see actual prototypes of the data output.

**7. Lack of Collaboration**

It has already been mentioned that data migrations involve disparate people, using different technologies, and in some cases a mix of internal employees and external contractors. Some of these people may not even be in the same location. Working in silos can reduce efficiency, create more data silos and sometimes lead to misinterpretations. Working together can be difficult and when things start to go wrong most try to avoid blame rather than resolving the issues. Collaborative tools enable all parties invested in a migration to see the same picture of data as it moves through the project stages, leaving little room for assumptions and misunderstandings.

**8. Inappropriate use of Expertise**

It makes sense to source experts, and usually this is applied to the management and technical aspects of a data migration. However, often the experts on data, usually hidden in the business do not make an appearance until late in the day. All too often those with access to data are unable to decode it, while those that can are unable to obtain access to it, sometimes until the new system is ready. Introducing data experts into your migration projects right from the beginning will ensure they make sense of the disparate data sources, but also guide the data transformation to suit the audience who will use it in the target system.

A data migration project is challenging and has substantial risk but if each of these hurdles are acknowledged during the planning stage and are overcome early on before data is transformed and transferred, you can be sure of success. [3]

## VI. CASE STUDY

While I was undergoing a summer internship at BARC, Mumbai, the organization 8mq0s-swas facing some problems with Oracle not being open source and hence access to the services on payable basis. There, I was given the task of migrating the local data of size of terabytes from Oracle to MySql. It is very true that the Sun MySQL database can dramatically reduce the database Total Cost of Ownership (TCO) for a company by lowering license, hardware and administration costs. The largest risk in moving to the MySQL platform is the risk and complexity of migrating business logic from Oracle, particularly when existing applications make significant use of PL/SQL procedures, triggers, packages and Oracle specific SQL statements.

I had faced the following issues while migrating the data:

**1. Subqueries are poorly optimized and complex queries are a weak point.**

MySQL 5.6 and below may encounter performance bottlenecks when processing subqueries associated with a large table because its optimizer only supports nested loop. The database version at the time was 5.5 and the original Oracle application had stored a large number of subqueries, so after migrating to MySQL, a significant number of SQL statements were accumulated which filled up the available connections and overloaded the database's CPU. We eventually had to modify the subqueries to recover the system. Typical subquery SELECT first_name FROM employees WHERE emp_no IN (SELECT emp_no FROM salaries_2000 WHERE salary = 5000); The MySQL processing logic involves traversing each record in an "employees" table and then substituting the records to a subquery

**2. The query executioner (aka query optimizer / planner) is less sophisticated.**

The Oracle optimizer contains more abundant and complete optimization algorithms than the MySQL optimizer. Just in terms of table connections, Oracle supports three algorithms, i.e., nested loop, hash join, and sort-merge join, while MySQL only supports nested loop. This means that MySQL often performs poorly when processing complex queries associated with large or multiple tables. How can we then identify which queries should not be migrated to MySQL?

Well, it can be determined based on the logical reads, physical reads.

**3. Int(…)**

The number of int(...) parameters in Oracle is limited to within 1000. Although MySQL has no limit in number, it has a limit in SQL length. Meanwhile, a MySQL optimizer uses binary search when optimizing in(...) queries, meaning that the more parameters in the query the worse performance will be, so the number of parameters should generally be kept within 100.

***4.Stored procedures and triggers are limited.***

If you need to convert PL/SQL code (procedures, packages, functions and triggers) or view/queries containing Oracle specific SQL syntax, you have to investigate what features are used and define the number of their occurrences. Examples of items that need to be accounted for are:

1. Non-ANSI compatible SQL functions, operators and statements
2. Results sets
3. Cursor loops
4. Exceptions
5. Temp tables
6. Object types and functions
7. Collections
8. Dynamic SQL

99. OLAP functions
110. Built-in packages
111. XML functions etc. [16]

MySQL stored programs can often add to application functionality and developer efficiency, and there are certainly many cases where the use of a procedural language such as the MySQL stored program language can do things that a non procedural language like SQL cannot. There are also a number of reasons why a MySQL stored program approach may offer performance improvements over a traditional SQL approach. It provides a procedural approach (SQL is a declarative, non-procedural language). It reduces client server traffic. It allows us to divide and conquer complex statements. [15]

## VII. CONCLUSION

In the world of databases, some developers try to build database independent applications, especially using ORMs (object relational mappers). On the surface, this seems like a great option, build your application to use only standard components and features, and then you can easily move to a different platform when requirements dictate. Unfortunately, things are not quite that simple.

Migration in the context of enterprise and web-based applications means moving from one platform to another. Database Migrations are particularly complicated as you have all the challenges of changing your software platform, where some old features are missing, or behave differently and some new features are available and you'd like to take advantage of those. The paper presented the overview on data migration as well as studied the various types of data migration available. Besides it also focused on the cross-platform data migration. No study is complete without mentioning the challenges faced while adapting the new techniques. Therefore, I have tried to list all the common challenges faced so far by the industrialists.

## REFERNCES

1. https://www.netapp.com/us/info/what-is-data-migration.aspx
2. http://www.dataintegration.info/data-migration
3. https://www.edq.com/uk/blog/8-hurdles-of-a-data-migration/
4. http://www.datacleansing.net.au/data-merging-services/
5. http://www.infotechnet.org/ntca/DataMigration.htm
6. https://searchstorage.techtarget.com/definition/data-migration
7. https://sapbasisdurgaprasad.blogspot.in/2012/12/what-is-difference-between-homogeneous.html
8. https://docs.oracle.com/cd/E25178_01/backup.1111/e10642/rcmxplat.htm
9. https://docs.oracle.com/database/121/BRADV/rcmxplat.htm#BRADV89977
10. https://en.wikipedia.org/wiki/Schema_migration
11. http://www.visolve.com/uploads/resources/Cross%20Platform%20Transportable%20Tablespaces%20Migration%20in%20Oracle%2011g_ViSolve%20Inc.pdf
12. https://docs.oracle.com/en/cloud/paas/exadata-cloud/csexa/mig-transportable-tablespace-inc-backup.html
13. http://www.datacenterjournal.com/steps-successful-data-migration/
14. https://www.xaprb.com/blog/2009/03/13/50-things-to-know-before-migrating-oracle-to-mysql
15. https://www.percona.com/files/presentations/percona-live/PLMCE2012/PLMCE2012-oracle_mysql_sc_2012.pdf
16. http://doc.ispirer.com/sqlways-oracle-to-mysql-whitepaper.pdf

## AUTHOR'S BIBLIOGRAPHY

### 1. Stuti Nareshkumar Garg:

Stuti Nareshkumar Garg, a fresh graduate in Computer Engineering is presently working with Allianz Infonet,Vadodara, as a Web Developer. She has done her summer Internship in Super Computing division of BARC(Bhabha Atomic Research Centre).Her area of interest include Big Data, Machine Learning, Artificial Intelligence

### 2. Pratibha Bahulkar:

Pratibha Bahulkar, is presently working with Bhabha Atomic Research Centre, Computer Division, Mumbai. Her area of interest include data mining and data analytics.

### 3. Ketan Rathod:

Prof. Ketan B Rathod, presently working as an Assistant Professor in Babaria Institute Of Technology, Vadodara, Gujarat and having teaching experience of more than 8 years. His area of interest includes Soft Computing, Artificial Intelligence and Data mining. He is also awarded with the Longest Continious SBC by Computer Society of India in annual meet at SCI Kolkata.

### 4. Amita Garg

Amita Garg, MCA, working as an Assitant Professor at Parul University, Vadodara, Gujarat. Her research interest includes Big Data, Cloud Computing, Artificial Intelligence, Data analytics and use of IT in business application. She has published papers in International journal in various national and internaltional conferences. She is also an active member of various professional bodies like CSI, IETE and IE.
His area of interest include Soft Computing, Artificial Intelligence