iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** XII **Month of publication:** December 2022

DOI: https://doi.org/10.22214/ijraset.2022.47846

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XII Dec 2022- Available at www.ijraset.com

Data Migration Techniques across DBMS by Using Metadata

Om Deshmukh
Master of Computer Applications – Cloud Technology (MCA-CT), Ajeenkya D.Y. Patil University, Charholi Budruk, Pune

Abstract: ***The data migration process is usually needed when there is a change in system, format, or storage type. Currently, there are several techniques and tools for migrating data, for example, CSV files, ODBC, SQL Dump, and so on. Inappropriately not all of these techniques can be implemented for migrating data between two different DBMS. This research describes a data migration technique that can be used to migrate data across DBMS. The data migration technique described makes use of existing metadata in each DBMS. The data migration process described here goes through three stages, namely capture, convert and construct. A prototype was built to test this data migration technique. By using the HR schema, cross-DBMS data migration trials were carried out between Oracle and MySQL. Using this technique, full-schema data migration takes an average of 20.43 seconds from Oracle DBMS to MySQL and 12.96 seconds for the reverse scenario. As for partial data migration, it takes an average of 5.95 seconds from Oracle DBMS to MySQL and 2.19 seconds for the reverse scenario***
Keywords: data migration; DBMS; metadata; data type.

I. INTRODUCTION

Currently, there are so many choices of Database Management Systems (DBMS) that can be used. Each DBMS has its own advantages and market. In its use, it is very possible for the data migration process to occur between one DBMS and another DBMS. The data migration process is usually needed when there is a change in system, format, or storage type. Currently, there are several techniques and tools for migrating data, for example, CSV files, ODBC, SQL Dump, and so on. Unfortunately, not all of these techniques can be implemented for migrating data between two different DBMS.

The CSV file format is considered very universal because it can be read by all DBMS, but unfortunately, CSV does not provide information about data types and is considered impractical when faced with a scenario of moving data from several tables in one schema. Therefore, even though it is universal, SCV is only suitable for moving data on a table scale. Open Database Connectivity (ODBC) offers the convenience of moving data on a larger scale because ODBC is able to move data from several tables that are in one schema at once. Unfortunately, ODBC drivers are not always compatible and not always available. Likewise, SQL Dump can only be used for data migration between similar DBMS because of its strong dependency on data types. Therefore, we need a data migration technique that is not only universal, and not affected by differences in data types, but can also be used on a large scale, has good compatibility, does not depend on driver availability, and gives full control to the user in the migration process data.

II. DATA MIGRATION WITH METADATA

There are several definitions of data migration in various references. In [1] it is stated that data migration is a technique or process of moving data, for data that changes for certain reasons such as a system change, where the new system to be implemented requires data from the old system. Meanwhile, in [2], it is stated that data migration is the process of moving data that has changed the type of storage, data format, or data processing system. Usually done with the help of a computer to minimize manual processes. Data Migration is done because the organization is upgrading or changing the system. Therefore data migration can be defined as a process or technique of transferring data that is carried out with the help of a computer where the old system changes both from the type of storage, data format, and data processing system in such a way that data from the old system can still be used on the new system. One of the biggest challenges in data migration is if data migration is carried out due to changes in the data processing system or database management system (DBMS). To meet this challenge, the most suitable data migration technique is to utilize metadata. Metadata is data about data. Metadata exists in almost any DBMS. Usually metadata stores information about the data stored in the database, such as the structure, type, and location of data storage. Techniques for migrating data are quite diverse, for example using SQL Dump [3], Comma Separated Values (CSV) [4], Open Database Connectivity (ODBC) [5], eXtended markup language (XML) [6], database reengineering [7] and so on. Each technique and method has its own strengths and weaknesses. Table 1 shows a comparison of capabilities between several data migration techniques.

©IJRASET: All Rights are Reserved

Capability	Data Migration Techniques			
	SQL Dump	*CSV*	*ODBC*	*Metadata*
Cross DBMS		√	√	√
View Schema			√	√
View Table			√	√
Select Schema	√			√
Select Table	√	√	√	√
Select Column				√
Data Type Conversion		√	√	√
Log Migration				√

Table 1. Data Migration Techniques

1. Capture 2. *Convert* 3. *Construct*

The following is an illustration and explanation of the three stages of migrating data across DBMS using metadata

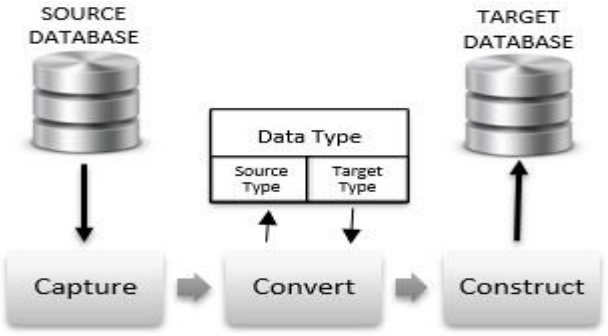


Figure 1. Stages of Data Migration with Metadata

A. Capture

At the capture stage, what is done is to establish a connection to the data source database and then perform a query to read the existing metadata in the data source database. The result of the capture process is information about the structure of the data to be moved, including the user who owns the data, lists of tables, columns, data types for each column, constraints, and indexes to get this information, more than one metadata is used. Each DBMS also has different metadata. The list of metadata used in both the Oracle and MySQL DBMS is shown in Table 2 and Table 3 below.

Metadata	Function
DBA_USERS, USER_ROLE_PRIVS	Identify the list of users and schemas owned and the privileges of each user
USER _TABLES	Identify a list of tables
USER_TAB_COLUMNS	Identifies a list of columns and data types, their length and precision
USER_CONS_COLUMNS, USER_*CONSTRAINTS*	Identify structure and breed constraint
USER_IND_COLUMNS	Identify the list and the type and location of the index which is owned

Table 2. List of Metadata Used in Oracle DBMS

Metadata	Function
SCHEMATA	Identify the list of users and schema owned
TABLES	Identifies the list of tables that belong
COLUMNS	Identifies a list of columns and data types, their length and precision
COLUMN_USAGE, TABLE_*CONSTRAINT*	Identify the structure and types of constraints

Table 3. List of Metadata Used in the MySQL DBMS

Metadata that overlap with each other is also used to get more information, for example, to get information about constraints in Oracle, two metadata that overlap each other are needed as shown in Figure 2.

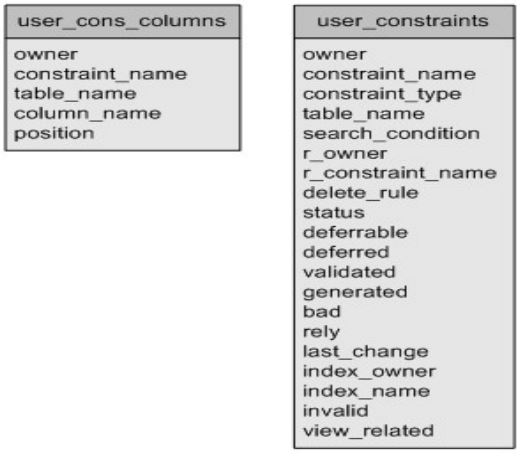


Figure 2. Metadata for Identifying Constraints in Oracle

Likewise with metadata in MySQL, to get information about constraints, two metadata that overlap each other are used as shown in Figure 3.

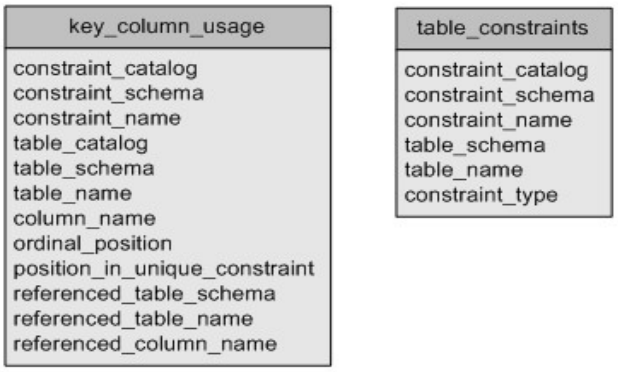


Figure 3. Metadata for Identifying Constraints in MySQL

The data rows are copied to a temporary file before changing the data type in the next stage, namely the convert stage.

B. Convert

At the convert stage, the metadata reading results from the capture stage are analyzed automatically. After that, a script containing a series of Data Definition Language (DDL) commands is also automatically created to create users, build schemas, create tables, and arrange relationships between tables complete with constraints and indexes in the destination database. This script is created based on the results of reading the metadata in the source database.

Data type conversions are made to accommodate different data types from one DBMS to another. The data type conversion mapping is carried out based on the similarity of data type characteristics as described in [8] and can be seen in Table 4 below.

Data Type MySQL	Data Type Oracle
BIGINT	NUMBER (19, 0)
INT	NUMBER (10, 0)
MEDIUMINT	NUMBER (7, 0)
SMALLINT	NUMBER (5, 0)
TINYINT	NUMBER (3, 0)
YEAR	NUMBER
DECIMAL	FLOAT (24)
DOUBLE	FLOAT (24)
REAL	FLOAT (24)
FLOAT	FLOAT
DATE	DATE
DATETIME	DATE
TIME	DATE
TIMESTAMP	DATE
TEXT	VARCHAR2
TINYTEXT	VARCHAR2

Table 4. Data Type Conversion Mapping.

In addition, a script containing a series of Data Manipulation Language (DML) commands is also automatically generated. This DML command is in charge of inserting data into tables in the target database according to the rows of data stored in the temporary file as a result of reading the data in the capture process that was carried out previously.

DDL and DML scripts that have been made are not immediately executed at the convert stage. However, both DDL and DML scripts are structured in such a way that the order of execution in the script can guarantee success in the next stage, namely the construct stage.

C. Construct

At this stage, a connection to the target database is built, then the DDL script generated from the convert stage is executed on the target database. The DDL script is executed in stages to ensure that users, schemas, and tables are successfully built on the target database complete with the relationships between tables, constraints, and their respective indexes.

After ensuring that the DDL script is executed perfectly, then the DML script obtained from the convert stage is also executed to copy data from the temporary file to the target database. Thus both the database structure and the data are successfully copied from the source DBMS to the target DBMS.

III.DATA MIGRATION PERFORMANCE TEST WITH METADATA

A prototype was developed to migrate data across DBMS using metadata according to the three stages previously described, namely capture, convert and construct. This prototype was built using the VB.NET programming language and has the following capabilities.

1) Set the connection configuration for both source and destination DBMS
2) Displays table structure information such as column data, constraints, and index.
3) Choose full or partial data migration level (table-specific migration).
4) Perform data migration and data type conversion from the original DBMS to the target DBMS.
5) Presenting information on the results of the data migration process in a log file

The built prototype was then tested with a series of cross-DBMS data migration experiments between Oracle and MySQL DBMS. This test aims to measure the effectiveness and efficiency of data migration techniques with metadata in terms of the required execution time.

A. Test Environment

The prototype was tested in the following test environment:

1) *Hardware:* Processor Corei3 2.53GHz, 2GB RAM and HDD Space 320GB
2) *Software :* Windows 8.1 Pro-32-bit operating system, Oracle 11g XE, and MySQL 5.0.8

B. Test Scenario

The prototype was tested with 2 scenarios, namely full schema data migration and partial data migration.

1) *Full Schema Data Migration:* The prototype is tested to move data from a schema, complete with the relationship between tables, constraints, and indexes respectively. In this test, the "HR" schema is used, which is a sample schema from the Oracle DBMS.
2) *Partial Data Migration:* The prototype is tested to move data from several schemas, namely from one of the tables, but complete with constraints and indexes attached to that table. In this test, the "Employees" table is used which is one of the tables from the "HR" schema. Details of the data migration test scenario can be seen in Table 5 below.

Parameter	**Data Migration Types**	
	Full Schema	***Partial***
Number of Table	7 (*Schema* HR)	1 (Employees)
Number of rows	215	107
Number of Column	35	11
Number of Constraint	19	4
Number of Index	21	7

Table 5. Data Migration Test Scenario Details

In both full schema data migration and partial data migration, the prototype was tested to migrate data from Oracle to MySQL and vice versa, from MySQL to Oracle. Each trial was repeated 10 times and the execution time required for each trial was recorded and then averaged.

Scenario	**Number of Trials**	
	From Oracle to MySQL	***from MySQL to Oracle***
Full *Schema*	10	10
Partial	10	10
TOTAL	**40**	

Table 6. Details of the Data Migration Testing Experiment

C. Test Result

The tested prototype successfully migrated data for both full schema and partial data migration scenarios. Cross-DBMS data migration was successfully performed both from Oracle to MySQL and vice versa. The following is an example of a log file generated during the testing phase

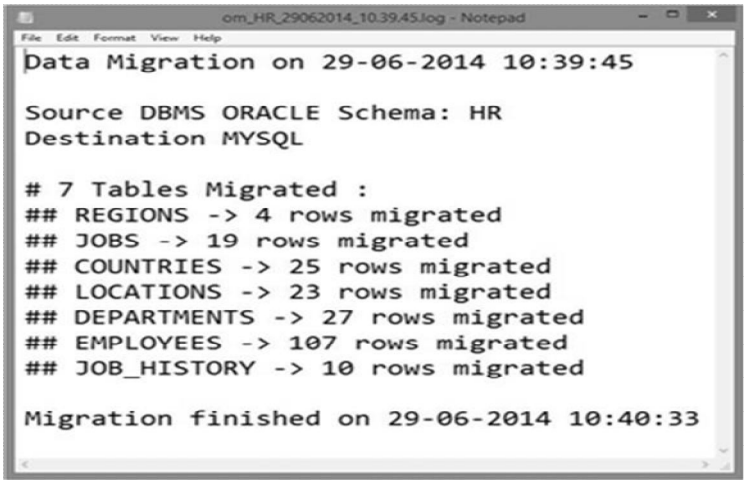


Figure 4. Example of a Log file

The detailed log file displays information about:

1) Source DBMS
2) Target DBMS
3) The name of the schema being migrated
4) Number and list of tables migrated,
5) The number of data rows from each table that were successfully migrated.

In addition, each log file has a record of the time the data migration started, namely at the beginning of the capture stage until the time the data migration process was completed, namely at the end of the construct stage. The time difference between the two is recorded as the data migration execution time. The execution time required for each scenario is recorded and then averaged. The following is Table 7 which displays the test results in the full schema data migration scenario. The results of testing the partial data migration scenario can be seen in Table 8 below.

Test to	Execution Time (seconds)	
	from Oracle to MySQL	*from MySQL to Oracle*
1	48,78	15,60
2	24,90	12,25
3	18,65	11,71
4	12,00	11,57
5	10,78	11,43
6	14,16	11,20
7	33,20	11,24
8	15,23	11,71
9	12,26	22,16
10	14,38	10,73
Rate-rate	**20,434**	**12,96**

Table 7. Full Schema Data Migration Execution Time.

Test to	Execution Time (seconds)	
	from Oracle to MySQL	*from MySQL to Oracle*
1	7,43	3,61
2	4,93	1,47
3	3,81	1,49
4	4,54	2,41
5	4,12	1,94
6	7,47	3,13
7	10,28	1,62
8	5,48	1,57
9	5,21	2,30
10	6,27	2,39
Rate-rate	**5,958**	**2,193**

Table 8. Partial Data Migration Execution Time

The test results show that for a full-schema data migration scenario it takes an average of 20.43 seconds from Oracle DBMS to MySQL and 12.96 seconds for the reverse scenario. As for the partial data migration scenario, it takes an average of 5.95 seconds from Oracle DBMS to MySQL and 2.19 seconds for the reverse scenario.

IV. CONCLUSION

Data migration techniques with metadata can be used to transfer data between two different DBMS without losing the constraints on the data. This technique can also be used for full or partial data migration.

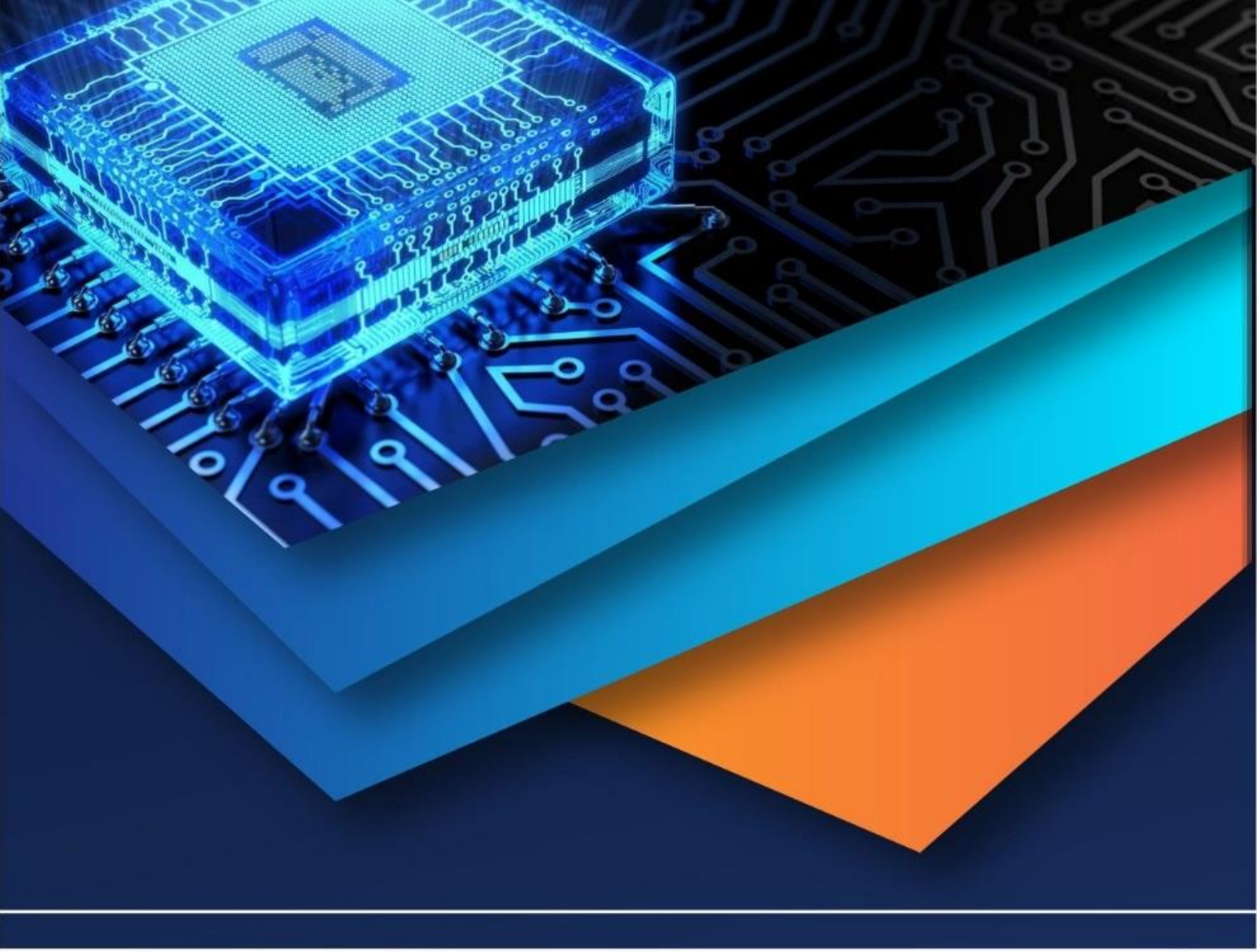
The results of reading the log file during the test show that for the full schema data migration scenario it takes an average of 20.43 seconds from Oracle DBMS to MySQL and 12.96 seconds for the reverse scenario. As for the partial data migration scenario, it takes an average of 5.95 seconds from Oracle DBMS to MySQL and 2.19 seconds for the reverse scenario. This shows that in terms of the time needed, the performance of data migration techniques with metadata still provides a lot of room for improvement.

V. SUGGESTION

Because the proposed data migration technique only utilizes some metadata, for further development, it is necessary to study and map other metadata that can be used for data migration. In addition, a better data type conversion mechanism also needs to be developed, especially for data types that have the ability to accommodate large data such as the Binary Large Object (BLOB) and Character Large Object (CLOB) data types.

REFERENCES

[1] J. Morris, Practical Data Migration, Swindon: The British Computer Society, 2009.
[2] Trish Rose-Sandler, "Introduction to Data Migration," in Visual Resources Association Conference, Kansas, 2007.
[3] K. Rich, Oracle Database Utilities, 10g Release 2, Oracle, 2005.
[4] Y. Shafranovich, October 2005. Common Format and MIME Type for Comma-Separated-Values [Online]. Available: http://tools.ietf.org/html/rfc4180.
[5] Kingsley Idehen, “Open Database Connectivity,” OpenlinkSoftware, Whitepaper 1993. [Online].
[6] R. Setiawan and A. Nugroho, "Data Exchange System between Databases with XML" in the National Seminar on Information Technology Applications (SNATI), Yogyakarta, 2005
[7] D.H. Putra and H.A. Wibawa, "Implementation of Interpretive Transformer Approach in Data Migration as a Database Network Reengineering", Journal of Informatics Society Vol. 5 No. 9 p 53-61, 2014.
[8] C. Murray, Oracle SQL Developer Supplementary Information for MySQL Migrations, California: Oracle Corporation, 2008.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)