LB/DON/106/2016

IT 01/139

Migrate and Transfer Schema and Data across Multiple Databases

I. Milinda Wijewardana
139179 E

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Degree of Master of Science in Information Technology.

April 2016

004 "16"
001 (043)

TH 3174
+
1 DVD - ROM
(TH 3160 - TH3180)

Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

I. M. Wijewardana

Name of Student

Signature of Student

Date: 26-04-2016

Supervised by

UOM Verified Signature

Name of Supervisor

Signature of Supervisor

Date:

Acknowledgements

I would like to thank my thesis supervisor Mr. S. C. Premaratne of the Faculty of Information Technology at University of Moratuwa.

Abstract

In instances when software is made to interact with multiple databases the upgrade of the software requires data migration. In addition, there are also software's that move schema and data between databases through migration. Database migrations normally take very long time to execute and it is an error prone process. Many commercial and open sourced database migration products are available, but the effectiveness of a tool always depends on features that it supports, however these features do not fully support for database migration projects' requirements. Based on our studies, we design and implement the solution which automates database migration for migrating and transferring schema and data through different types of vendor databases such as MySQL, Oracle, PostgreSQL and MS SQL Server. The solution is developed on top of Java technologies such as NetBeans Platform APIs and Java Database Connectivity API. Creating a Java Swing application on top of NetBeans Platform can largely reduce development time. The NetBeans Platform is an open source framework for Java Swing application. The Java Database Connectivity API is the main standard for database independent connectivity among a Java application and different types of vendor databases. The proposed solution supports project based mechanism to manage the migrations and it caters simple user-friendly wizard based GUI for migration. The solution also provides a set of automated database migration features, which allows migration of complicated database schema and data from one vendor database to another vendor database. The solution automates up to ninety percent of the manual work of database migration. It gives full migration solution for different types of vendor databases such as MySQL, Oracle, PostgreSQL and MS SQL Server. This automated database migration solution helps to save migration effort and costs involved in database migration projects.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Prolegomena

Assuring quality database administration is one of the greatest challenges faced by organizations [1]. Legacy software applications are slowly but steadily fading away. This has given rise to many application developers switching to current technologies. This has given rise to the need for database migration. This involves converting the schema and data from one system to another database system. What could be regarded as a well-designed database migration tool? Such a tool should successfully load data from the source applications, retain consistency & reliability in data and follow quality principles while compiling the data [2].

Most of database administrators perform database migration which is a very common job for them. In certain instances the success of a business may depend on the database migration, which includes the transferring of data and structure in from a database into a target database. These instances could occur due to company mergers or due to the need of moving data between contractors. As technology changes and businesses growth to the different level, a database migration plan is vital to safeguard all systems, services, and applications have access to important information from different solutions. No matter what the reason for having to drive through database migration, businesses need a simple and smooth database migration plan to simplify the process.

Although very often there is no proper budget associated to database migration and as such over expenditure and difficulties of a data conversion have been highly underestimated. Improper migration can cause mismatches in information between the original and new platform [3].

It has been identified that applications that are developed for one database cannot be easily moved to another database. This issue is prominent in database languages such as Oracle PL/SQL and MS SQL Server Transact-SQL. Thereby using conventional methods for database migration is too demanding a cost in terms of human resource, man hours and manual effort that may show unpredictable migration results.

After researching many licensed and open source database migration tools, it was identified that there is no current product could successfully meet all requirements unless significant customization is undertaken. Therefore, it was decided to build a new database migration tool.

1.2 Background and Motivation

In the software industry, companies have to work with different customer expectations as well with a number of different vendor databases. Frequently a given vendor database is required to migrate or convert to different type of vendor databases. However said migration or conversion mechanisms are expensive, and furthermore it was experienced that free open source software does not often provide a comprehensive and suitable solution. Organizational changes due to business growth, customers, or new market demands can all result in the implementation of new systems. This creates the need to move valuable data from one system to another. Database migration tools are always complex and require expert solutions to guarantee success.

There are many data migration solutions currently out on the market. These solutions, however, are typically flawed [4]. Many database migration tools are available for free or for a low price, and although they complete the task of moving data, the process is very unclear. Moreover, these tools often limit the amount of data that can be imported, exported, and deleted. Many solutions also require a data management software download or in some cases, hardware installation. These data management solutions can chaos and complicate data migration initiatives, further confusing processes. Businesses already employ numerous applications, services, and systems, requiring additional software and hardware for database migration only makes matters more problematic. Some solutions are quite costly, often requiring consultants or experts to come in and manage the migration. This method is not only costly, it can also be time consuming. Often, an experienced third party vendor is called in to handle the database migration, but this process can take time to manage and carry out.

1.3 Problem Definition

Migration and transformation of schema and data across multiple databases have been research challenge.

1.4 Aim and Objective

- The proposed application offers productive database migration tool for schema and data migration across multiple databases that also proves to be reliable and protect data integrity through its flexible, user-friendly and extensible migration process.
- It offers comprehensive migration solution across a wide range of databases such as Oracle, SQL Server, MySQL and PostgreSQL.
- This automated database migration solutions helps to save migration effort and costs involved in database migration projects.
- The proposed application provides a lightweight integration platform that delivers powerful database migration capabilities.
- The solution uncomplicated database migration, making it easy for customers to streamline and automate business processes.
- This saves time, boosts productivity, and allows businesses to spend more time concentrating on core business needs.
- When the solution simplifies the integration process, businesses can focus time and resources on core business needs.
- The main objectives of this solution is to offer a flexible, open and extensible migration process
- That is fast and protects data integrity with no data loss.
- During migration the solution reorganizes and transforms schema and data consistently with minimum.
- The solution also provides a uniform approach for migration across multiple databases.
- This can also be used as a unique mechanism for verification of the migrated database.
- This mechanism is known as database verification.
- It is user-friendly while user installation is easy.

1.5 Hypothesis

The hypothesis is defined as developing an automated database migration tool which migrates and transforms schema and data across multiple databases such as Oracle, SQL Server, MySQL and PostgreSQL.

1.6 Structure of the Thesis

The rest of the thesis is structured as follow. The chapter 2 is on literature review of database migration tools. The chapter 3 presents technology adopted towards an automated database migration tool for migrating and transforming schema and data across multiple databases. The chapter 4 provides the overall picture of our novel approach to an automated solution for migrating and transforming schema and data across multiple databases. The chapter 5 discusses the design of the solution. The chapter 6 is about the implementation of the solution. The chapter 7 reports on the evaluation of the proposed solution. The chapter 8 concludes the thesis with a note on further work.

1.7 Summary

This chapter gave an overview of the automated solution for database migration. A brief introduction was given about database migration and this chapter provides a discussion of some of the background and motivation for this study. We defined the problem definition and the hypothesis for this thesis. This chapter mentioned about aim and objective of this research work. Next chapter shows the current developments of database migration area.

Chapter 2

Current Development in Database Migration

2.1 Introduction

MySQL, MS SQL Server, Oracle and PostgreSQL are very similar to each other but differ in metadata organization, internal data manipulation capabilities and in their supported data types. Database migration and synchronization tools helps to copy a database easily and quickly by just a few clicks guided by step by step instructions. Thereby cancelling the need for manual data transfer (which is highly time inefficient and tedious and thereby prone to error) and the requirement of an experienced user. Basically, such tools can be used by even the most novice users.

Data migration and synchronization tools keep the data integrity, database structures, the relations between tables and map data types. This is done by doing the following;

- Forming a new target database
- Creating tables
- Creating indexes
- Migrates data from existing database
- Examines elements of source database before migration
- Flags foreseen problems in data type's incompatibility between the existing database and target one

Therefore choosing the right tool for data migration is very important. But unfortunately many organizations end up using an in-house tools that caters to a totally different type of users instead of purchasing a new one which would be better designed for data migration.

2.2.1 FlySpeed Database Migration Tools

FlySpeed DB Migrate [5] is an efficient migration tool that assists enables the easy modification of the database structure and user control over the migration process. This tool addresses the two main issues faced when migrating databases by firstly completely transferring the database table structure and data based on all the details of the destination server. Secondly transferring data into existing tables while maintaining data actuality (single-side synchronization). FlySpeed DB Migrate is therefore fast in migration while minimizing data loss. The FlySpeed DB Migrate tools allow to import structure and data of tables from

SQL Server, Oracle, DB2, Informix, Access, PostgreSQL, MySQL, Interbase, Firebird, dBase files, paradox files, etc. There are two tools to achieve database migration. One is for MySQL and the other one is for SQL Server but this is a main limitation of this work because one tool should support for the two databases to execute database migration. The other main limitation is this is a commercial product. FlySpeed DB Migrate costs US$125 per license (estimation) without maintenance and US$162 per license with maintenance. The final limitation is FlySpeed DB Migrate only supports MySQL and SQL Server databases as target databases.

2.2.2 ESF Database Migration Toolkit

ESF Database Migration Toolkit [6] is a data migration tool developed by EasyFrom. This tools enables the user to transfer data without having to write scripts. This is made possible by offering a 3 step wizard. ESF is reviewed to be the fasted migration tool available. Due to its wizard approach it reduces the effort of the user, cost and risk. It migrates from and to any of the following database formats; Oracle, MySQL, SQL Server, PostgreSQL, IBM DB2, IBM Informix, InterSystems Cache, Teradata, Visual Foxpro, SQLite, FireBird, InterBase, Microsoft Access, Microsoft Excel, Paradox, Lotus, dBase, CSV/Text and can transfer any ODBC DSN data source to them as well. ESF migrates all table structure, data, schemas (Oracle, SQL Server, and PostgreSQL), LOB (Large Text/Binary Objects), primary key/foreign key, indexes, auto-increment (serial) and default values. It will also change methods with regards to table/field name, data type, length, default value and etc., or filter data. In addition is supports UNICODE and enables to migrate between different character-sets automation. Although ESF is very expensive and thereby can only be purchased by companies which have strong portfolios.

2.2.3 SwisSQL - Data Migration Tool

SwisSQL [7] is a complete data migration tool that enables to transfer, convert, import, export, port data from or to Oracle, DB2, SQL Server, Sybase, MySQL, PostgreSQL and Access databases. Databases such as Oracle, IBM DB2, MS-SQL Server, Sybase, MySQL, PostgreSQL and MS Access are supported by this tool in migration and transfer of database schemas. In addition it has a facility to migrate data from Excel and CSV (Comma Separated Value) files into databases. It is highly reliable and user friendly. The tool is also very fast ensuring data integrity with no loss of data. The uniqueness of SwisSQL is its "Database Migration Verifier" which helps to verify the migrated data. The tool can restructure/transform schema and data with ease during migration. This tool offers a uniformed approach in migrating data databases on different platforms such as Windows or UNIX or Linux and different databases. SwisSQL supports Oracle, IBM DB2, SQL Server, Sybase, MySQL, MaxDB, MySQL, PostgreSQL and MS Access databases. Its functionalities include the migration of tables, indexes, constraints inclusive of data. It also can migrate data SQL SELECT queries. SwisSQL also can develop scripts in a command line mode which can also be used for periodic scheduled migration. Stored procedures, functions, triggers are also support for migrating. Including if the source and target RDBMS are the same. An added feature of criteria based data migration is also possible. This when migrating from Oracle, SQL Server, MySQL, PostgreSQL and Sybase ASE to any database. This can be done by setting a filter criteria for a table that will enable migrating only when the rows match. In such situations its dependent tables are also migrated accordingly. Although the company that had developed and was marketing this tool had announced closure and thereby has discontinued the tool. Existing customers of SwisSQL are supported only until their prevailing contract expires.

2.2.4 MySQL Workbench - Database Migration

Oracle Corporation has designed MySQL Workbench Migration Wizard [8]. By implementing an easy to use visual point and click way of working throughout the system this tool has managed to save developer time by simplifying an otherwise complex process. This tool enables configurations, copy, edit, execute and scheduling. The tool support migration from Microsoft SQL Server, Microsoft Access, PostgreSQL, Sybase ASE, Sybase SQL Anywhere, SQLite, etc. Some added features include source and target selection for defining specific data sources and enabling to analyze source data while migration is taking place. MySQL Workbench allows for object migration which involves the selection of objects to migrate and then assign a source target mapping, edit migration scripts and create the target schemas. It is possible to convert an existing database to MySQL in minutes using the MySQL Workbench Migration Wizard. Instead of using the manual methods. MySQL Workbench Database Migration is an open source tool. But is migrates only a selected vendor databases to a MySQL database. This tool has been developed and is owned by Oracle and therefore it does not transfer an Oracle database to MySQL database.

2.2.5 Microsoft SQL Server Migration Assistant (SSMA)

SSMA [9] is an open source tool developed and owned by Microsoft. This tool mainly simplifies that data migration process from Oracle to SQL Server and Azure SQL DB. This is done by automating all the aspects of migration assessment analysis, schema and SQL statement conversion, data migration as well as migration testing. This tool comprises of a user friendly screen to manage the migration process. In addition to an SSMA extension pack to be installed on target SQL Server. SSMA extension pack includes functionalities to emulate Oracle features not natively supported in SQL server, tester database to support SSMA Testing features, and an application to facilitate direct server to server data migration. However SSMA tools only support to migrate from Oracle, MySQL and Sybase databases to SQL Server databases.

2.2.6 Oracle SQL Developer

Oracle SQL Developer [10], is an open source tool that supports database migration. Users are able to migrate database objects and data. This can be done from IBM DB2, MySQL, Microsoft SQL Server, Microsoft Access, Sybase and Teradata to Oracle. A special feature in this tool is the support of both online and offline migration. Online migrations is connecting directly to the given database and migrating the database objects and data to Oracle. While offline migrations incorporates user files that is prepared from the given database, use files that user has prepared from the given database. Connecting to the database is not required as long as the user files is in order. This tool incorporates the concept of a migration project while users can use a wizard to take the user through a step by step method in completing the migration project. As a down-side, this product only supports to migrate from above mentioned databases to Oracle.

2.3 Problem definition – Research Question

The above study shows numerous limitations of the migrating schema and data across multiple databases. These limitations are summarized in Table 2.1.

Research	Limitation
FlySpeed DB Migrate	The following database migrations cannot be performed: From Oracle To Oracle From SQL Server To Oracle From MySQL To Oracle From PostgreSQL To Oracle From Oracle To PostgreSQL From SQL Server To PostgreSQL From MySQL To PostgreSQL From PostgreSQL To PostgreSQL

ESF Database Migration Toolkit	This product commercially very expensive. Small and medium sized companies cannot afford to have this product for database migration.
SwisSQL - Data Migration Tool	The company announced end of life notice for this tool. The company decided to discontinue any development and investments on this product to best align with their business goals and direction.
MySQL Workbench - Database Migration	The following database migrations cannot be performed: From Oracle To Oracle From SQL Server To Oracle From MySQL To Oracle From PostgreSQL To Oracle From Oracle To SQL Server From SQL Server To SQL Server From MySQL To SQL Server From PostgreSQL To SQL Server From Oracle To MySQL From Oracle To PostgreSQL From SQL Server To PostgreSQL From MySQL To PostgreSQL From PostgreSQL To PostgreSQL
Microsoft SQL Server Migration Assistant	The following database migrations cannot be performed: From Oracle To Oracle From SQL Server To Oracle From MySQL To Oracle From PostgreSQL To Oracle From PostgreSQL To SQL Server From Oracle To MySQL From SQL Server To MySQL From MySQL To MySQL From PostgreSQL To MySQL From Oracle To PostgreSQL From SQL Server To PostgreSQL From MySQL To PostgreSQL From PostgreSQL To PostgreSQL

Oracle SQL Developer	The following database migrations cannot be performed:
	From PostgreSQL To Oracle
	From Oracle To SQL Server
	From SQL Server To SQL Server
	From MySQL To SQL Server
	From PostgreSQL To SQL Server
	From Oracle To MySQL
	From SQL Server To MySQL
	From MySQL To MySQL
	From PostgreSQL To MySQL
	From Oracle To PostgreSQL
	From SQL Server To PostgreSQL
	From MySQL To PostgreSQL
	From PostgreSQL To PostgreSQL

Table 2.1: Summary of Limitation

Based on the above, the research problem is defined as migrating and transforming schema and data across multiple databases has been the solution. No adequate studies are done in migrating and transforming schema and data across Oracle, SQL Server, MySQL and PostgreSQL databases. I intent to solve the problem using Core Java APIs, Java Database Connectivity API and NetBeans Platform APIs with above mentioned databases.

2.4 Summary

This chapter illustrated current development of database migration tools. Table 2.2 shows that the summary of reviewed database migration tools and the proposed solution supports for multiple databases and checks whether these tools are commercial or open source products. The next chapter presents technology adopted to solve the research problem.

Product	Oracle	MS SQL Server	MySQL	PostgreSQL	Commercial?
FlySpeed Database Migration Tools	-	Support	Support	-	Yes
ESF Database Migration Toolkit	Support	Support	Support	Support	Yes
SwisSQL - Data Migration Tool	Support	Support	Support	Support	Yes
MySQL Workbench - Database Migration	-	-	Support	-	No
Microsoft SQL Server Migration Assistant	-	Support	-	-	No
Oracle SQL Developer	Support	-	-	-	No
Proposed Solution	Support	Support	Support	Support	No

Table 2.2: Summary of Database Migration Tools

Chapter 3

Technology Adopted for Database Migration

3.1 Introduction

Chapter 2 presented the current developments for giving an automation solution for migrating and transforming schema and data across multiple databases. This chapter presents the technologies to develop the database migration tool with leading databases such as Oracle, SQL Server, MySQL and PostgreSQL. The chapter highlighted the technologies that we are going to adopt to develop the solution.

3.2.1 NetBeans Platform Features

NetBeans Platform will significantly increase your development time of a desktop application [11]. It a generic framework for Swing applications. It saves time for developers by handling backend piping such as saving state, connecting actions to menu items, toolbar items and keyboard shortcuts, window management, etc. , that previously developers would have had to write on their own.

Thereby as the NetBeans Platform offers the above mentioned services the need for manual coding of these features is no longer required by the developer. Although it is true that the platform creates a lightweight application and it still can save much development time.

NetBeans Platform is a consistent and flexible application architecture that does not have to look anything like an IDE. It has been in the market for an ample period of time so as to prove its reliability. Which can be utilized for free. Its architecture encourages sustainable development practices. As the NetBeans Platform is based on a modular architecture using this platform, it has proved easy to create robust applications that stand the test of time.

An important question to explore is the reason as to why developers would use NetBeans Platform. This is attributed to the many features that the platform offers developers. The features are listed below;

3.2.2 Module System

Developers are able to successfully implement complex requirements using NetBeans by combining several small, simple, and easily tested modules due to its modular nature. Modules are able to work together dues to powerful versioning. Also developers are able to create more flexible applications that are also easy to maintain through strict control over the public APIs. Developers can combine third party modules or develop their own as they can use either standard NetBeans Platform modules or OSGi bundles.

3.2.3 Lifecycle Management

Lifecycle services are supplied for to Java desktop applications by NetBeans by way of its runtime container. The NetBeans runtime container composes NetBeans modules into a single Java desktop application which means that the developer is not required to a main method for an application. It also supports by way of persisting user settings such the default size and positions of application windows.

3.2.4 Pluggability, Service Infrastructure, and File System

Benefits are given from pluggable applications to end users of the application as they are enabled to install modules into their running applications using NetBeans. Through the runtime container these modules can be installed, uninstalled, activated, and deactivated at runtime. The infrastructure of the NetBeans Platform provides for registering and retrieving service implementations, enabling to minimize direct dependencies between individual modules and enabling a loosely coupled architecture (high cohesion and low coupling). It utilizes a virtual file system. This file system has a hierarchical registry for storing user settings, comparable to the Windows Registry on Microsoft Windows systems. Unified API that is also included in the platform provides stream-oriented access to flat and hierarchical structures, such as disk-based files on local or remote servers, memory-based files, and even XML documents.

NetBeans is useful for applications that need more than one window, such as Window System, Standardized UI Toolkit, and Advanced Data-Oriented Components. This is because NetBeans lets users maximize/minimize, dock/undock, and drag-and-drop windows, without providing any code at all. Thereby it simplifies tasks that would otherwise require coding.

Swing and JavaFX, which are the standard tools in Java can be used in the NetBeans Platform. In addition using this platform users are able to change the look and feel of the GUI and using

Look and Feel support in Swing and CSS. It also enables to GUI components across all operating systems. It is flexible to allow incorporation of third party tools such as Swing and JavaFX components.

Developers using the NetBeans Platform don't have to be worried with the same issues that are usually faced when using Swing. That of the JTree model that is used deferring from JList model, and thereby rewriting the model in order to switch between the two. This is because the NetBeans Nodes API uses a generic model for presenting data. It uses many advanced Swing components when used for showing nodes.

The NetBeans Platform while offering a windows system, also offers UI components such as a property sheet, a palette, and complex Swing components for presenting data, a Plugin Manager, and an Output window.

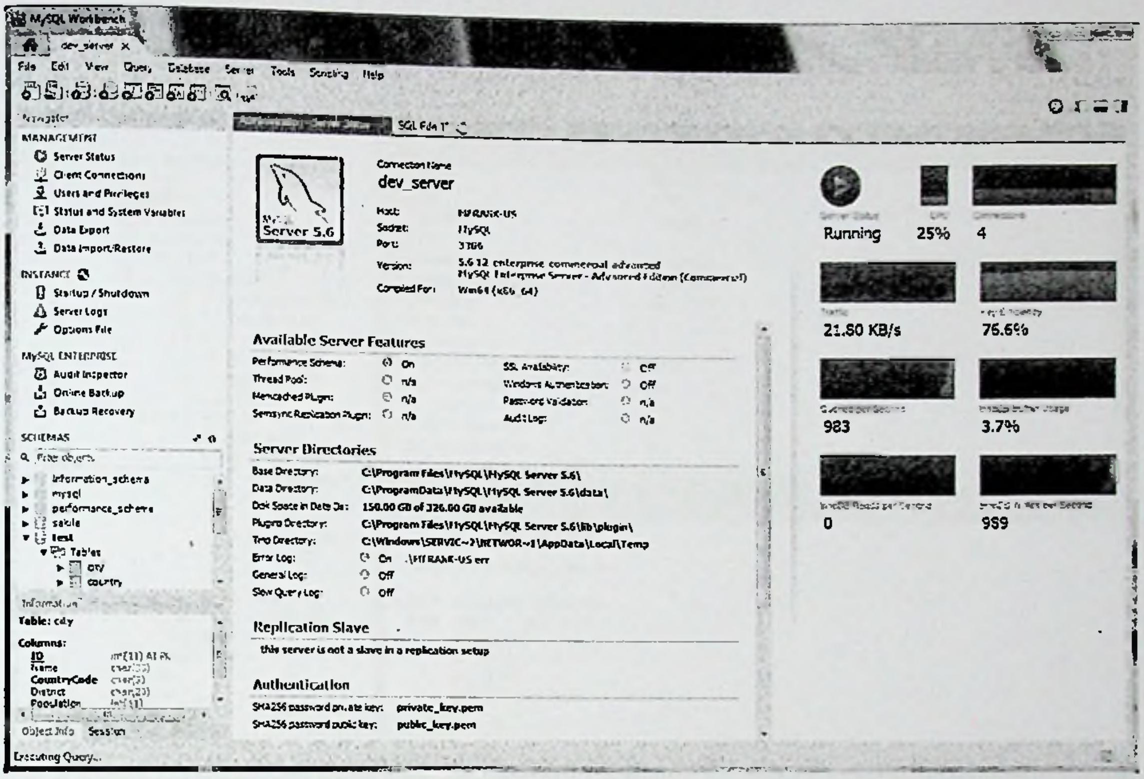
3.2.5 Miscellaneous Features, Documentation, and Tooling Support

The software development kit (SDK) for the NetBeans Platform is NetBeans IDE. It includes many templates and tools. Tools such as Matisse GUI Builder (which has won many awards) helps developers in the designing of application layouts easily. The platform has a very good collection of APIs. They are well tested and upgraded when necessary. The support features available to users are many and diverse. They include blogs, books, tutorials, and training materials translated into many languages and are being updated continuously.

3.3 MySQL Workbench

MySQL Workbench [12] is a tool that can be used by database architects, developers, and DBAs as a visual tool for data modeling, SQL development, server configuration, user administration, backup, etc. This can be operated in Windows, Linux and Mac OS X operating systems.

MySQL Workbench is a visual tool that assists the data architects to save much time and effort by allowing them to visualize requirements, communicate with stakeholders and resolve structural problems. It is a model-driven database design. Thereby it has the flexibility for efficiently creating well performing quality databases that easily collaborate with changing business requirements. In order to prevent mistakes from occurring during the building of new ER diagrams or generating physical MySQL databases, the model and schema validation utilities enforce best practice standards and MySQL-specific physical design standards when performing data modeling tasks.



Connection Name
dev_server

Host: MYRANK-US
Socket: MySQL
Port: 3306
Version: 5.6.12 enterprise commercial advanced
MySQL Enterprise Server - Advanced Edition (Commercial)
Compiled For: Win64 (x86_64)

Available Server Features

Performance Schema:	On	SSL Availability:	Off
Thread Pool:	n/a	Windows Authentication:	Off
Memcached Plugin:	n/a	Password Validation:	n/a
Semisync Replication Plugin:	n/a	Audit Log:	n/a

Server Directories

Base Directory:	C:\Program Files\MySQL\MySQL Server 5.6\
Data Directory:	C:\ProgramData\MySQL\MySQL Server 5.6\data\
Disk Space in Data Dir:	150.00 GB of 326.00 GB available
Plugins Directory:	C:\Program Files\MySQL\MySQL Server 5.6\lib\plugin\
Tmp Directory:	C:\Windows\SERVIC~2\NETWOR~1\AppData\Local\Temp
Error Log:	On .\MYRANK-US.err
General Log:	Off
Slow Query Log:	Off

Replication Slave

this server is not a slave in a replication setup

Authentication

SHA256 password private key: private_key.pem
SHA256 password public key: public_key.pem

Table: city

Columns:
ID int(11) AI PK
Name char(35)
CountryCode char(3)
District char(20)
Population int(11)

Executing Query...

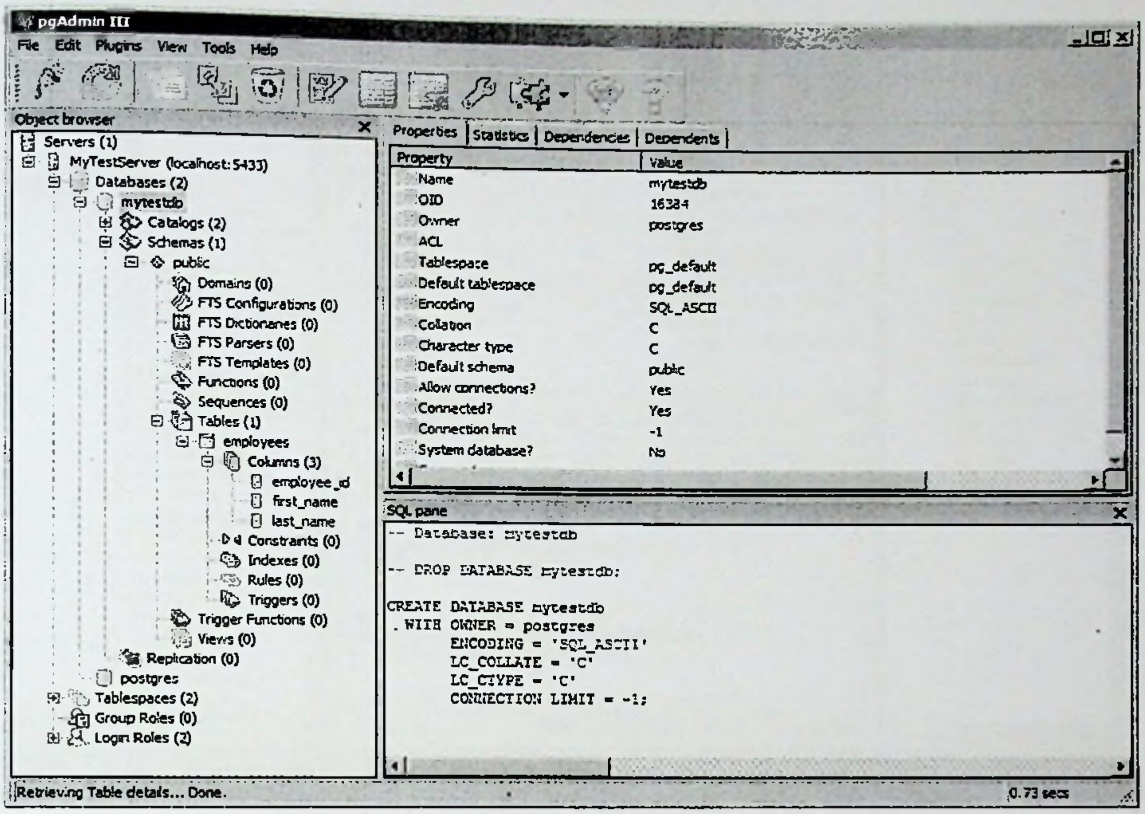
Figure 3.1: MySQL Workbench main screen window

3.4 pgAdmin

pgAdmin [13] can be considered the most popular tool. It is heavily laid with features relating to Open Source administration and development platform for PostgreSQL (which is the most advanced Open Source database in the world). The application can be run Linux, FreeBSD, Solaris, Mac OSX and Windows operating systems managing both commercial and derived versions of PostgreSQL 7.3 and above.

pgAdmin can be used for writing simple SQL queries to developing complex databases. Through its graphical interface users are supported with all PostgreSQL features which in turn makes administration easy. It also includes features such as; a syntax highlighting SQL editor, a server-side code editor, an SQL/batch/shell job scheduling agent, support for the Slony-I replication engine, etc. For security it may TCP/IP or Unix Domain Sockets (on *nix platforms), and SSL encryption. Also additional drivers are not required for communicating with DB servers.

A team of PostgreSQL experts have developed pgAdmin. It is available in several languages and is open source under PostgreSQL License.



```
Servers (1)
  MyTestServer (localhost:5433)
    Databases (2)
      mytestdb
        Catalogs (2)
        Schemas (1)
          public
            Domains (0)
            FTS Configurations (0)
            FTS Dictionaries (0)
            FTS Parsers (0)
            FTS Templates (0)
            Functions (0)
            Sequences (0)
            Tables (1)
              employees
                Columns (3)
                  employee_id
                  first_name
                  last_name
                Constraints (0)
                Indexes (0)
                Rules (0)
                Triggers (0)
            Trigger Functions (0)
            Views (0)
        Replication (0)
      postgres
    Tablespaces (2)
    Group Roles (0)
    Login Roles (2)
```

Property	Value
Name	mytestdb
OID	16384
Owner	postgres
ACL	
Tablespace	pg_default
Default tablespace	pg_default
Encoding	SQL_ASCII
Collation	C
Character type	C
Default schema	public
Allow connections?	Yes
Connected?	Yes
Connection limit	-1
System database?	No

```
-- Database: mytestdb

-- DROP DATABASE mytestdb;

CREATE DATABASE mytestdb
  WITH OWNER = postgres
       ENCODING = 'SQL_ASCII'
       LC_COLLATE = 'C'
       LC_CTYPE = 'C'
       CONNECTION LIMIT = -1;
```

Retrieving Table details... Done. 0.73 secs

Figure 3.2: PostgreSQL server with pgAdmin III

3.5 SQL Server Management Studio

SQL Server Management Studio (SSMS) [14] is a tool that can be used for accessing, configuring, managing, administering, and developing all components of SQL Server. It comprises of a vast range of graphic tools and script editors which give developers / administrators of all levels access to SQL Server. SSMS includes features such as Enterprise Manager, Query Analyzer, and Analysis Manager, (these have been included in previous releases of SQL Server as a single environment). Another plus point is that this tool works with all components of SQL Server i.e. Reporting Services and Integration Services. Therefore it offers easy to use graphical tools that combine to offer a comprehensive and comfortable experience for developers.

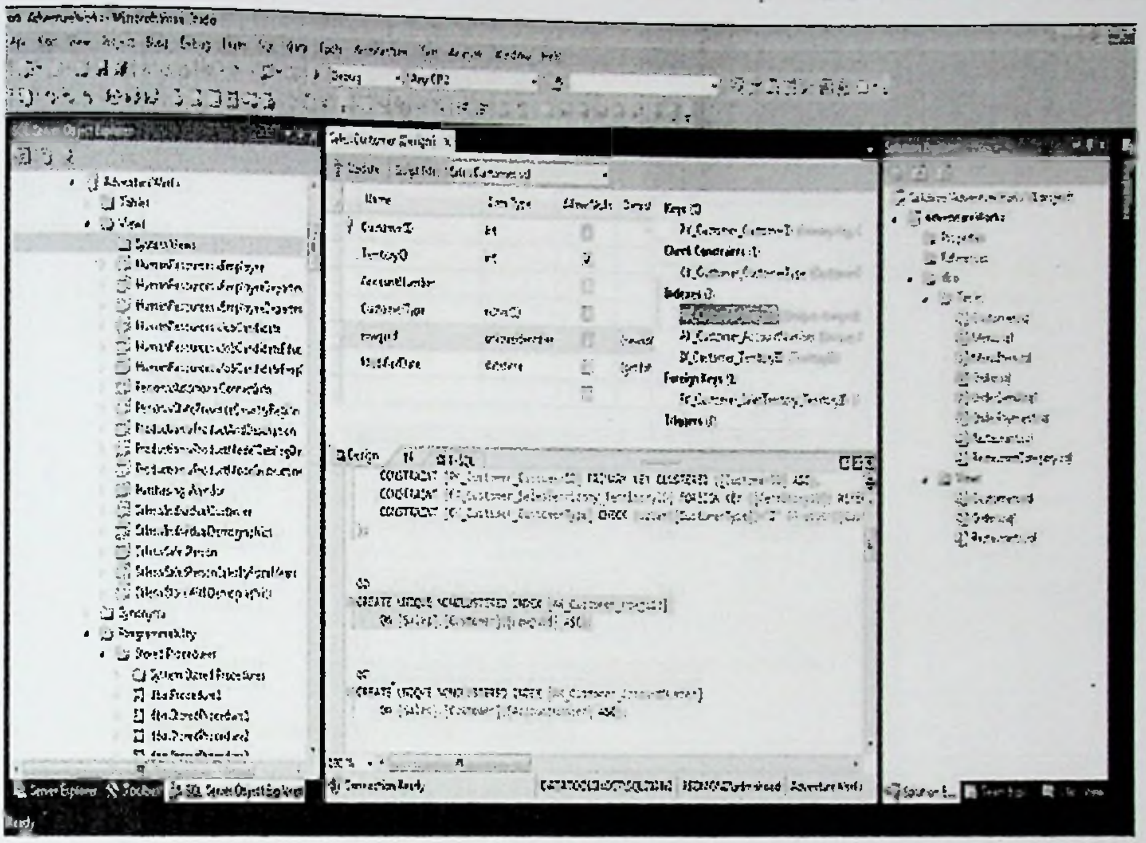
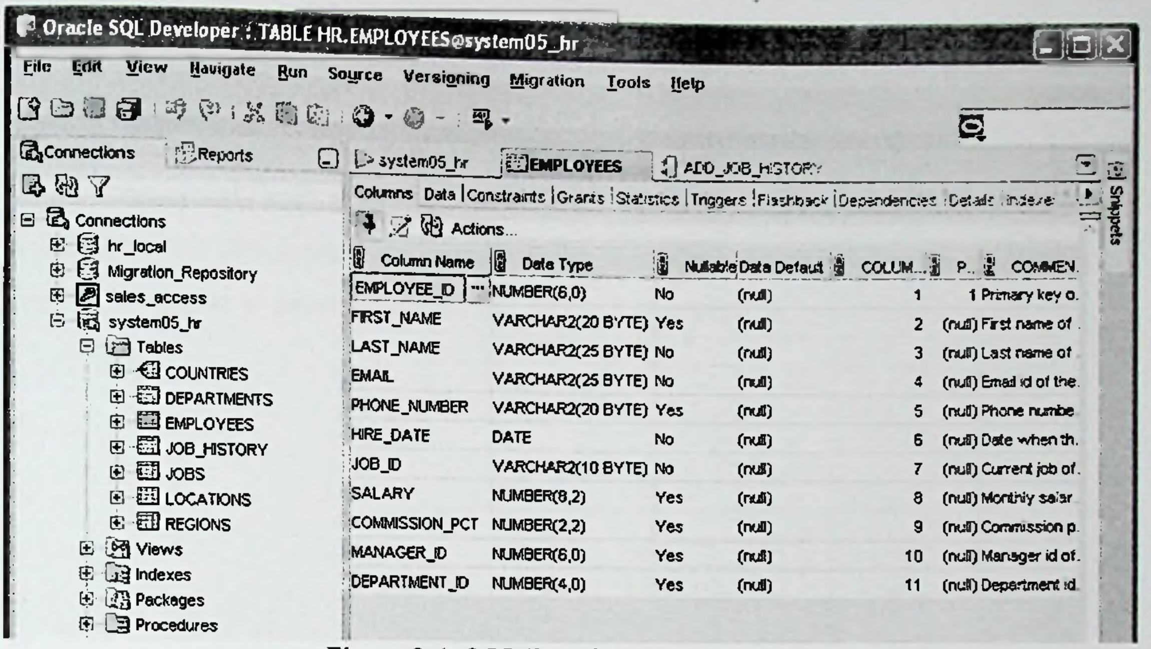


Figure 3.3: Microsoft SQL Server Management Studio

3.6 Oracle SQL Developer

Oracle SQL Developer [10] is an open source development environment. Its main functionality is to simplify the development and management of Oracle Databases. It includes worksheets for running queries and scripts, a DBA console for managing the database, reports interface, data modeling and a migration platform for moving a 3rd party databases to Oracle. Thereby it is considered a complete tool for the development of PL/SQL applications.



Column Name	Data Type	Nullable	Data Default	COLUM...	P..	COMMEN.
EMPLOYEE_ID	NUMBER(6,0)	No	(null)	1	1	Primary key o.
FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)	First name of .
LAST_NAME	VARCHAR2(25 BYTE)	No	(null)	3	(null)	Last name of .
EMAIL	VARCHAR2(25 BYTE)	No	(null)	4	(null)	Email id of the.
PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)	Phone numbe
HIRE_DATE	DATE	No	(null)	6	(null)	Date when th.
JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7	(null)	Current job of.
SALARY	NUMBER(8,2)	Yes	(null)	8	(null)	Monthly salar
COMMISSION_PCT	NUMBER(2,2)	Yes	(null)	9	(null)	Commission p.
MANAGER_ID	NUMBER(6,0)	Yes	(null)	10	(null)	Manager id of.
DEPARTMENT_ID	NUMBER(4,0)	Yes	(null)	11	(null)	Department id.

Figure 3.4: SQL Developer main window

3.7 The Java Database Connectivity (JDBC)

The Java Database Connectivity [15] API is used for connecting databases and other data sources (i.e. spreadsheets or flat files) to the Java programming language. The JDBC API provides a call level API for SQL based database access. This API enables the developers to use the capability of Java ("Write Once, Run Anywhere") for applications that need access to enterprise data. Developers can use the available drivers for connecting organizational data to source and destination databases. The JDBC API also allows for metadata access. This assists in the development of complex applications that require to define the abilities of a specific database connection.

JDBC technology uses the advantages that can be taken from internet-standard URLs in order to identify database connections. It implements a much better method of identifying and connecting to a data source. Namely by using a DataSource object which results making the code more flexible and maintenance friendly. Such objects enable connection pooling and distributed transactions. These functionalities are very much needed in enterprise database computing. Wherever the Java platform is located the JDBC API is available. Thereby enabling to access written database applications from the data location. The JDBC API provides server side functionality for assisting organizational success and is available in Java Platform, Standard Edition and Enterprise Edition.

3.8 Summary

This chapter presented technology adopted to develop an automated solution for database migration. The Application has been fully developed with Java technologies such as NetBeans Platform APIs and Java Database Connectivity API. There are other technologies to support for the development such as MySQL Workbench, pgAdmin, SQL Server Management Studio and Oracle SQL Developer. Next chapter shows the approach that how we are going to apply these technologies to develop the solution.

Chapter 4

A Novel Approach to Database Migration

4.1 Introduction

Chapter 3 presented the technologies for giving an automation solution for migrating and transforming schema and data across multiple databases. This chapter presents our approach to develop a database migration tool with industry's most utilized databases; Oracle, SQL Server, MySQL and PostgreSQL under several headings namely hypothesis, input output, processes, users and features. The chapter highlights the features that distinguish our novel approach from the existing approaches to database migration.

4.2 Hypothesis

Using database migration tool can be migrated, copied and transformed schema and data across multiple databases like Oracle, SQL Server, MySQL and PostgreSQL.

4.3 Inputs

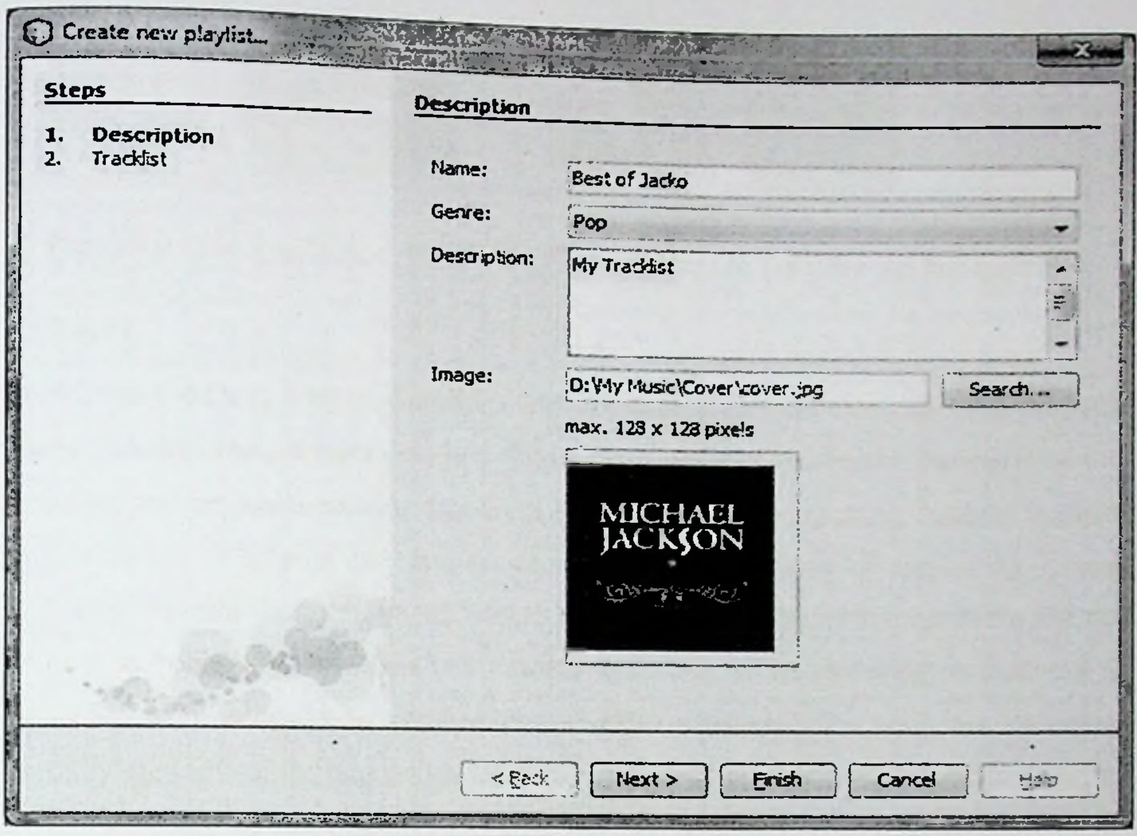
The main purpose of this migration tool is for selecting source and target databases by purifying the connection and then selecting the required tables for performing the migration. The tool enables to conduct data mapping and other migration exercises. The tool features a wizard that guides the user through a step-by-step process and thereby ensuring smooth and easy connectivity between source and destination databases. It also makes note of the issues that may arise when migrating data of different vendors and it efficiently transforms schema and data between databases. Therefore users can transform schema and define schema attributes while the migration is underway. Options to map source database schema to the destination database schema, functionalities that facilitate the selecting of only the necessary columns for migration and the mapping of columns in source database and some of the other special features included in this tool. It allows users to change the data type of table columns and across all the tables. It is possible to migrate schema and data together or either one individually if necessary. It uses a project based approach with regards to the organizing of the migrations and thereby it can accommodate regular migrations.

4.4 Outputs

This tool supports databases such as Oracle, MS SQL Server, MySQL and PostgreSQL. It is regards as a total Data Migration Tool while being flexible, user friendly, reliable and maintains data integrity. It displays the status of the migration data using reports. The tool maintained constraints and relationships among tables. Loss of data when migrating and moving data is mostly avoided. It comprises of easy ways to move new data to existing tables. It provides detailed and summary migration reports.

4.5.1 Process for Application Frontend

The NetBeans Platform [16] uses easy to work with wizards that give flexibility and usability in handling complex operations in applications. The wizard is hosted by the Dialogs API in order to help users work through a particular process in a step-by-step manner. Such strategies (wizards) are broadly used NetBeans IDE, for creating new windows or actions. The wizard offers easy to understand panels that can be used in each step for data entry and coordination. The following examples show how the wizards are used in designing a wizard and its architecture, in particular. In the below example a wizard is created for making playlists. The wizard gives two steps; the first step enables for adding a description for the playlist Figure 4.1 while the second enables for adding the music titles allows users to describe the playlist. The solution is based on the NetBeans Platform Wizard Architecture APIs.


Create new playlist...

Steps

1. Description
2. Tracklist

Description

Name: Best of Jacko

Genre: Pop

Description: My Tracklist

Image: D:\My Music\Cover\cover.jpg

max. 128 x 128 pixels

Figure 4.1: First step of a sample wizard

4.5.2 Process for Application Backend

Java Database Connectivity API which is used by the application to interact with the database. JDBC API makes Java application almost independent of the database it uses. The application has been using JDBC API to interact with the database, then we need not to change much in the code each time to change the database of the application. Application can switch easily from one database to another. JDBC API provides standard mechanism to establish a connection with the database, to send the queries to the database, to navigate the results returned by the database and to update a database. The application, JDBC API and Database can be schematically represented as below Figure 4.2. JDBC driver is a software component which is required by the JDBC API to interact with the database. Each database will have their own JDBC driver. This driver is supplied by the database vendor along with the database. As an example, JDBC driver of Oracle is ojdbc6.jar and it can be downloaded from Oracle websites.

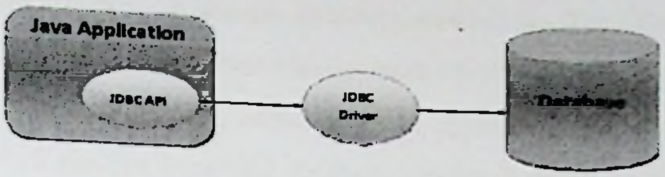


Figure 4.2: The Application, JDBC API, JDBC Driver and Database can be represented

4.6 Users

Sometimes users may want to reconstruct and restructure tables while they are being migrated. These instances require more than just plain schema and data movement. Examples of such instances include while moving data from master database to reporting database applying transformation of schema and data for faster querying, migration of application versions migrating between vendor solutions, etc. In these situations users need the schema and data while it is being migrated. This tool enables to do this while improving productivity for database administrators who are handling multiple databases. The easy installation and usability are some of the plus points for administrators or users to be comfortable using this tool. In summary the entire migration is done fast and seamlessly.

4.7 Features

When moving from an old to a new database version there is always doubt regarding the accuracy if the migration as it is never easy to upgrade to latest (E.g., SQL Server 2012 to SQL Server 2014). Thereby in some cases upgrades are postponed and avoided. Nevertheless this tool gives users a simple yet effective migration that is accurate. By the inclusion of options such as table and column name changes, data type mapping, selecting specific columns, column mappings etc., and the tool successfully performs the migrations without glitch. Additional features include, selecting columns to be migrated, table and column name changes, data type changes, column mapping in destination database, applying functions on data while migration and etc. The fact that the tool uses wizard for the process enhances its simplicity in usage and improves its usability.

For the databases management, we have used four software tools which are MySQL Workbench, pgAdmin, SQL Server Management Studio and Oracle SQL Developer. In the chapter 3, we have given a complete description of these software tools. MySQL databases have been managed by MySQL Workbench. The GUI in pgAdmin accommodates all

PostgreSQL database features and thereby improves its usability. The Object Explorer can be regarded as a main feature in SQL Server Management Studio, as it enables the user to browse, select, and act upon any of the SQL server database objects within the server. Oracle SQL Developer offers facilities for exploring database objects, running queries and scripts and managing the oracle database using database administrator console.

Need to make sure the installation which includes the Java Runtime Environment (JRE). Installation is pretty straight forward. This product is based on Java, and requires Java Runtime Environment. For this product to work, the machine has to be made sure that it is not blocking Java.

4.8 Summary

This chapter presented our novel approach to develop an automated solution for database migration. In this sense, it is pointed out how the novel approach offers an efficient and accurate solution for database migration across multiple databases. The NetBeans Platform offers an API for creating dialogs and wizards. It is easy for developers to learn how to create simple and complex dialogs in the NetBeans Platform. Also the developer is able to create structured wizard that can guide the user through the process easily and quickly. Next chapter shows the design of the novel approach presented here.

Chapter 5

Solution Design

5.1 Introduction

Chapter 4 presented the approach to develop an automated solution for migrating schema and data across multiple databases migration. This chapter elaborates the approach and describe the architecture of the solution. NetBeans Platform Wizard Architecture and Java Database Connectivity are the main foundation for this application. NetBeans Platform Wizard Architecture facilitates to design a wizard programmatically. It guides the user through an easy to use and understand step-by-step process of migration. This is regarded as a total Database Migration Tool. It assists in migrating and transferring database schemas and data in Oracle, Microsoft SQL Server, PostgreSQL, and MySQL databases using Java Database Connectivity (JDBC).

5.2 Wizard Architecture

The WizardDescriptor is a subclass of the DialogDescriptor class which describes and configures a wizard in principle. DialogDescriptor class, in turn, is a subclass of NotifyDescriptor. The WizardDescriptor controls the wizard by way of containing and managing the panels and baring responsibility for controlling the action buttons and displaying the table of contents and other key tasks.

The WizardDescriptor has a data model in order to collect data which is saved as properties. Although the developer can (if preferred) provide his own data model. A panel is showed for each step of the wizard which is built using two classes. While the first class (This class is known as the Visual Panel, and normally extends JPanel) handles the implementation of the GUI the second class does the management and validation of the panel (this is known as the Wizard panel). The first class implements the GUI. This class extends the WizardDescriptor.Panel<Data> class. The class creates the visual panel when required in order to make it available for the wizard. As per the MVC (Model-View-Controller) paradigm, the Visual Panel is regarded as the view while the Wizard Panel supplies the control. As the purpose of the Visual Panel is to enable the user to enter data it should not include business logics nor does not deal with wizard specific classes.

Therefore the panel is completely reusable even outside of the wizard as in a dialog used for editing of a playlist. The relationship between WizardDescriptor, WizardPanel, and VisualPanel is illustrated again in Figure 5.1.

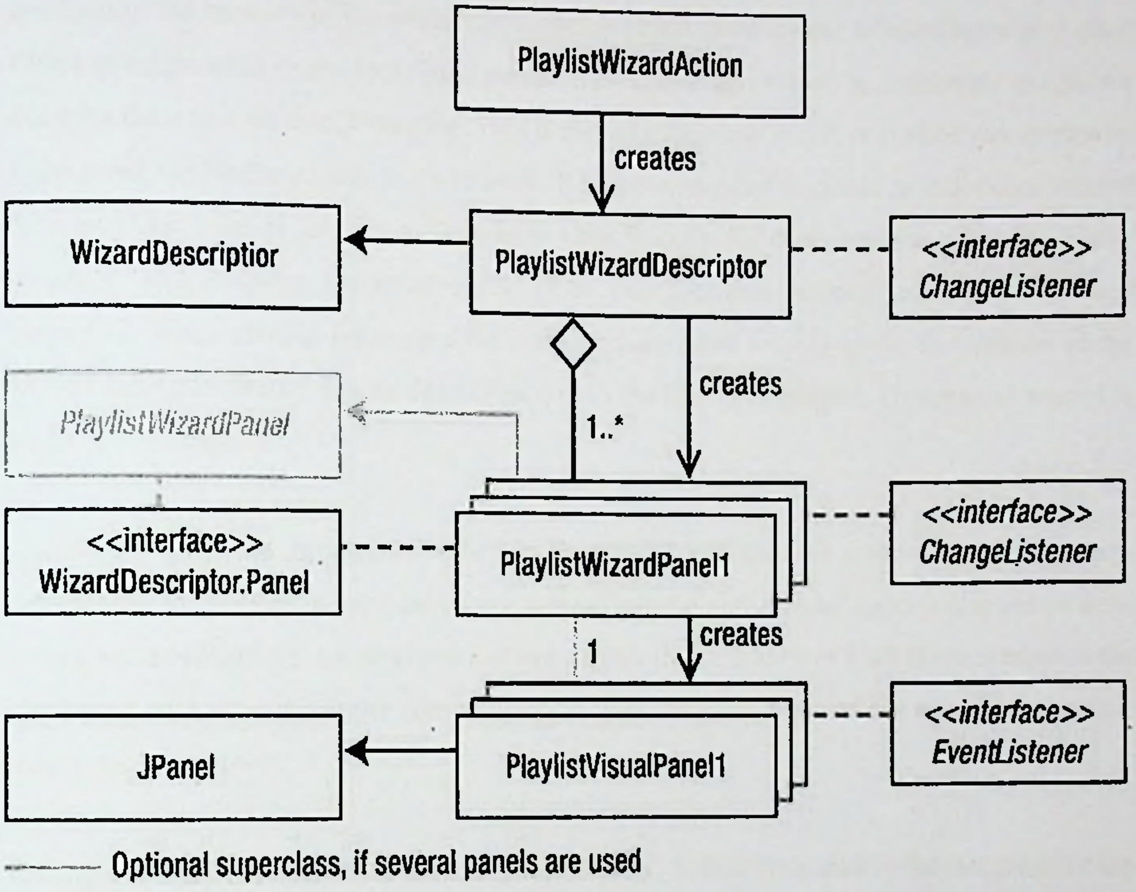

PlaylistWizardAction
creates
WizardDescriptior
PlaylistWizardDescriptor
<<interface>> ChangeListener
creates
PlaylistWizardPanel
1..*
<<interface>> WizardDescriptor.Panel
PlaylistWizardPanel1
<<interface>> ChangeListener
1
creates
JPanel
PlaylistVisualPanel1
<<interface>> EventListener
Optional superclass, if several panels are used

Figure 5.1: Architecture of a wizard

Creating Panels: The basic structure of the wizard is created in NetBeans platform. First creates the user interface of the first panel which afterwards goes on to create the structure/layout of the panels inclusive of a visual and wizard panel. The developer is able to edit the content for predefined methods. Initially the UI of the first panel needs to be created. This would be the class VisualPanel1 in NetBeans Form Editor. It is important that many fields are defined so that the user is able to describe the UI in detail while choosing a genre and providing a description and image. The completed panel should look like Figure 5.1, whereby the panel is already shown combined into the wizard. The panel is a normal Swing component, extending JPanel.

Creating a Wizard from Panels: Up to this point it has been explained how to create a single panel of the wizard that would be a step in the wizard. It included a description on how the tasks of view and controller are distributed and separated. Now for the final step before completing the creation of the wizard; a wizard is represented by the WizardDescriptor class which manages each of the individual panels. The class can be used to instantiate the panels and pass them to a WizardDescriptor. This is the way the class which is created automatically when using the NetBeans IDE is set to work. It is recommended to create an individual wizard descriptor that extends the WizardDescriptor class in order for encapsulation, making a better structure, and enabling reusability. The class can therefore handle creating panels and properties. When starting the wizard the action classes need only to create an instance of the wizard descriptor, which can be directly passed to the DialogDisplayer. Thereby the wizard is made fully transparent.

Event Handling: The sequence diagram in Figure 5.2 explains the communication concept between the three layers wizard descriptor, wizard panel and the visual pane. It also shows how events and notifications are received and processed. In the matter of both these scenarios the displaying of a wizard and the communication with different parts of the wizard as the user enters data are shown.

Ending a Wizard Prematurely: As usually the "Finish" button is located in the last panel of the wizard it is necessary to give the users the option to cancel the wizard before coming to the last panel, as it may serve the purpose of the user to do so. The interface WizardDescriptor.FinishablePanel needs to be implemented on the panel to enable this. This specifies the method isFinishPanel() with which developer can return the value true, if the wizard can be finished. As is shown in the given example, it is a good idea to add this option from the first panel onwards as then the user has the option to end the wizard prior to entering data if required.

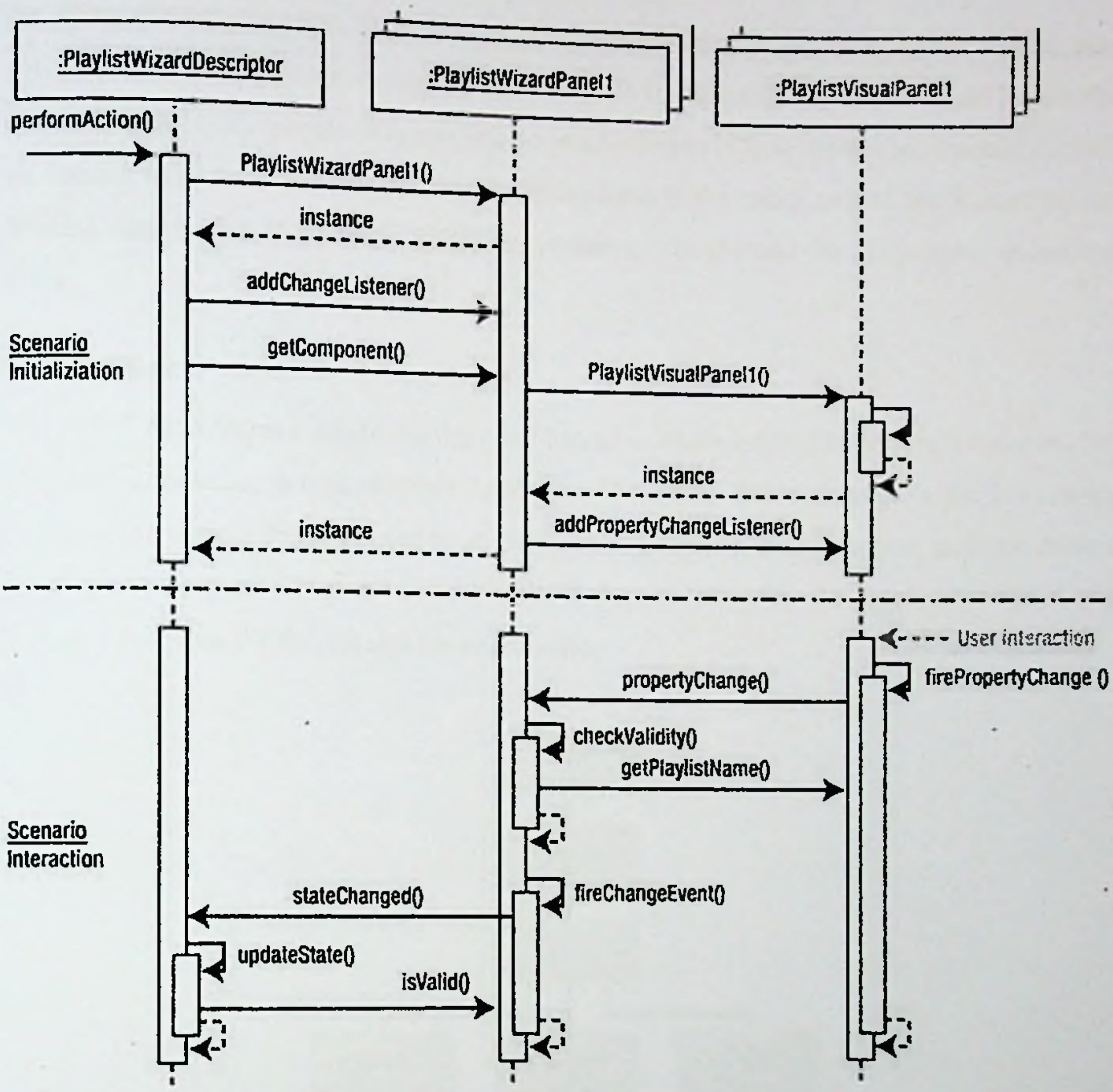


Figure 5.2: Sequence diagram for the wizard objects

Additional Validation of Data: The validity of the data of a panel is notified to the wizard using the isValid() method. This method is called when opening a panel and on notification by the ChangeListener. If when the user clicks next and thereby addition verifications are done the WizardDescriptor.ValidatingPanel interface could be implemented. This interface conducts detailed verifications using the validate() method. If there are any errors, they are notified to the WizardValidationException. In the case of an error the user receives a JComponent in focus, which can be accompanies with the failure message if needed.

Iterators: The descriptor panels of the wizard is managed by an iterator which handles the order of the panels. Such an iterator is implemented by the WizardDescriptor.Iterator class which sets the panels in sequential order. This class is also used when passing panels as an array to the WizardDescriptor class. Nevertheless it is important that when enabling the user to skip certain panels to the iterator implementation to the WizardDescriptor, as this would handle the dynamic order of the panels. It is possible to use NetBeans IDE to create the structure of such an iterator to be used in a wizard. When coming back to the initial step of the Wizard set the Wizard Step Sequence to Static as setting Dynamic would make the IDE create an iterator class.

5.3 Database Connectivity Architecture

The JDBC API utilizes a driver manager that includes database-specific drivers in order to give a smooth connection across multiple databases. The JDBC driver manager's job is to make sure that the correct driver is used to access each data source. It can support multiple drivers concurrently. Figure 5.3 gives an architectural diagram showing the location of the driver manager based on the drivers and the application.

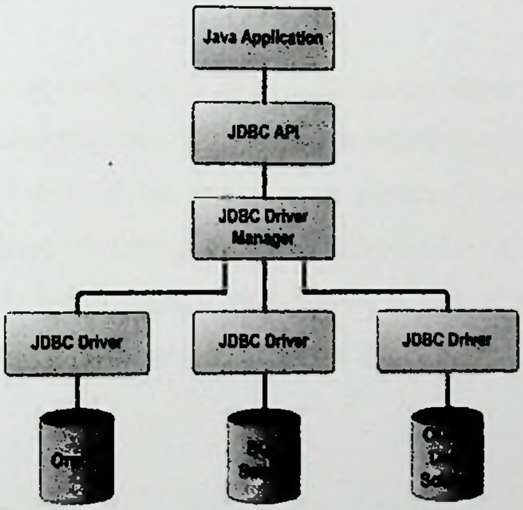


Figure 5.3: JDBC Architectural Diagram

In the solution includes the use of the following interfaces and classes in the JDBC API:

- DriverManager: The purpose of this class is to manage the database drivers and match connection requests between the application and the appropriate driver by communicating through the sub protocol. Thereby the database connection will be established using the first driver that recognizes a certain sub protocol under JDBC.
- Driver: As only very little direct communication will be done with the drivers this interface will handle the communications with the database server. Thereby the

DriverManager objects will be used instead to manage objects of this type. It will also abstract the details that are linked to the driver objects that will be used. .

- Connection: The connection interface includes all methods related to connecting to a database. Thereby all communication with the database is done only through connection object.
- Statement: This interface is used to submit the SQL statements to the database. Some interfaces that derive from this interface are able to accommodate parameters in addition to executing procedures.
- ResultSet: The data that is received from a database after executing an SQL query using Statement objects is held by these type of objects. It acts as a mediator to enable to move its data through.
- SQLException: The errors that occur in the database is handled through this class.

5.4 Summary

This chapter presented design architecture of the automated solution for database migration. It showed architecture of the solution which is going to be implemented in the development stage. This design offers a well-defined and user-friendly solution for database migration across multiple databases. NetBeans Platform provides a professional APIs for the frontend and Java Database Connectivity defines interfaces and classes to communicate with the backend databases. Next chapter shows the implantation of the solution.

Chapter 6

Implementation of the solution

6.1 Introduction

Top level design is based on two major components, the first one is NetBeans platform wizard architecture for frontend and the second one is Java Database Connectivity architecture for backend. This chapter describes the implementation of the automated solution for database migration. In that sense this chapter is about how the system is implemented. The solution has been implemented to run on platform independent manner on Windows, Linux, Mac OS and etc. It has been developed on top of Java 8 platform. Let's discuss the implementation of the automated solution for database migration across multiple databases.

6.2 Implementation

The solution is loaded with splash screen (Figure 6.1).

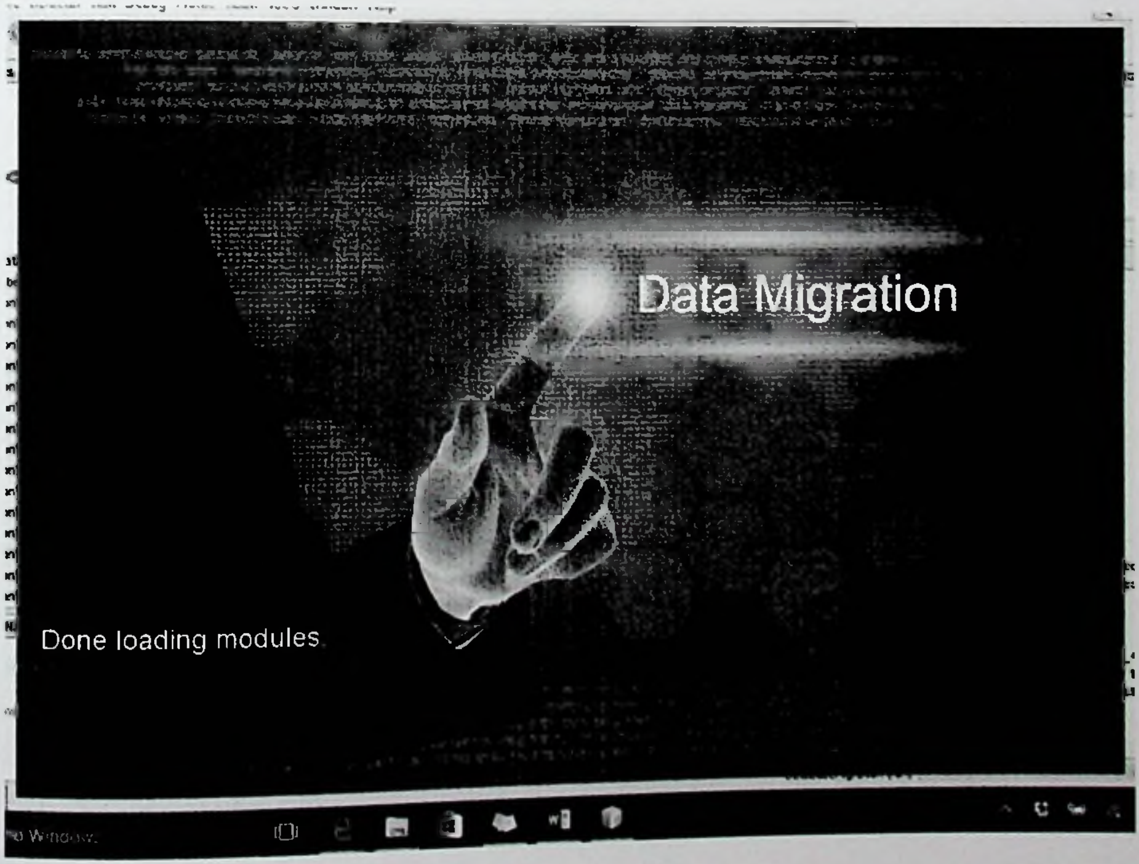


Figure 6.1: Splash screen

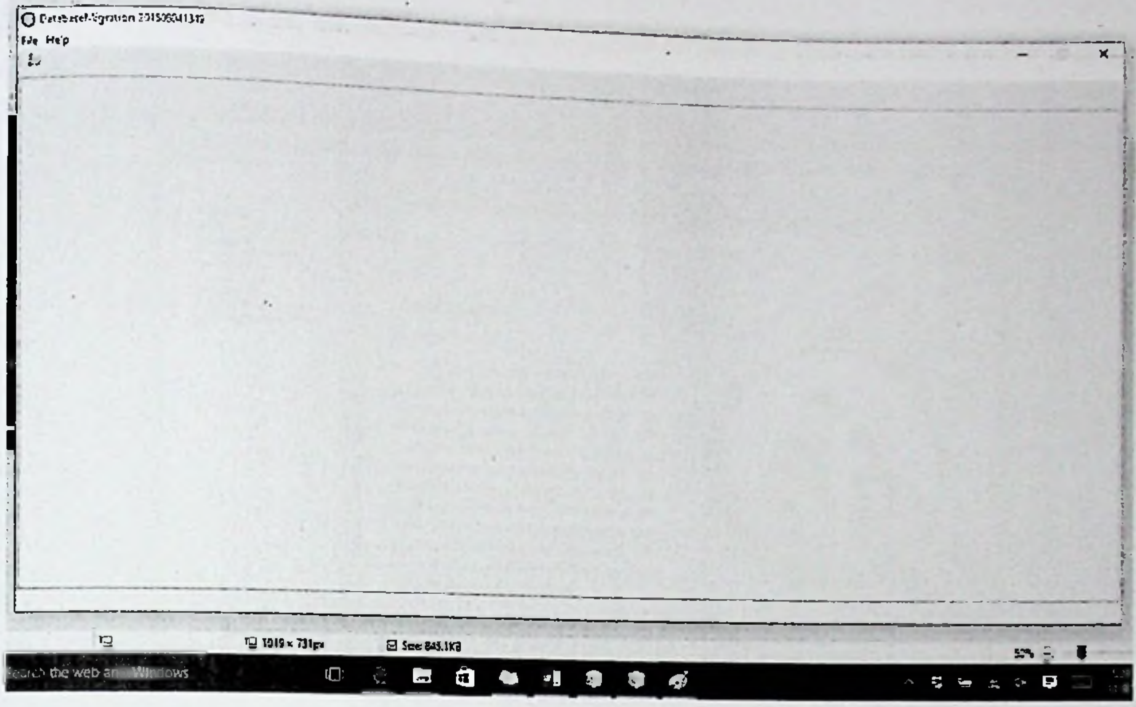


Figure 6.2: The application on top of the NetBeans Platform

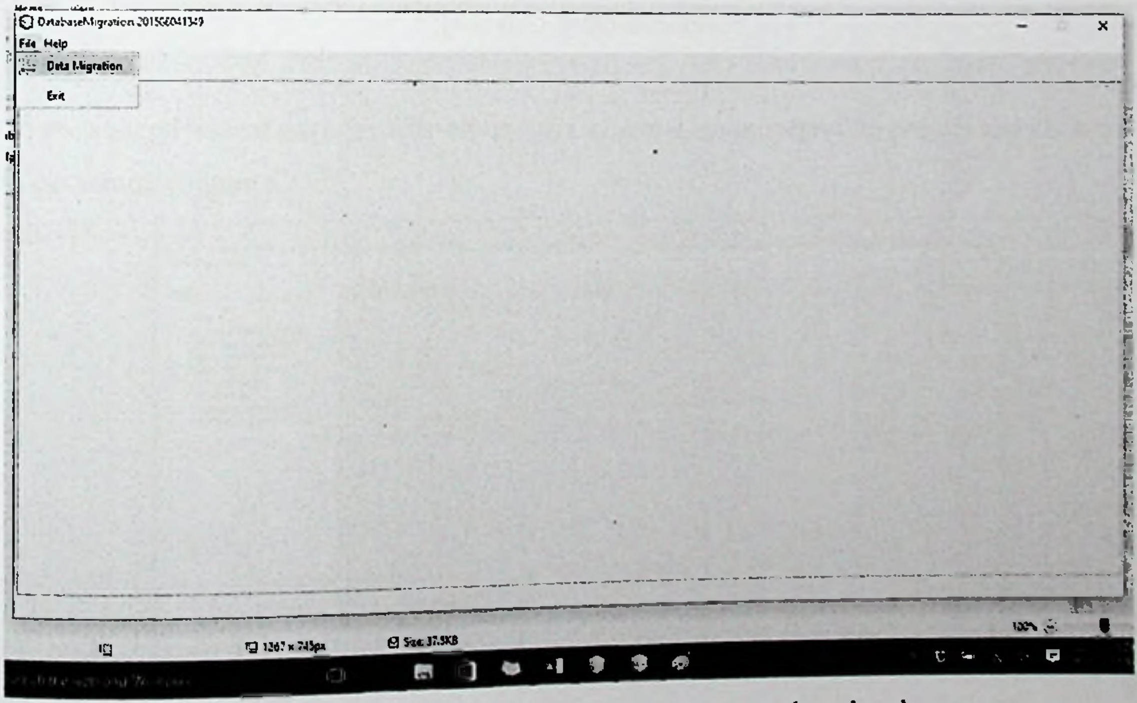
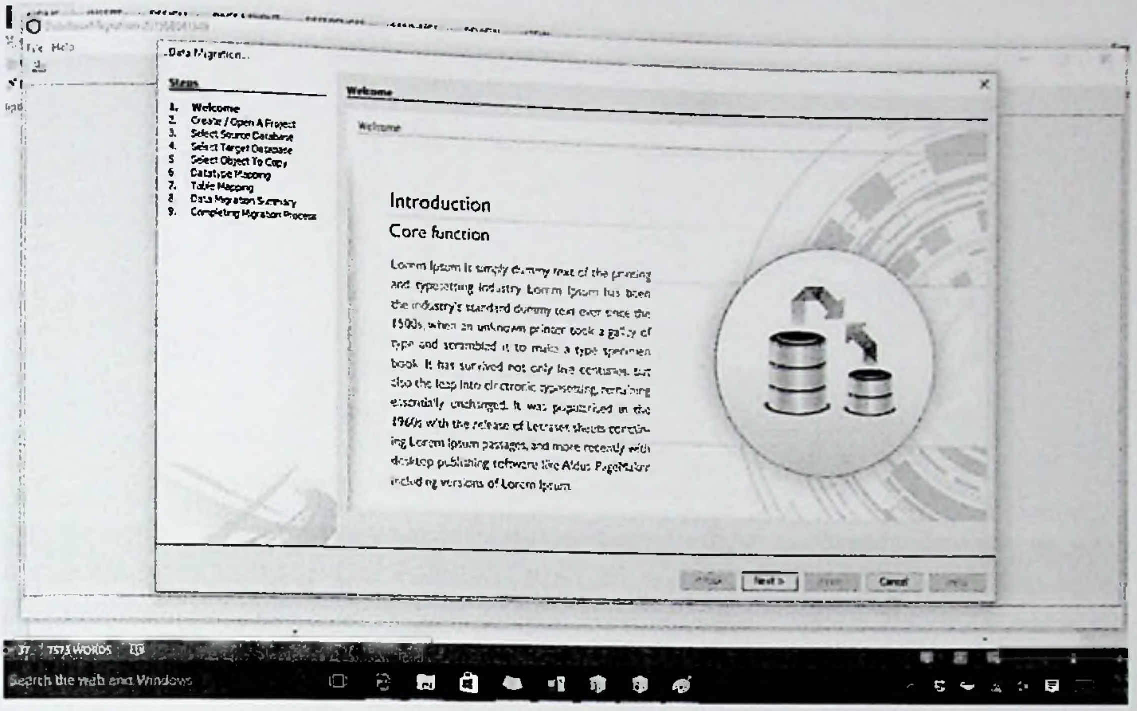


Figure 6.3: Clicking the menu item, it shows the wizard

Users are now walked through the steps relating to the migrating of tables from one RDBMS to another. As per Figure 6.4, the user is offered a wizard which has listed on the left window the 9 steps in the migration process.

In the first wizard page, the user can see the welcome screen and click the Next button (Figure 6.4).



..Data Migration..

Steps

1. **Welcome**
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Welcome

Welcome

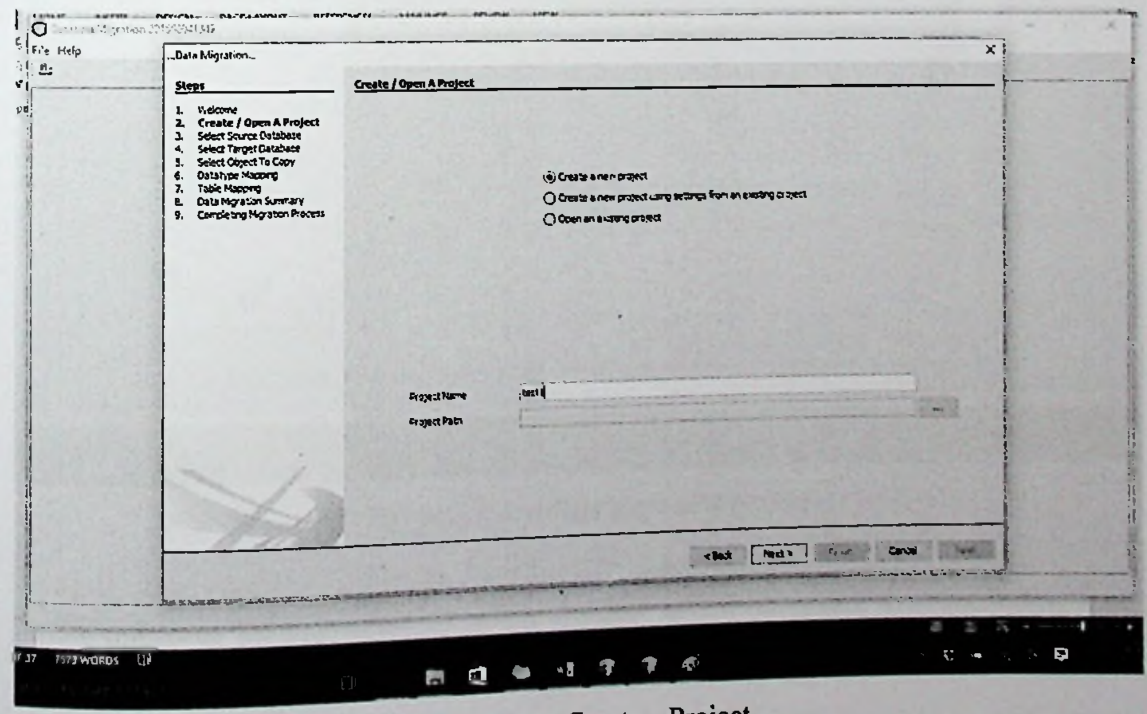
Introduction

Core function

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Figure 6.4: Welcome screen

In the second wizard page the user either have to create a new migration project and click the Next button (Figure 6.5).



..Data Migration..

Steps

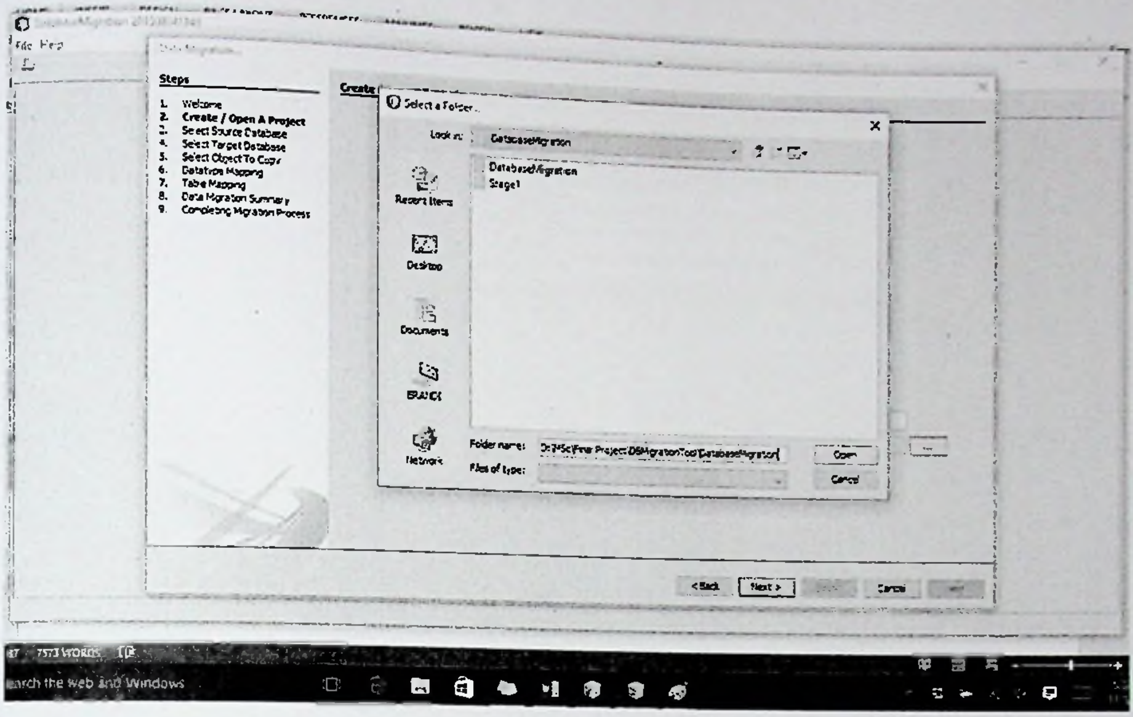
1. Welcome
2. **Create / Open A Project**
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Create / Open A Project

[✓ Create a new project]
Create a new project using settings from an existing project
Open an existing project

Project Name: test
Project Path:

Figure 6.5: Create a Project



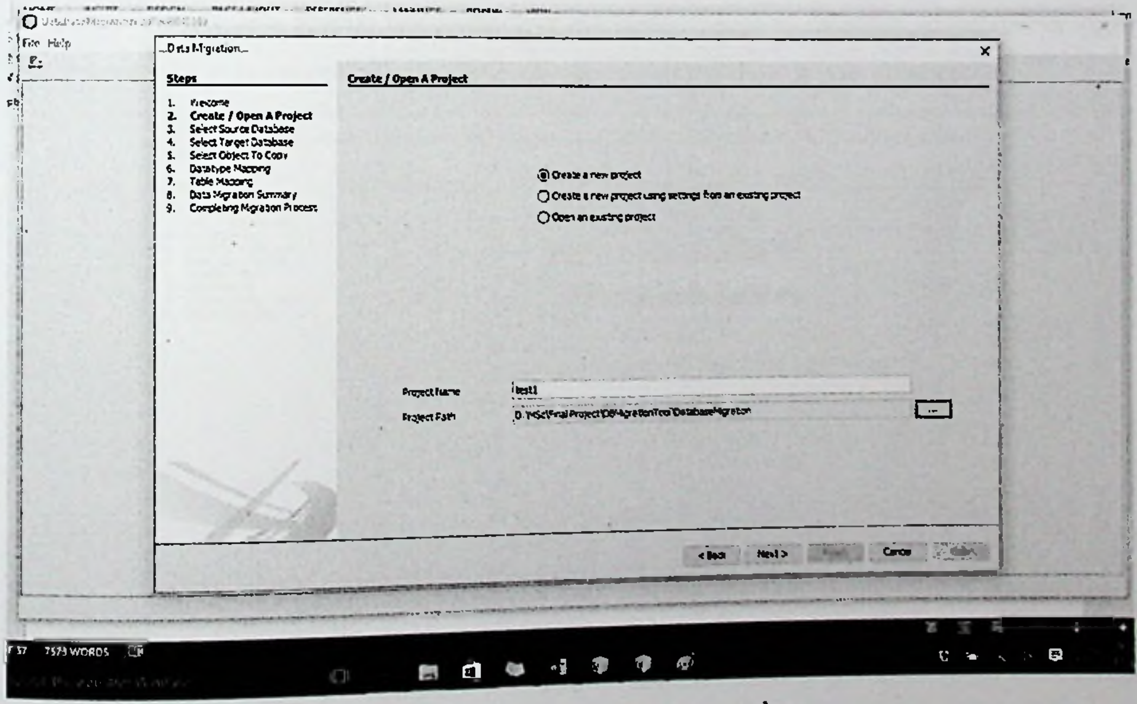
Look in: DatabaseMigration

DatabaseMigration
Stage1

Folder name: D:\MSc\Final Project\DBMigrationTool\DatabaseMigration

Files of type:

Figure 6.6: Select a project path


File Help
Data Migration
Steps
1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process
< Back
Next >
Cancel

Create / Open A Project

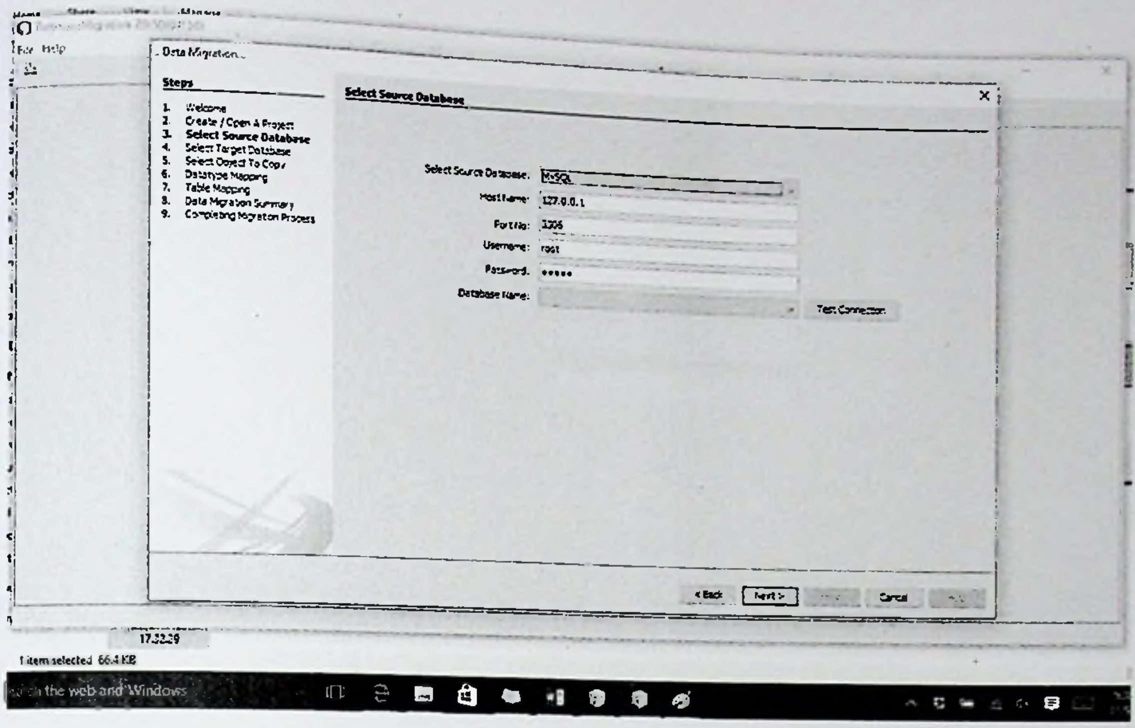
[✓ Create a new project]
Create a new project using settings from an existing project
Open an existing project

Project Name: test1

Project Path: D:\MSc\Final Project\DBMigrationTool\DatabaseMigration

Figure 6.7: Configuring a new project

Figure 6.8 leads the user to step 3, where the user selects his source database management system. Select the database server instance from the 'Select source database' combo box, which can be either Oracle or SQL Server or PostgreSQL or MySQL.



Data Migration...

Steps

1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Select Source Database

Select Source Database: MySQL

Host Name: 127.0.0.1

Port No: 3306

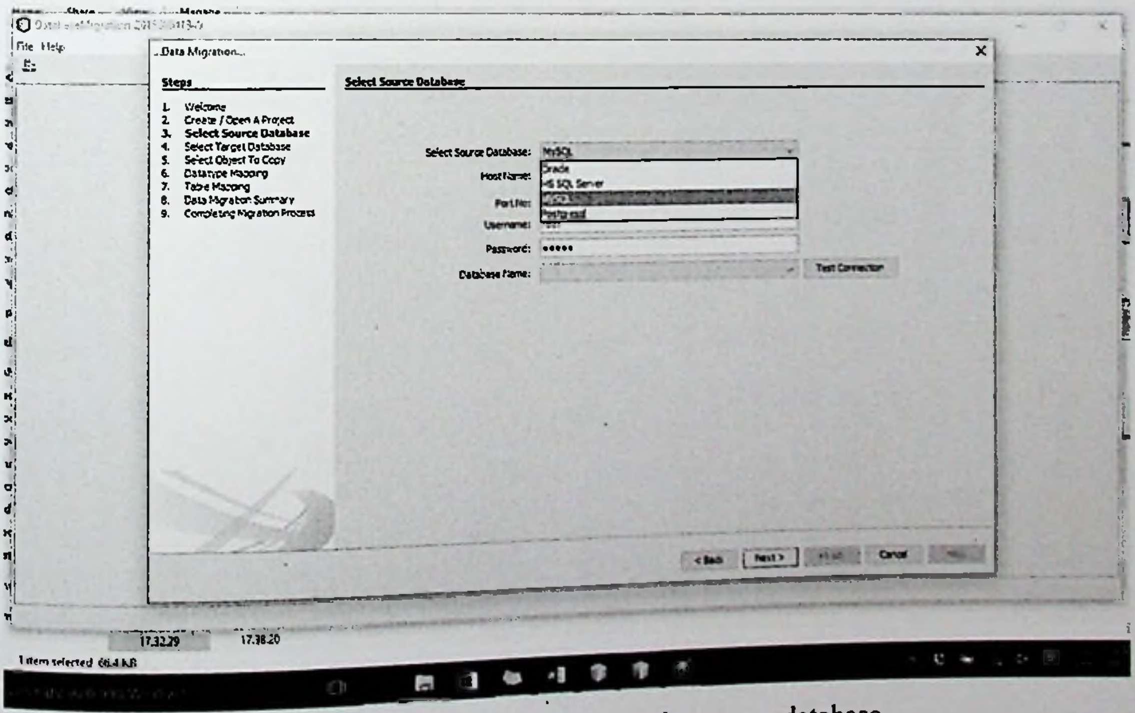
Username: root

Password: •••••

Database Name:

Figure 6.8: Select a Source Database

This step lets the user specify the connective information such as database server name, login and password for connecting to the database server and the default port number.



Data Migration...

Steps

1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Select Source Database

Select Source Database: MySQL

Oracle
MS SQL Server
MySQL
PostgreSQL

Host Name:

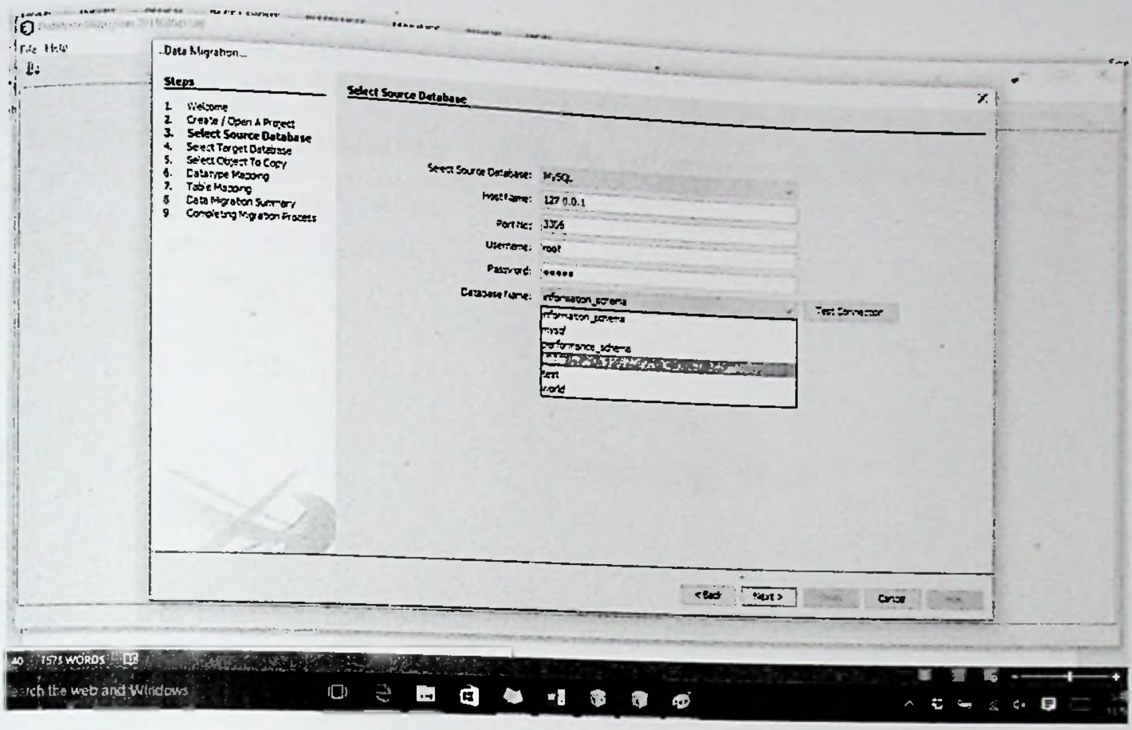
Port No:

Username:

Password: •••••

Database Name:

Figure 6.9: Select a specific vendor source database



Data Migration...

Steps

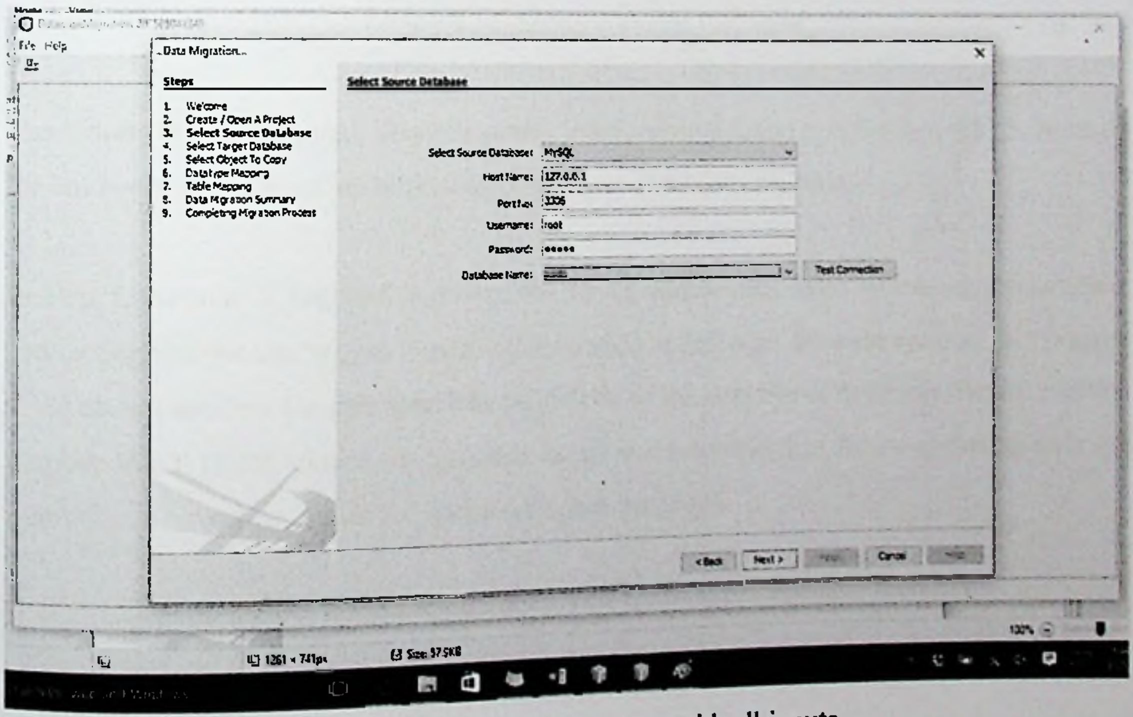
1. Welcome
2. Create / Open A Project
3. **Select Source Database**
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Select Source Database

Select Source Database: MySQL
Host Name: 127.0.0.1
Port No: 3306
Username: root
Password: •••••
Database Name: information_schema

information_schema
mysql
performance_schema
test
world

Figure 6.10: Clicking on "Test Connection" button, the dropdown will be populated with all database names



Data Migration...

Steps

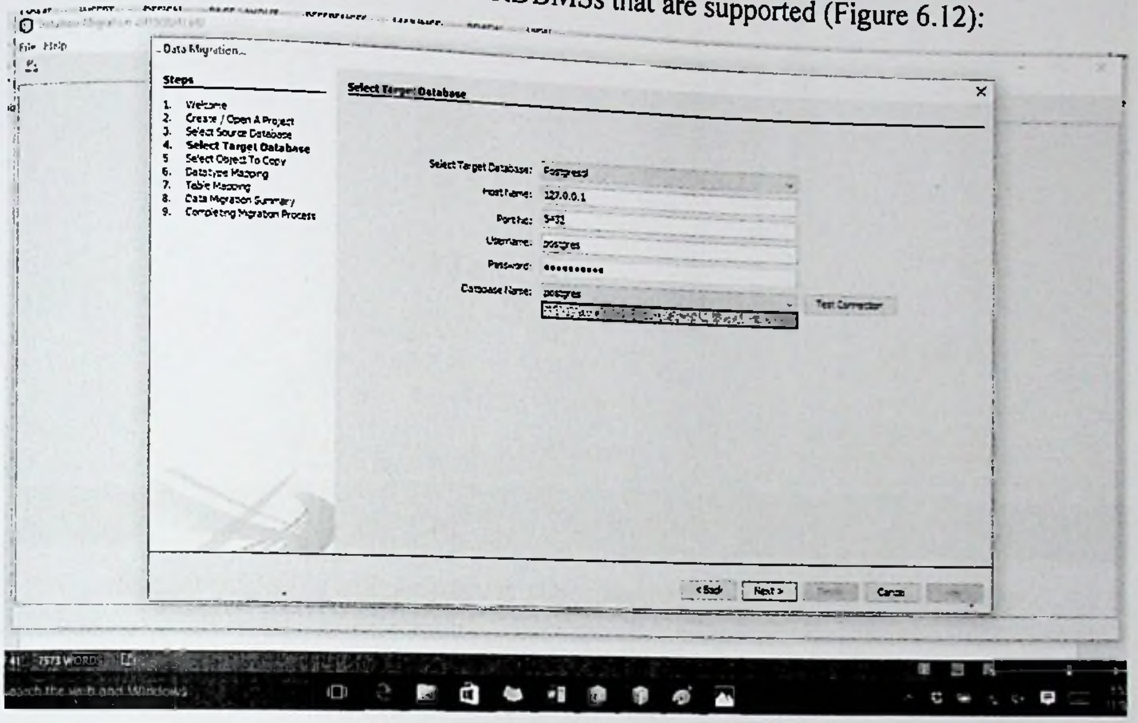
1. Welcome
2. Create / Open A Project
3. **Select Source Database**
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Select Source Database

Select Source Database: MySQL
Host Name: 127.0.0.1
Port No: 3306
Username: root
Password: •••••
Database Name: [illegible]

Figure 6.11: Source Database with all inputs

Enter the connectivity information for the source database and click on Next. In step 4 the user needs to select a target database management system using the 'Select target database' combo box. This combo box includes a list of RDBMSs that are supported (Figure 6.12):


Data Migration

Steps

1. Welcome
2. Create / Open A Project
3. Select Source Database
4. **Select Target Database**
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Select Target Database

Select Target Database: Postgresql

Host Name: 127.0.0.1

Port No: 5432

Username: postgres

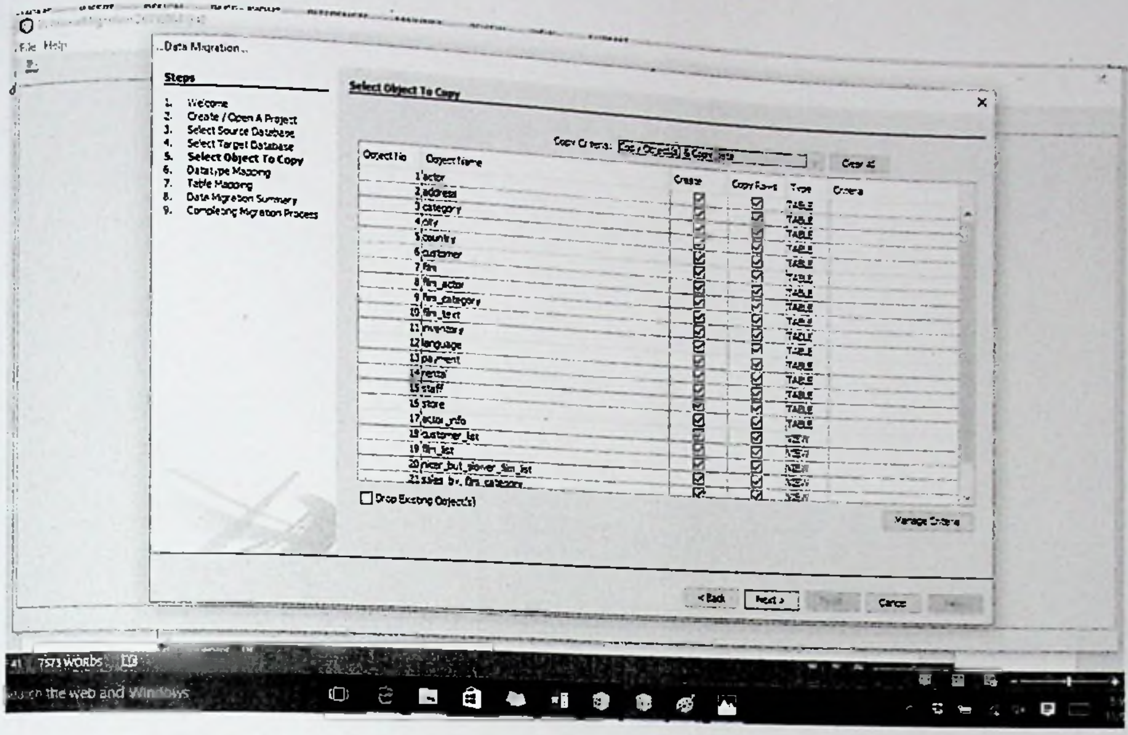
Password: ••••••••••

Database Name: postgres

Figure 6.12: Select a Target Database

User enters the server name, database name, login, password and port number which make up the connectivity information of the target database and clicks on Next.

In step 5 the user is required to select the tables and/or data from which source database connection that the user wishes to perform migration to the target database system. This (Figure 6.13) screen also lets the user specifies the criteria of the data that is to be transferred. For this purpose the WHERE clause for the table needs to be specified in the rows that qualify for migration. This action is also for future reference purposes.



..Data Migration..

Steps

1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. **Select Object To Copy**
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

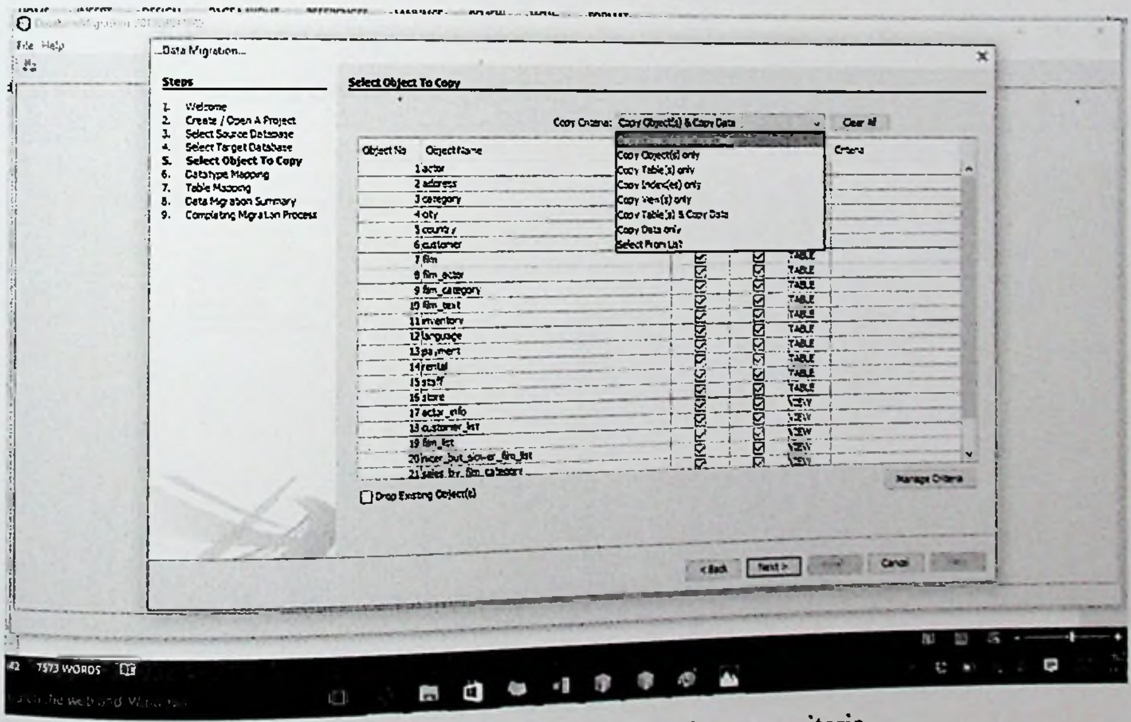
Select Object To Copy

Copy Criteria: Copy Object(s) & Copy Data

Object No	Object Name	Create	Copy Rows	Type	Criteria
1	actor	☑	☑	TABLE	
2	address	☑	☑	TABLE	
3	category	☑	☑	TABLE	
4	city	☑	☑	TABLE	
5	country	☑	☑	TABLE	
6	customer	☑	☑	TABLE	
7	film	☑	☑	TABLE	
8	film_actor	☑	☑	TABLE	
9	film_category	☑	☑	TABLE	
10	film_text	☑	☑	TABLE	
11	inventory	☑	☑	TABLE	
12	language	☑	☑	TABLE	
13	payment	☑	☑	TABLE	
14	rental	☑	☑	TABLE	
15	staff	☑	☑	TABLE	
16	store	☑	☑	TABLE	
17	actor_info	☑	☑	VIEW	
18	customer_list	☑	☑	VIEW	
19	film_list	☑	☑	VIEW	
20	nicer_but_slower_film_list	☑	☑	VIEW	
21	sales_by_film_category	☑	☑	VIEW	

☐ Drop Existing Object(s)

Figure 6.13: The user selects the tables and/or data from the source database



..Data Migration..

Steps

1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. **Select Object To Copy**
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Select Object To Copy

Copy Criteria: Copy Object(s) & Copy Data

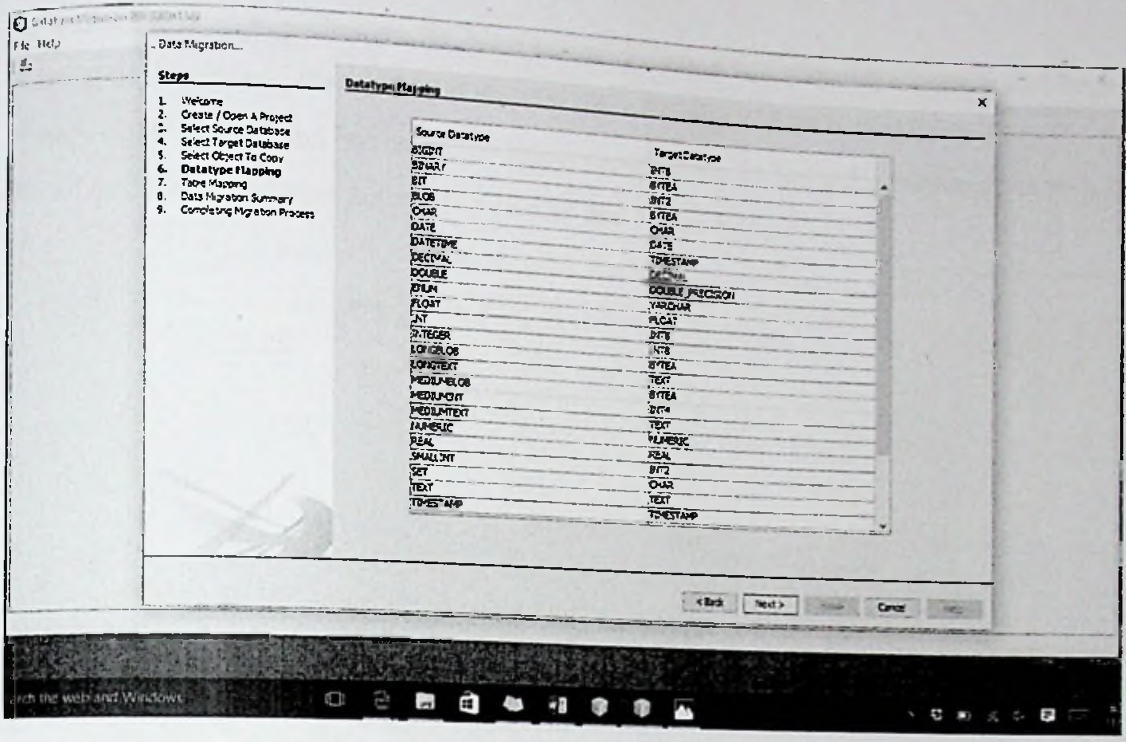
- Copy Object(s) only
- Copy Table(s) only
- Copy Index(es) only
- Copy View(s) only
- Copy Table(s) & Copy Data
- Copy Data only
- Select From List

Object No	Object Name	Criteria
1	actor	
2	address	
3	category	
4	city	
5	country	
6	customer	
7	film	
8	film_actor	
9	film_category	
10	film_text	
11	inventory	
12	language	
13	payment	
14	rental	
15	staff	
16	store	
17	actor_info	
18	customer_list	
19	film_list	
20	nicer_but_slower_film_list	
21	sales_by_film_category	

☐ Drop Existing Object(s)

Figure 6.14: The user can select the copy criteria

The step 6 shows the default data type mappings. Data type mapping gives options for modifying data types of all the database tables.



Data Migration...

Steps

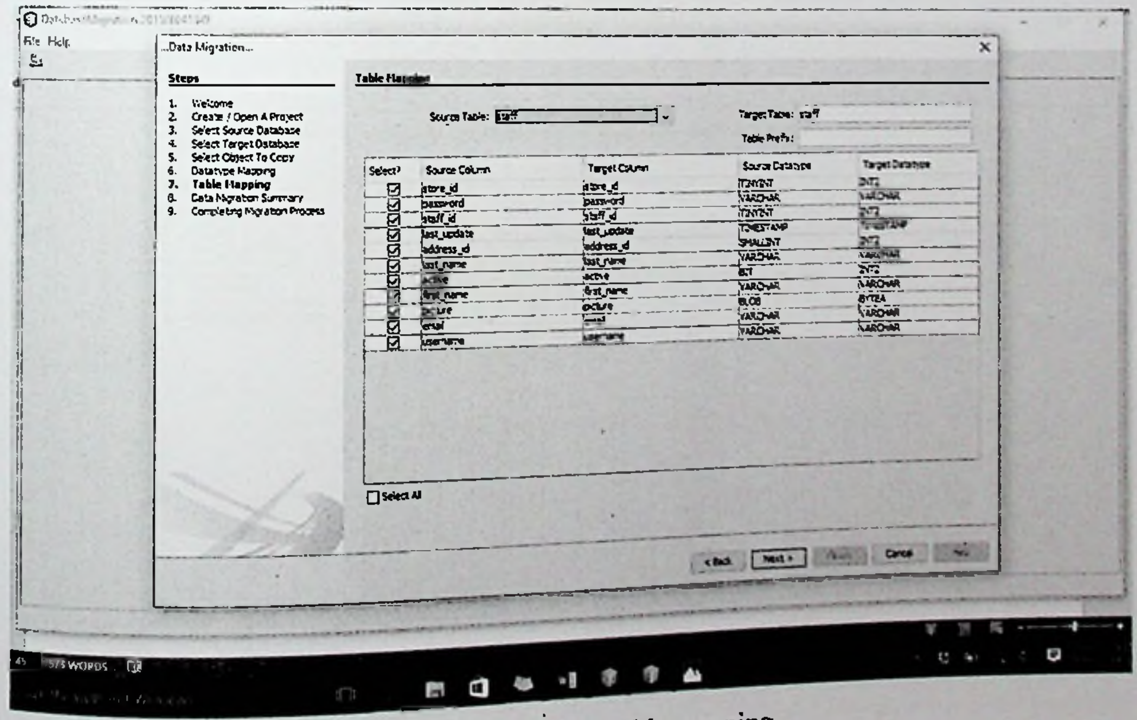
1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. **Datatype Mapping**
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process

Datatype Mapping

Source Datatype	Target Datatype
BIGINT	INT8
BINARY	BYTEA
BIT	INT2
BLOB	BYTEA
CHAR	CHAR
DATE	DATE
DATETIME	TIMESTAMP
DECIMAL	DECIMAL
DOUBLE	DOUBLE_PRECISION
ENUM	VARCHAR
FLOAT	FLOAT
INT	INT8
INTEGER	INT8
LONGBLOB	BYTEA
LONGTEXT	TEXT
MEDIUMBLOB	BYTEA
MEDIUMINT	INT4
MEDIUMTEXT	TEXT
NUMERIC	NUMERIC
REAL	REAL
SMALLINT	INT2
SET	CHAR
TEXT	TEXT
TIMESTAMP	TIMESTAMP

Figure 6.15: The step 6 shows the default data type mappings

In the step 7, the user can select a table in the source table combo box. Then the user can see the selected table's full details as Figure 6.16.



Data Migration...

Steps

1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. **Table Mapping**
8. Data Migration Summary
9. Completing Migration Process

Table Mapping

Source Table: staff

Target Table: staff

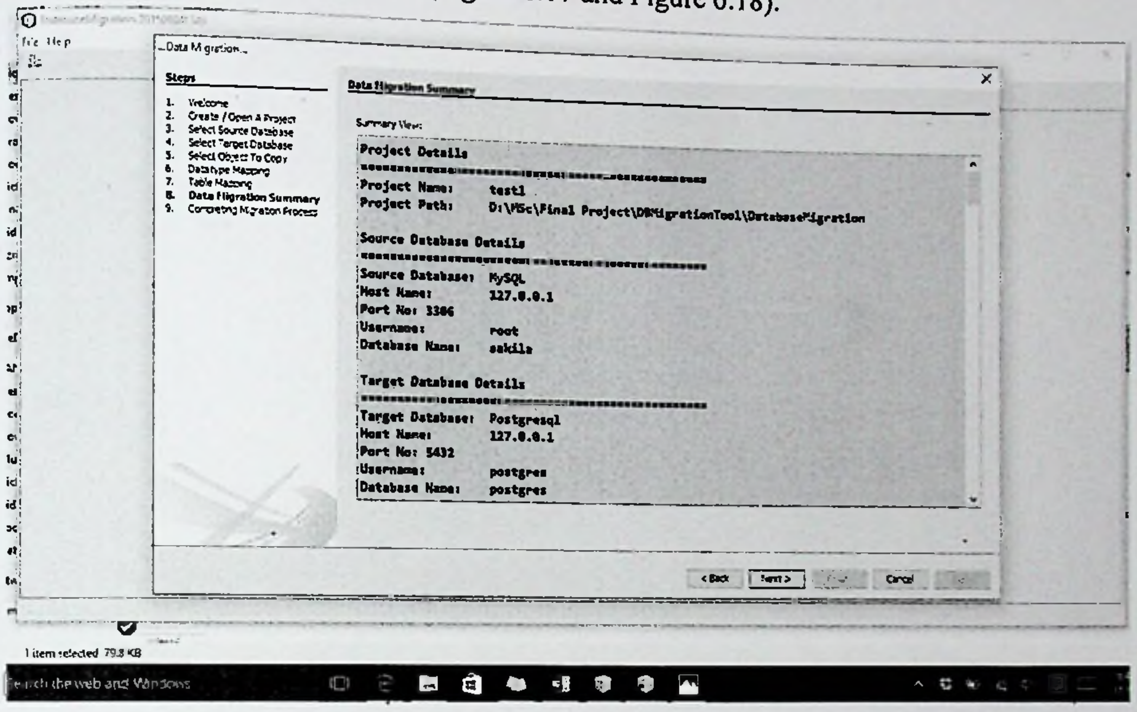
Table Prefix:

Select?	Source Column	Target Column	Source Datatype	Target Datatype
☑	store_id	store_id	TINYINT	INT2
☑	password	password	VARCHAR	VARCHAR
☑	staff_id	staff_id	TINYINT	INT2
☑	last_update	last_update	TIMESTAMP	TIMESTAMP
☑	address_id	address_id	SMALLINT	INT2
☑	last_name	last_name	VARCHAR	VARCHAR
☑	active	active	BIT	INT2
☑	first_name	first_name	VARCHAR	VARCHAR
☑	picture	picture	BLOB	BYTEA
☑	email	[illegible]	VARCHAR	VARCHAR
☑	username	username	VARCHAR	VARCHAR

☐ Select All

Figure 6.16: The table mapping

To go to the next step click on the Next button. In step 8 the selection of source and target servers, the databases, the tables and etc. that are to be migrated are summarized. Finally click on the Next button on the below shown screen to begin the migration. In step 8, the wizard looks as shown below (Figure 6.17 and Figure 6.18).

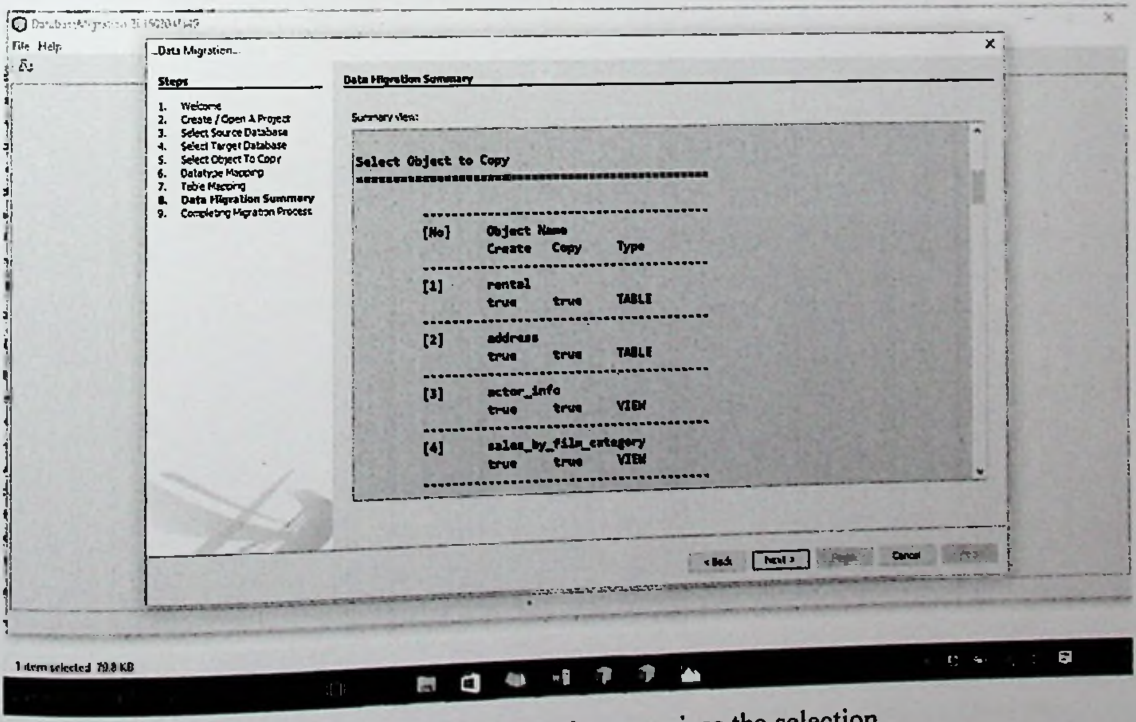

File Help
Data Migration
Steps
1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process
< Back
Next >
Cancel
1 item selected 79.8 KB
Search the web and Windows

Data Migration Summary

Summary View:

```
Project Details
=====================================================
Project Name:     test1
Project Path:     D:\MSc\Final Project\DBMigrationTool\DatabaseMigration

Source Database Details
=====================================================
Source Database:  MySQL
Host Name:        127.0.0.1
Port No: 3306
Username:         root
Database Name:    sakila

Target Database Details
=====================================================
Target Database:  Postgresql
Host Name:        127.0.0.1
Port No: 5432
Username:         postgres
Database Name:    postgres
```

Figure 6.17: Database migration summary



Data Migration Summary

Summary View:

```
Select Object to Copy
=====================================================

          ------------------------------------------
          [No]     Object Name
                   Create   Copy     Type
          ------------------------------------------
          [1]      rental
                   true     true     TABLE
          ------------------------------------------
          [2]      address
                   true     true     TABLE
          ------------------------------------------
          [3]      actor_info
                   true     true     VIEW
          ------------------------------------------
          [4]      sales_by_film_category
                   true     true     VIEW
          ------------------------------------------
```

Figure 6.18: The wizard summarizes the selection

By pressing the Next button on the above, shown screen the final wizard page is displayed. Then in order for the actual migration to take place the user needs to click on the Migrate button. The screen shows the progress of the migration through a log (Figure 6.19 and Figure 6.20).

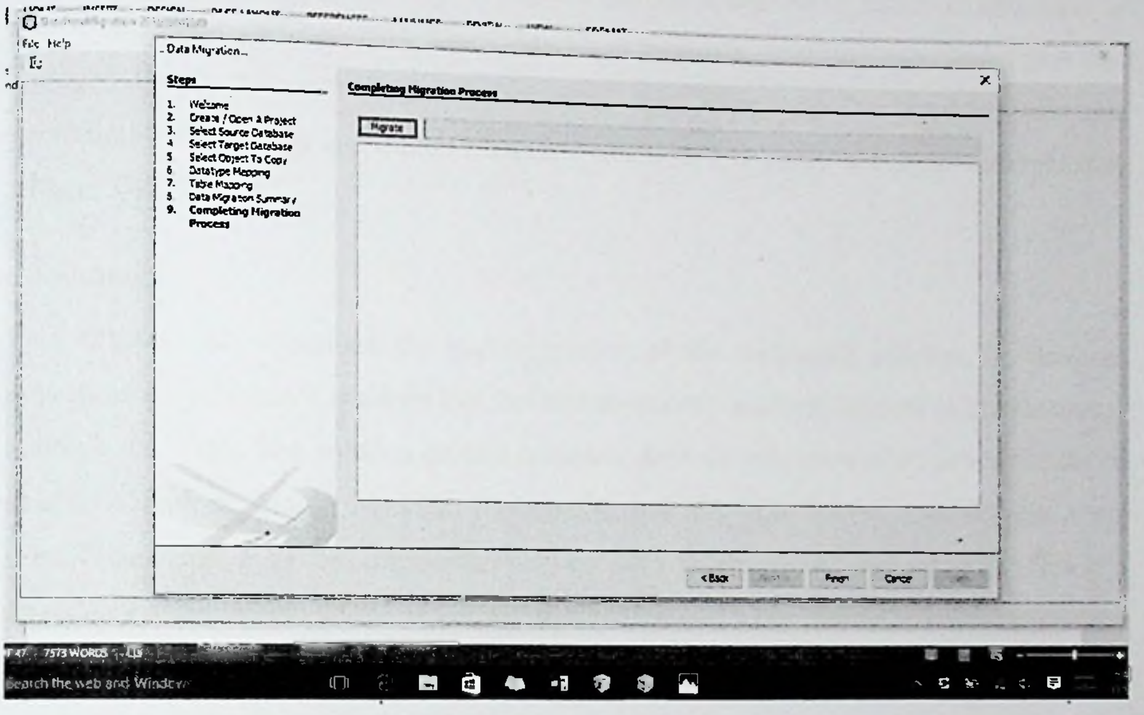

File Help
Data Migration
Steps
1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process
Completing Migration Process
Migrate
< Back
Finish
Cancel
7573 WORDS
Search the web and Windows

Figure 6.19: The last screen of the wizard is showed

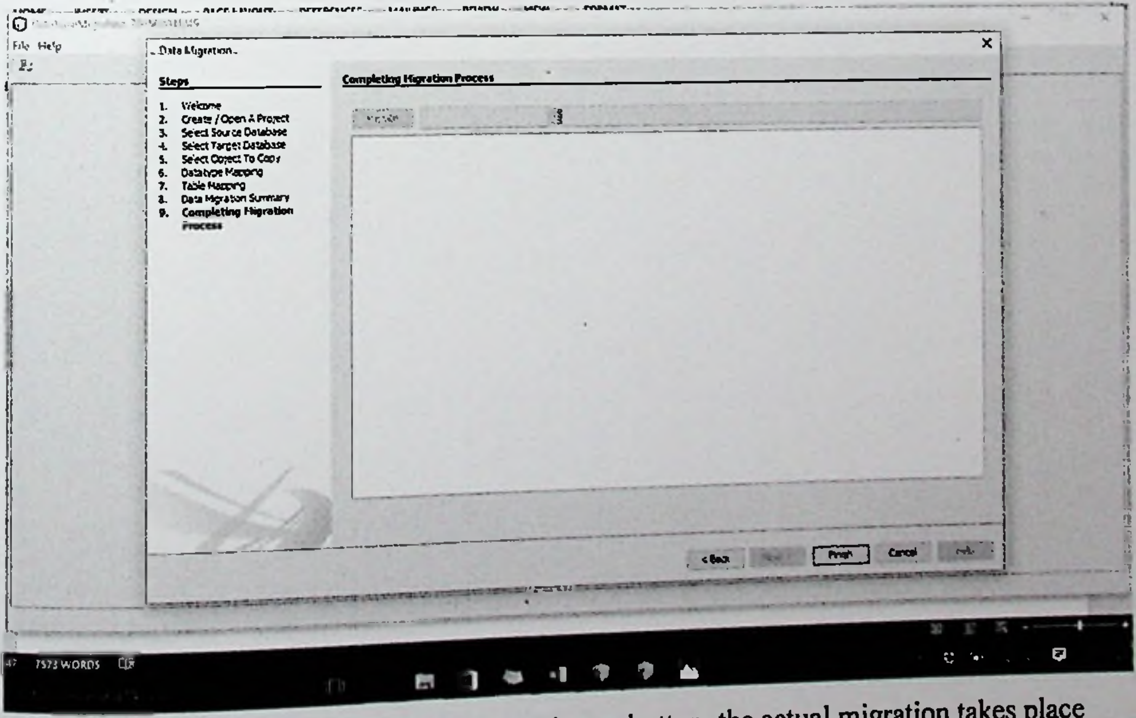

File Help
Data Migration
Steps
1. Welcome
2. Create / Open A Project
3. Select Source Database
4. Select Target Database
5. Select Object To Copy
6. Datatype Mapping
7. Table Mapping
8. Data Migration Summary
9. Completing Migration Process
Completing Migration Process
< Back
Finish
Cancel
7573 WORDS

Figure 6.20: When the user clicks the migrate button, the actual migration takes place

Now the user can connect to the target database server and checks to make sure all the objects are transferred correctly.

The application needs to dynamically locate information relating to result set structures or specific database configurations. This is known as metadata. There are two classes provided through JDBC that are used to deal with this. They are DatabaseMetaData and ResultSetMetaData. The application's backend mechanism is totally developed with metadata APIs in JDBC.

6.3 Summary

This chapter fully described the implementation of the automated solution for database migration. In this sense, it is shown that the implementation details of database migration across multiple databases. The solution gives a complete database migration of schema and data of databases such as Oracle, MySQL, PostgreSQL and MS SQL Server. It is flexible, user-friendly and does a smooth migration while ensuring reliability and data integrity. This is a significant improvement when comparing other products such that makes the migration process easy to do through a high quality user interface. Next chapter explains the evaluation of the product that we have already developed.

Chapter 7

Product Evaluation

7.1 Introduction

This chapter presents how the software solution can be tested with respect to different aspects such as functionality, reliability, usability, efficiency, maintainability and portability.
By definition testing involves the process of checking the system and its components against the requirements that gave reason for their implementation and ensuring that the requirements are satisfied applicably. Thereby by testing we can identify the inconsistencies and bug that are in the system while comparing the functionality with the requirements. There are different types of testing that can be used to test a software. In this application, we recommend to do automation testing which involves automation of a manual process. Automation can be defined as instance when the QA tester writes scripts and uses another software to test the application. Automation testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly. Therefore Squish Test Automation Suite [17] is used to test this application. Let's discuss that how we can use this automation testing tool to test the solution.

7.2 Squish Test Automation Suite

Squish is amongst the top range automated testing software in the market. It is cross-platform GUI that is used as a regression testing tool. This tool can test applications that use a variety of GUI technologies i.e. Java Swing. QA testers are able to use standard scripting languages such as JavaScript, Perl, Python, and Tcl to record and write tests. This is used widely and successfully around the world by many software testers of small and large organizations.

Testers have much access to the internals of the applications as Squish provides extremely tight integration with the specific GUI technologies it supports. This results in reliable and robust GUI tests. Squish communicates with the objects in the application, it records the users actions and then runs the written scripts while enabling objects to progress through the script as fast as application permits.

Squish then verifies that the scripts are producing the expected results and all junctions of the process. It performs both positive and negative testing, while validating each scripted test case. This is done by confirming that the data is visible to the user and verifying that related objects and properties give out the values that are expected or comparing such results visually.

7.3 Summary

This chapter fully described the evaluation method and the tool for database migration solution. As it is a difficult task to test the user interface of certain applications automating this task although challenging is very much beneficial. Next chapter explains the conclusion and the further work of the product.

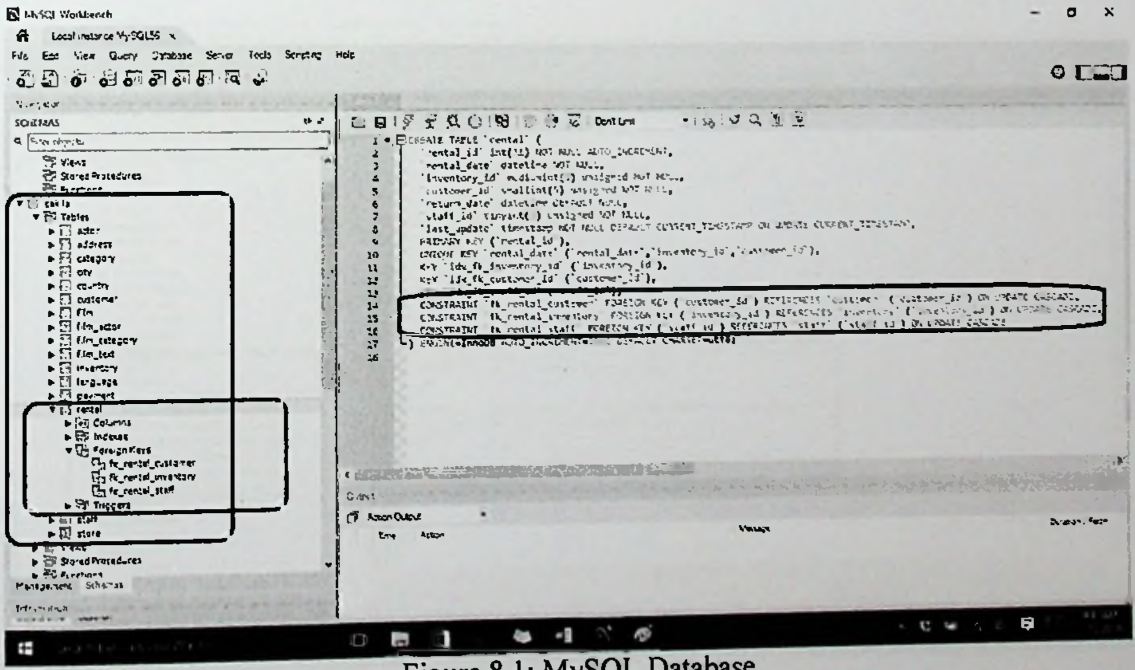
Conclusion

8.1 Introduction

This chapter illustrates the results which is generated from the solution and the further improvements can be done to the solution. As an example, by using this solution, we can simply migrate schema and data of a MySQL database to PostgreSQL or MS SQL Server or Oracle databases with the integrity constrains. The integrity constraints are mainly primary keys and foreign keys. The generated outputs are shown in the results section of this chapter. List of additional new features has been identified and these new features are listed in the further work section in this chapter.

8.2 Results

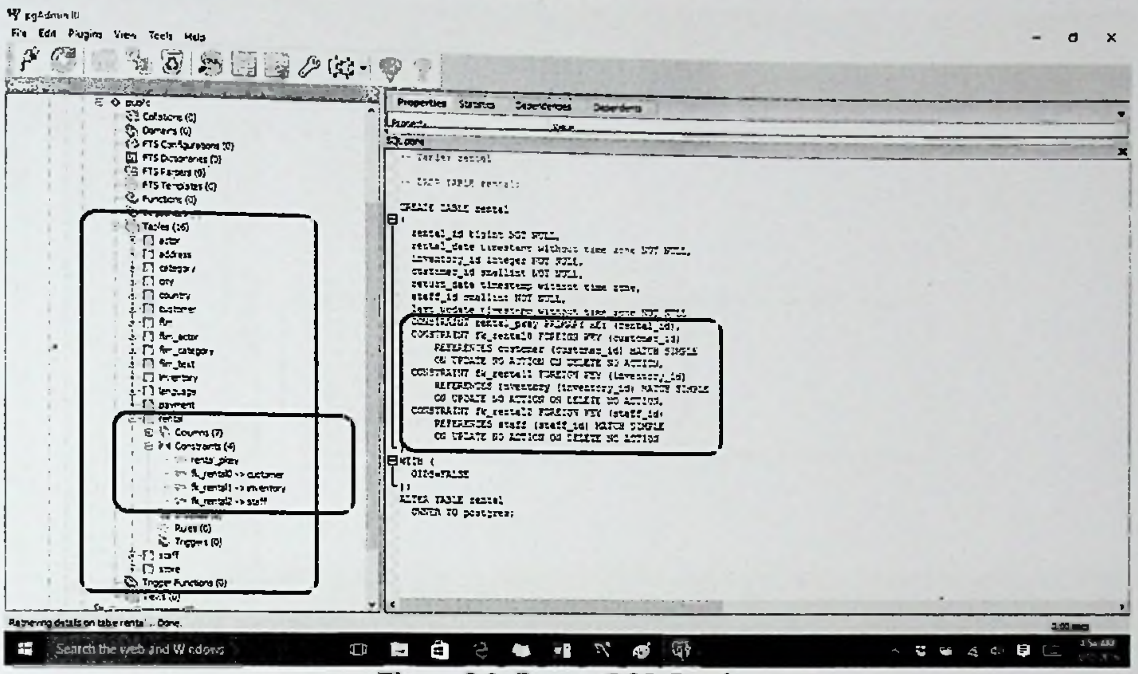
Figure 8.1 shows SAKILA database in MySQL and this database has 16 tables with data. As an example, Primary keys and foreign keys of rental table are also highlighted in the Figure 8.1. This database uses as an input for the application and it migrates from this MySQL database to PostgreSQL, MS SQL Server and Oracle databases.



```
CREATE TABLE `rental` (
  `rental_id` int(11) NOT NULL AUTO_INCREMENT,
  `rental_date` datetime NOT NULL,
  `inventory_id` mediumint(8) unsigned NOT NULL,
  `customer_id` smallint(5) unsigned NOT NULL,
  `return_date` datetime DEFAULT NULL,
  `staff_id` tinyint(3) unsigned NOT NULL,
  `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`rental_id`),
  UNIQUE KEY `rental_date` (`rental_date`,`inventory_id`,`customer_id`),
  KEY `idx_fk_inventory_id` (`inventory_id`),
  KEY `idx_fk_customer_id` (`customer_id`),
  [illegible]
  CONSTRAINT `fk_rental_customer` FOREIGN KEY ( `customer_id` ) REFERENCES `customer` ( `customer_id` ) ON UPDATE CASCADE,
  CONSTRAINT `fk_rental_inventory` FOREIGN KEY ( `inventory_id` ) REFERENCES `inventory` ( `inventory_id` ) ON UPDATE CASCADE,
  CONSTRAINT `fk_rental_staff` FOREIGN KEY ( `staff_id` ) REFERENCES `staff` ( `staff_id` ) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=[illegible] DEFAULT CHARSET=utf8;
```

Figure 8.1: MySQL Database

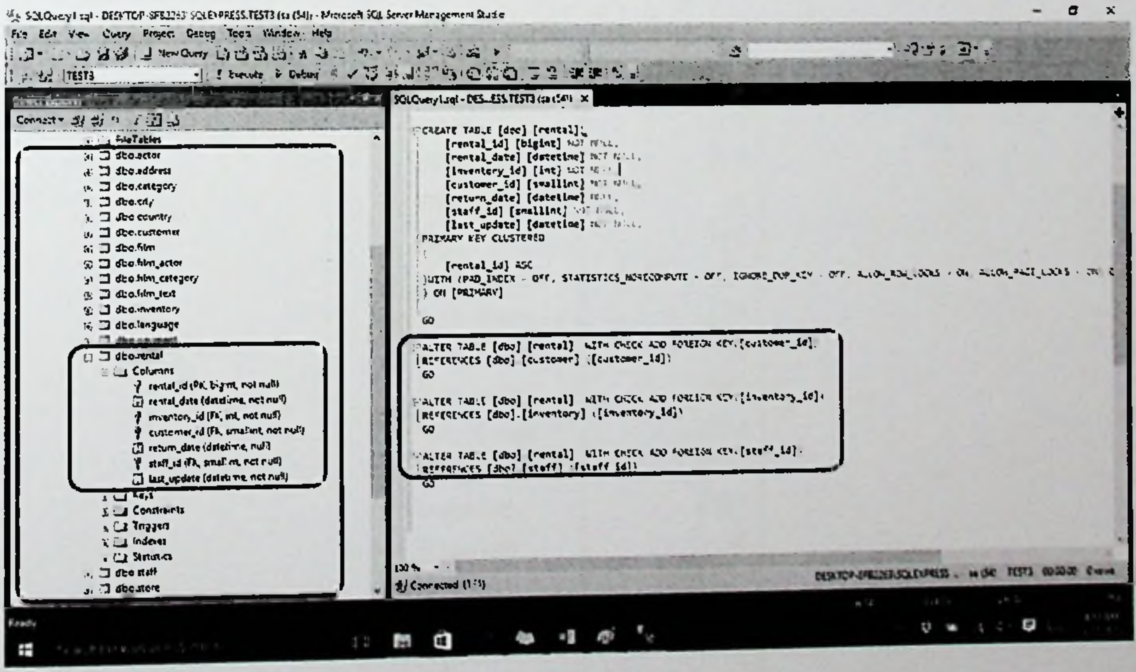
Figure 8.2 illustrates the migrated PostgreSQL database with all 16 tables and it also shows the rental table's integrity constraints.



```
-- Table: rental

-- DROP TABLE rental;

CREATE TABLE rental
(
  rental_id bigint NOT NULL,
  rental_date timestamp without time zone NOT NULL,
  inventory_id integer NOT NULL,
  customer_id smallint NOT NULL,
  return_date timestamp without time zone,
  staff_id smallint NOT NULL,
  last_update timestamp without time zone NOT NULL,
  CONSTRAINT rental_pkey PRIMARY KEY (rental_id),
  CONSTRAINT fk_rental0 FOREIGN KEY (customer_id)
      REFERENCES customer (customer_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_rental1 FOREIGN KEY (inventory_id)
      REFERENCES inventory (inventory_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_rental2 FOREIGN KEY (staff_id)
      REFERENCES staff (staff_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE rental
  OWNER TO postgres;
```

Figure 8.2: PostgreSQL Database

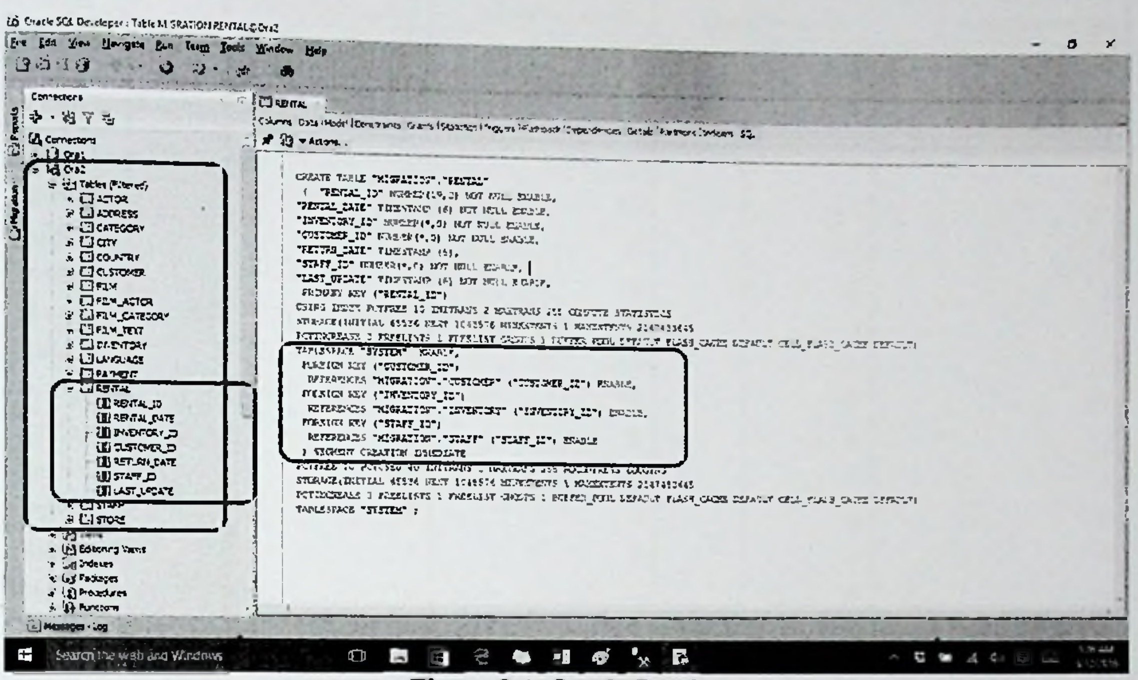
Figure 8.3 illustrates the migrated MS SQL Server database with all 16 tables and it also shows the rental table's integrity constraints.



```
CREATE TABLE [dbo].[rental](
	[rental_id] [bigint] NOT NULL,
	[rental_date] [datetime] NOT NULL,
	[inventory_id] [int] NOT NULL,
	[customer_id] [smallint] NOT NULL,
	[return_date] [datetime] NULL,
	[staff_id] [smallint] NOT NULL,
	[last_update] [datetime] NOT NULL,
PRIMARY KEY CLUSTERED
(
	[rental_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[rental]  WITH CHECK ADD FOREIGN KEY([customer_id])
REFERENCES [dbo].[customer] ([customer_id])
GO

ALTER TABLE [dbo].[rental]  WITH CHECK ADD FOREIGN KEY([inventory_id])
REFERENCES [dbo].[inventory] ([inventory_id])
GO

ALTER TABLE [dbo].[rental]  WITH CHECK ADD FOREIGN KEY([staff_id])
REFERENCES [dbo].[staff] ([staff_id])
GO
```

Figure 8.3: MS SQL Server Database

Figure 8.4 illustrates the migrated Oracle database.with all 16 tables and it also shows the rental table's integrity constraints.



```
CREATE TABLE "MIGRATION"."RENTAL"
 (  "RENTAL_ID" NUMBER(10,0) NOT NULL ENABLE,
"RENTAL_DATE" TIMESTAMP (6) NOT NULL ENABLE,
"INVENTORY_ID" NUMBER(*,0) NOT NULL ENABLE,
"CUSTOMER_ID" NUMBER(*,0) NOT NULL ENABLE,
"RETURN_DATE" TIMESTAMP (6),
"STAFF_ID" NUMBER(*,0) NOT NULL ENABLE,
"LAST_UPDATE" TIMESTAMP (6) NOT NULL ENABLE,
 PRIMARY KEY ("RENTAL_ID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"  ENABLE,
 FOREIGN KEY ("CUSTOMER_ID")
  REFERENCES "MIGRATION"."CUSTOMER" ("CUSTOMER_ID") ENABLE,
 FOREIGN KEY ("INVENTORY_ID")
  REFERENCES "MIGRATION"."INVENTORY" ("INVENTORY_ID") ENABLE,
 FOREIGN KEY ("STAFF_ID")
  REFERENCES "MIGRATION"."STAFF" ("STAFF_ID") ENABLE
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM" ;
```

Figure 8.4: Oracle Database

Table 8.1 lists row count of each table in SAKILA database in MySQL.
SELECT table_name, TABLE_ROWS FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'sakila';

Table Name	Row Count
actor	200
address	603
category	16
city	600
country	109
customer	599
film	1000
film_actor	5462
film_category	1000
film_text	1000
inventory	4581
language	6
payment	1000
rental	999
staff	2
store	2

Table 8.1: Row count of each table

Tables	MySQL to PostgreSQL Time(Seconds)	PostgreSQL to MS SQL Server Time(Seconds)	MS SQL Server to Oracle Time(Seconds)	Oracle to MySQL Time(Seconds)
film_category	3	3	3	3
country	0	0	0	0
address	4	4	5	4
city	2	2	2	2
film_actor	16	16	17	17
staff	0	0	0	0
language	0	0	0	1
store	0	0	0	1
film	9	10	11	12
inventory	17	16	17	17
rental	5	6	6	7
actor	1	1	1	1
film_text	3	5	4	6
payment	4	5	6	5
category	0	0	0	0
customer	3	4	4	4
	67	72	76	80

Table 8.2: Time taken to migrate data for each table with respect to different databases

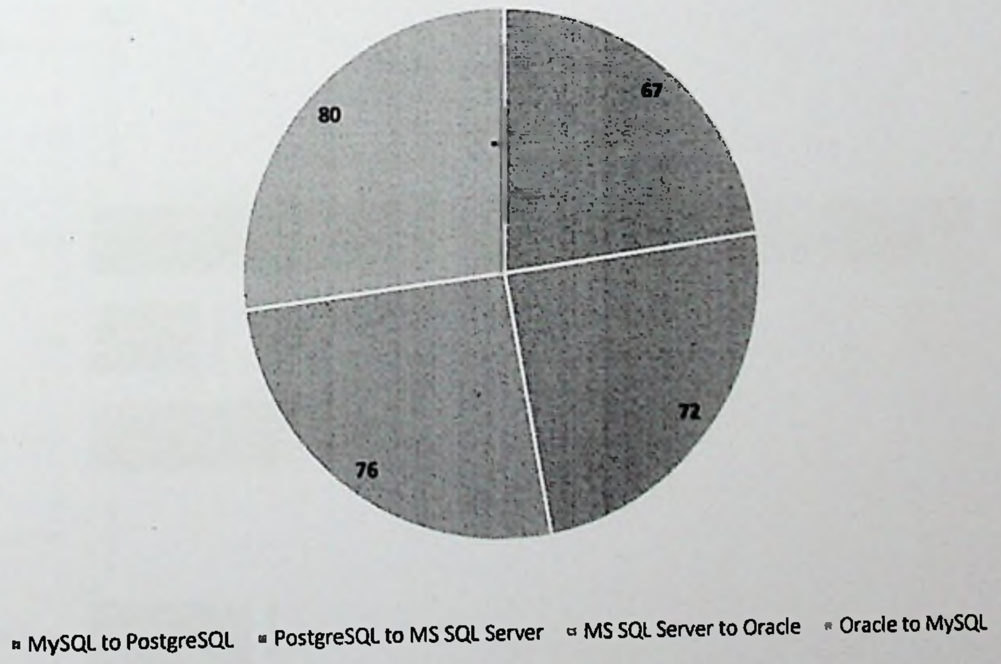


Figure 8.5: Total time (Seconds) taken to migrate data

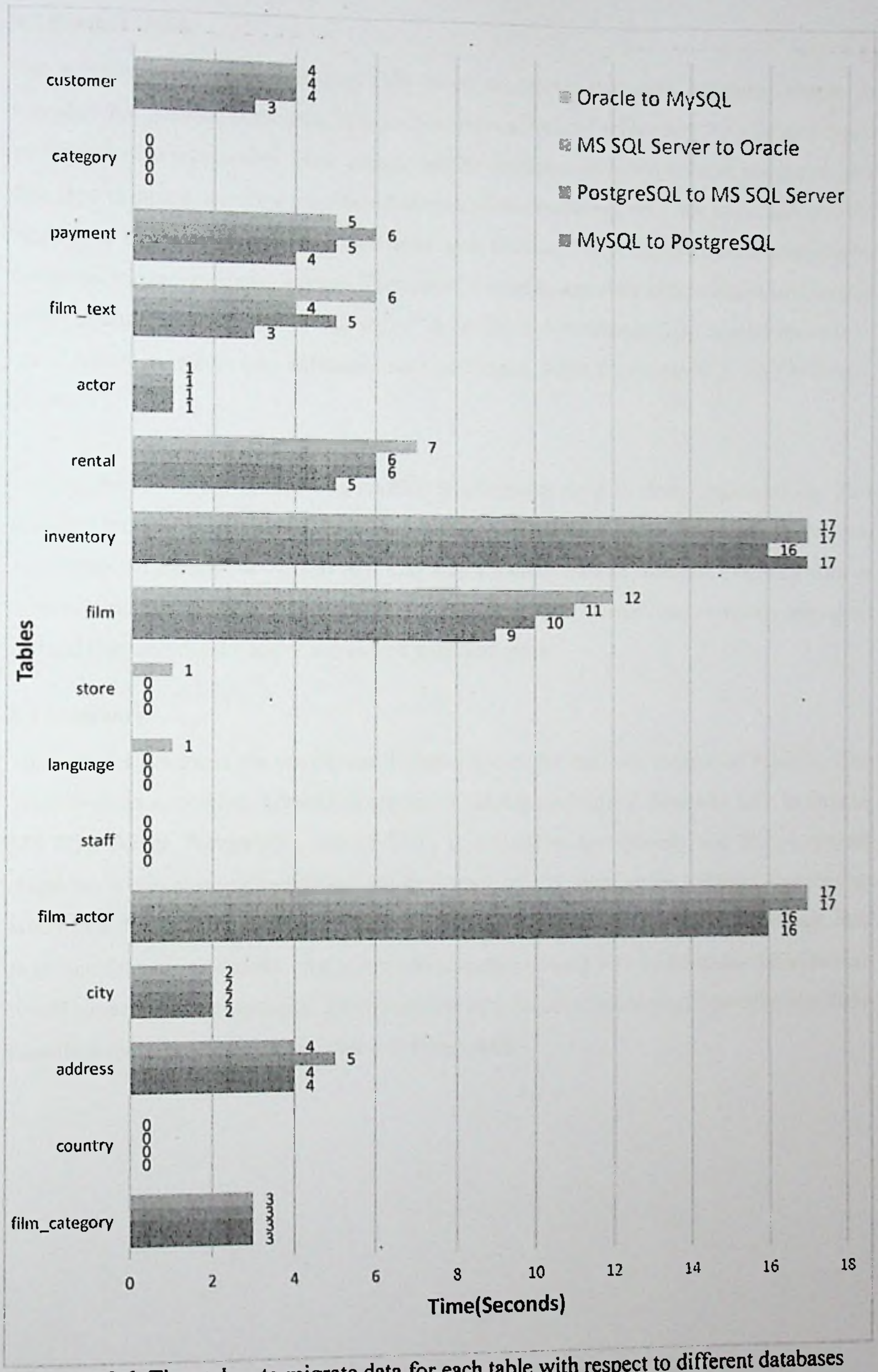


Figure 8.6: Time taken to migrate data for each table with respect to different databases

8.3 Further Work

The solution will migrate and copy data based on criteria that could be either simple or complex. For instance if the criteria is set for tables all related tables with be migrated based on the specified relationship. New options will be included, table and column name changes, data type mapping, selecting specific columns, column mappings etc., can be included while migration. The solution will be added more new features like migrating stored procedures, functions, triggers and table indexes. There will be another important option which will backup tables in target database before migration. The solution will migrate SQL queries specific to one database language into databases such as Oracle, MySQL, PostgreSQL and MS SQL Server.

Moving data from one database to another is a frequent need in many organizations. This includes moving departmental data into the central database and development data into production. In addition to transfer new data into the table without disrupting existing data or creating loss of data. This software thereby offers a straight forward and complete migration method that includes the above mentioned tasks and more.

8.4 Summary

This chapter described the results and future work of the database migration solution. The solution gives a complete database migration of schema and data of databases such as Oracle, MS SQL Server, PostgreSQL, and MySQL. It is flexible, user-friendly and does a smooth migration while ensuring reliability and data integrity. In addition the software includes an automated database migration tool which handles complex schema and organizational data from one database to another. The automation handles at least 90% of the tasks that otherwise would have to be done manually. There are more new features that we have identified and these identified features will be implemented as future work.

Chapter 9

Reference

[1] "Database Administration: The Complete Guide to DBA Practices and Procedures," 2002. http://ptgmedia.pearsoncmg.com/images/9780321822949/samplepages/0321822943.pdf.

[2] "Kimball & Caserta -The Data Warehouse ETL Toolkit," 2004. http://users.itk.ppke.hu/~szoer/DW/Kimball%20&%20Caserta%20-The%20Data%20Warehouse%20ETL%20Toolkit%20%5BWiley%202004%5D.pdf.

[3] Brown, Carol, and Heikki Topi. "IS Management Handbook, 8th Edition," 2003. https://books.google.lk/books?id=k_eE4Oa7yAoC&pg=PA925&lpg=PA925&dq=is+management+handbook+eighth+edition+pdf&source=bl&ots=Yvo1A_LuHx&sig=9T161doW0hDao7rKl6eW3J7YnI4&hl=en&sa=X&redir_esc=y#v=onepage&q=is%20management%20handbook%20eighth%20edition%20pdf&f=false.

[4] "Data Migration | No Software, No Tool | MuleSoft," 2016. https://www.mulesoft.com/resources/esb/data-migration-solution.

[5] Active Database Software. "FlySpeed DB Migrate," 2015. http://www.activedbsoft.com/overview-migrate.html.

[6] EasyFrom. "ESF Database Migration Toolkit," 2016. https://www.easyfrom.net/.

[7] Zoho Corporation. "SwisSQL Data Migration Tool," 2012. http://www.swissql.com/products/datamigration/data-migration.html?ad-main1.

[8] Oracle Corporation - MySQL. "MySQL Workbench: Database Migration," 2016. https://www.mysql.com/products/workbench/migrate/.

[9] Microsoft. "SQL Server Migration Assistant," 2015. http://blogs.msdn.com/b/ssma/.

[10] Oracle SQL Developer. "Oracle SQL Developer," 2016. http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html.

[11] "NetBeans IDE - NetBeans Rich-Client Platform Development (RCP)," 2016. https://netbeans.org/features/platform/.

[12] "MySQL Workbench," 2016. https://www.mysql.com/products/workbench/.

[13] "pgAdmin: PostgreSQL Administration and Management Tools," 2016. http://www.pgadmin.org/index.php.

[14] "Use SQL Server Management Studio," 2016. https://msdn.microsoft.com/en-us/library/ms174173.aspx.

[15] "JDBC Overview," 2016. http://www.oracle.com/technetwork/java/overview-141217.html.

[16] Bcok, Heiko. *The Definitive Guide to NetBeansTM Platform 7*. Apress, 2012. http://www.apress.com/9781430241010.

[17] "Froglogic - Automated Cross-Platform GUI Testing," 2016. http://www.froglogic.com/index.php.