International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 05 | May -2017 www.irjet.net p-ISSN: 2395-0072

Web Based Data Migration Tool

Rajeshwari GS[1], Prof. Harish G[2] , Miss.Smitha Shekar[3]

[1] *M. Tech Student, Department of Computer Science & Engineering, Dr. Ambedkar institute of Technology, Bangalore*

[2] *Associate Professor, Department of Computer Science & Engineering, Dr. Ambedkar institute of Technology, Bangalore*

Abstract - *Data migration is an important activity in every organization, this paper focus on data flow process between one or more databases. To enable data migration there is a need to transfer data between systems easily and efficiently. In this paper we propose an effective approach of data transfer from MySQL to PostgreSQL. Our work notifies the usage of model-view-controller architecture along with rails framework. We have experimented with use cases which will prove a better performance in migration and provides accuracy.*

Key Words**: Data Migration, Manual Migration, Automated Migration, Models, Databases, Retrieve, Recreate**

1. INTRODUCTION

The process of moving data between computer storage systems such as files and databases is called as Data migration. To ensure effective data flow, data present in old system needs to be mapped to data on newer system. This process involves data Extraction that is reading data from source system and data loading that is data is written to destination system. During the process, data is verified and validated to ensure accuracy [1].

This paper reflects An User Interface (UI) based data flow application which migrates/copies/syncs data from one system to another irrespective of Structured Query Language (SQL) databases types used. Here UI is provided with unique selection options for every database engines [6]. UI representation of each database allows selection of models. Model and database names are dynamically generated using controller actions. Source (model where in its input port is under usage) and destination models (model where in its output port is under usage) are created followed by storing data as template in selected database, retrieving and recreating whole template. Template is in the form of JSON (JavaScript Object Notation).

Generation of Uniform Resource Locator (URL) is carried out and recreation of template is achieved. Business logic is part of system which determines how data is calculated and routed during the workflow [2]. It involves migration across models with same or different data type, row wise and column wise flow also selection of column data from one table and combine it in column of different table. A number of prototypes and tools have been developed to facilitate migration of relational databases into target databases. Some of them are discussed in next section.

2. LITERATURE SURVEY

Migration tools involve MySQL migration toolkit, Progression DB, open DB copy .Following are some of the data flow tools come across during literature survey.

2.1 Host-based file-level migration

This is an open source tool has been around for a long time and distinguishes itself by being very simple, yet powerful, and totally host- and storage-agnostic. It's very flexible and can be adapted to almost every data migration need, but it shines especially brightly with largely static unstructured content [2].

2.2 Host-based block-level migration

With large structured files like databases, block-level migration tools make the most sense. Host-based volume managers are often overlooked as a data migration

tool, yet they provide a powerful way to non-disruptively migrate data from one storage array to another [3]. Most operating systems already have a capable volume manager that is heterogeneous and already installed.

2.3 DB export and DB import utility

It Imports or exports a database to a text file that is stored on disk or tape, it can modify the database schema and change the data format, it can move data between operating systems

Drawback of this system is it has got faster performance than the DB load utility, but slower performance than the DB load utility also it Moves the entire database [7].

2.4 DB load utility

Transfers data from one or more text files into one or more existing tables, it can modify database schema, it can move data between operating systems, it is moderately easy to use and also it will import data from non-Informix sources [8]. Drawback of this system is it has got slower performance than the DB export, DB import, and on load utilities.

2.5 On unload and On load utilities

It Unloads data from a database into a file on tape or disk; loads data, which was created with the on unload command, into the database server, it has got Fast performance Optional logging [9]

Disadvantages associated with this tool are:

- It only moves data between database servers of the same version on the same operating system
- Cannot modify the database schema
- Logging must be turned off
- Difficult to use

2.6 HPL

HPL stands for High performance loader, this tool Loads data from any ASCII or COBOL file that meets certain format requirements, For extremely large databases, it has a performance advantage over other IBM Informix data-migration utilities, because it performs I/O and code-set conversions in parallel and it Can modify database schema, it Can move data between operating systems also it Can import data from non-Informix sources [10]. Disadvantage with this is it requires significant preparation time

System architecture for data migration tool discussed in this paper is presented in next section.

3. SYSTEM ARCHITECTURE

The System architecture consists of layers such as User interface, Middle layer, Database layer.

Schematic representation is system architecture is described below.

- **User interface:** The user interface (UI), in the industrial design field of human-machine interaction, is the space where interactions between humans and machines occur. A web client allows the user request data on the server, and shows the user the result of the request.

  Generally, the goal of UI design is to produce a user interface which makes it easy (self-explanatory) and efficient tool to operate a machine in the way which produces desired result. Front-end technologies such as HTML5, CSS3, JavaScript, jQuery, Semantic-UI and JointJS have been used to develop UI.

- **Middle Layer:** This layer is also referred as business layer, the modular representation as shown in figure-1 reveals that modules such as (Database) DB mapper, it is an Object Relational

Mapper written in ruby. Table mapper, the goal is to map database tables for data migration. Table conversion, is to convert data type of table data to change from one data type to another data type. Verification, let developers to verify that if target database exists or not. Validation, it validates target database correctness. CRUD module for create, read, update, delete operation. Notification, it is meant to notify e-mail service included that is it notifies whether email sent/received. Authentication and authorization, is to provide security for logged users [4].

- **Database Layer:** Databases are present for storing tables/data at the backend. Multiple databases such as SQLite, MySQL and pgSQL (PostgreSQL) are used. It manages storing data as template in module, retrieving and re-creation of template. It supports row-wise and column-wise data migration

Other layers Includes OAuth2 and API-Gateway, which are described below.

- **oAuth2:** It is an **open** standard for **authorization**, commonly used as a way for internet users to authorize websites or applications.
- **API-Gateway**: It provides simple API gateway to complex subsystems, it decouples interface the clients see from underlying API implementation. API-Gateway acts as API front-end, receives API-requests and process it and they also operates with third party APIs. Entire flow of data migration over databases is mentioned in next section.

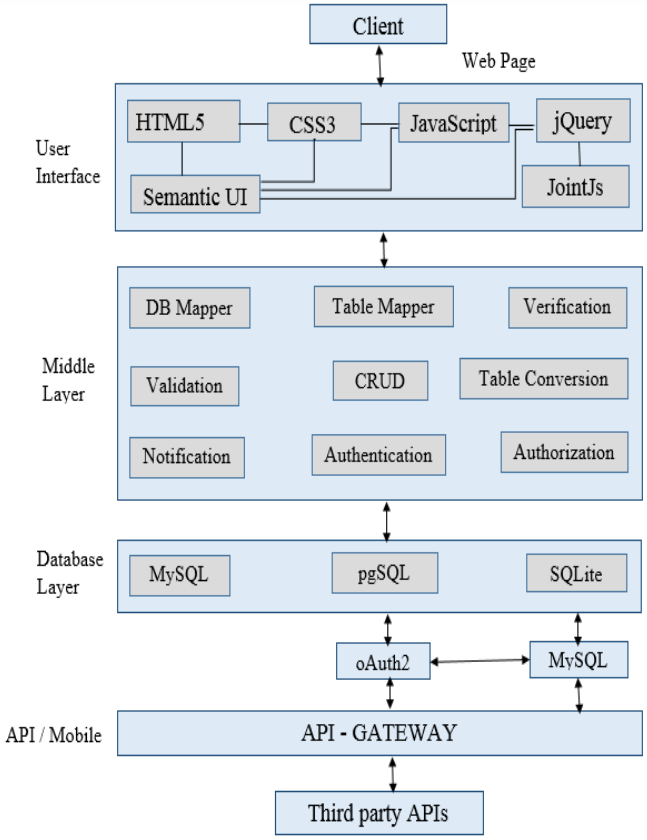

Figure-1: System Architecture

4. Data Migration on Databases

In this paper, we focused on using Ubuntu 14.10 LTS as operating system, front end languages like HTML5, CSS3, JointJS, Semantic UI, jQuery, JavaScript, data bases like MySQL and PostgreSQL, Framework is Rails, Backend language is Ruby 1.2. Other requirements include Docker image with pre-installed components also IBM Bluemix Cloud environment for deployment of processed data.

Data migration tool depicted over here is web application tool for migrating data from one database engine to another database engine. This tool is based on Model View Controller (MVC) Architecture and built with Ruby on rails framework. Model is the layer of system responsible for business data and logic. View is to represent data on UI Controller actions meant for responding to user input and performing interactions on data model objects.

Models/tables present in database engines are stored as templates. Those models can be retrieved and recreated over user interface [5]. Linking of elements involves selecting appropriate models. Later Data is migrated/moved/copied from source to destination. For example, Migration across models present within single database (within MySQL and within pgSQL) as well as multiple database (From MySQL to pgSQL and vice versa). Use cases regarding migration includes both manual and automated incrementation of datatype, sequential migration, fetching table data over sidebar with toggle effect. Table level and table column level synchronization is achieved. Table level synchronization involves selection of all rows and columns present in table followed by inserting entire selected table to target model/table. Column level synchronization consists of selection of only specific columns then inserting the same to target model/table.

In order to access stored template for recreating data on UI, URL is provided. Pagination is implemented to allow specified row list. Proposed methodology of this tool is presented in next section.

5. Proposed SQL independent migration strategy

The Proposed strategy for "Data Migration" ,consists of three systems and schematic representation is shows as below Figure-2.

The methodology consists of three systems. First system contains the database tables that uses MySQL database engine, second system contains database tables that uses pgSQL database engine. In both the systems many tables are created as per their respective engines specifications. A third system is the one where user interface is designed. It involves UI representation of system1 and system2 and then mapping the data flow as required. Here the system is capable of syncing two systems. Mechanism of this migration strategy is explained in upcoming section.

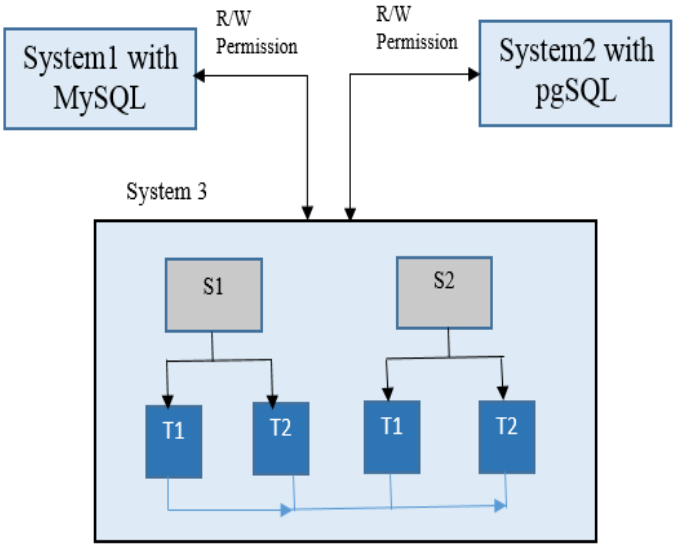

Figure-2: Proposed methodology

5.1 Dataflow mechanism

The mechanism of migrating of data is defined in the flowchart represented in Figure-3. UI is having representation of database tables and models names generated dynamically through back-end via Controller, they are segregated according to each type of database engine. User will select database and table names and each model over UI is having input port and output port. Implementation is done in such way that output port can only be linked to input port. Any number of models can be used for migration. Before migration, Data is to be verified and validated to ensure accuracy. UI implementation allows verification that is user can check entire data of selected model before moving. User can store selected models in database by choosing STORE option. For every stored template, URL is generated to fetch stored data incase if user needs. URL needs to be encrypted for security purpose. Overview of this proposed method along with manual and automated migration strategies are depicted in next section.

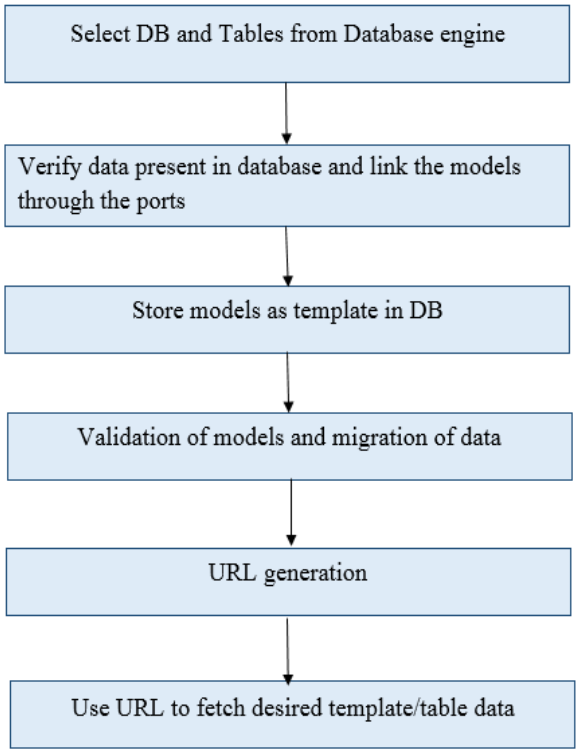

Figure-3: Flow chart

6. Overview of Proposed method

Proposed method of migration involves selecting tables from both type of databases then moving table data, which provides row wise and column wise synchronization. This type of migration can be achieved both manually and automatically.

6.1. Stage 1: Manual method of migration

Manual method involves selecting models/tables from respective database engines and producing it on UI. Major task involved in data migration is proper linking of selected models. Linking involves determining source and target models followed by verifying data present in each model. Data migration is done after verifying every table data, it consists of row-wise (migration of specific row) and column wise (migration of specific column) migration, implementation of use cases such as changing length of datatype manually, verifying backend data over UI, migration from MySQL to pgSQL database and pgSQL to MySQL database, migration across multiple tables. URL generated will be authenticated and shared among users. Manual method is depicted in Figure-4.

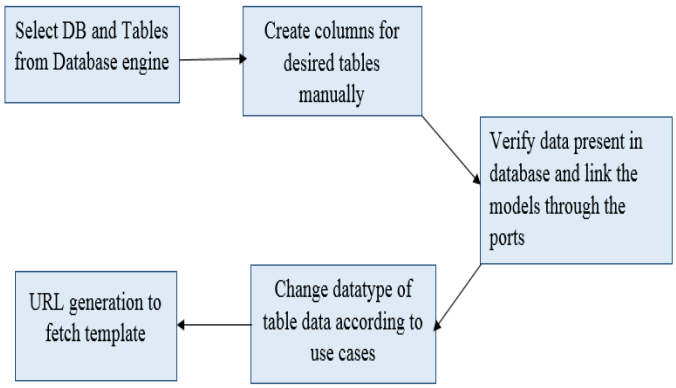

Figure-4: Manual method

6.2. Stage 2: Automated method of Migration

Automated method of migration of data involves data mapping which depicts relation between source and target, linking models, migration and generation of encrypted URL. Automated Migrations allows easier data flow over platform. HTTPS establishes SSL (secure socket layer) before http data is transferred. This ensures that all URL data is carried within encrypted connection and is protected from any attacks. This also provides security such that no data is transferred before secure connection is established. Encryption is achieved using secrete key shared between user and developer. Figure-5 defines automation and Results follows next section.

Figure-5: Automated data migration

7. CONCLUSIONS

In this paper, we have proposed an effective approach to do Data Migration across two different types of database engines. We have used MVC framework to develop a migration tool. Here Rails framework has been used as full stack framework to overcome some of the bottlenecks of relational database. Using Data flow tool it is easier to migrate the databases of one engine to another engine with minimal efforts. Things such as Migrating a single table, renaming the table dynamically and selective data-flow are all achieved easily. Offline data transfer is initiated when first time a rule is defined. Once a rule is defined it can be stored and then can act as online data sync tool. All these operations were really difficult to do it manually and require lot of efforts. Dynamic data exposure from source to destination DB is achieved and this is important to integrate a third party Systems with full control being in our side.

Future work concerns extension of the proposed methodology to a next level, where in Hybrid approach is preferred. It includes migrating data from cloud to local machine and then migration to be done from local machine to cloud. Then Moving entire application to cloud with the usage of deployment models. Extension of this proposed tool as SAAS (software as a service) model.

REFERENCES

[1] Philip Howard, Carl potter, Data migration in global 2000, research, forecasts and survey results – a survey paper by Bloor research: publication date: 2007

[2]Syed Ahmed Karim and VVS Raghaveendra, data migration using iterative methodology, proceedings of the 24th IASTED international; multi conference software engineering February 14-16,2006, Innsbruck, Austria

[3] Data transformation and migration in polystores, Adam Dziedzic and Aron j. Elmore, Department of CSE, the University of Chicago

[4] Data migration helper using Domain Information, Irfan Kamil, M.M Inggriani, School of electrical engineering and informatics

[5] PMT: A PROCEDURAL MIGRATION TOOL, Mengying Zhang, shanghai jiao Tong University, shanghai, China

[6] An Enhanced Extract-Transform-Load System For migration of Data in telecom billing, Himanshu, agraval, IBM India research Laboratory, New Delhi, India.

[7]https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_031.html

[8]https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_031.htm

[9]https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_031.htm

[10]https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_031.htm