A Project Report

On

"NoSQL to RDBMS Converter"

submitted for partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

in

Computer Science

Submitted by

Saumya Singh (2000290120138)

Vageesha Rai (2000290120181)

Vikas Kumar Verma (2000290120189)

Under supervision of

Prof. Abhishek Goyal

KIET Group of Institutions, Ghaziabad

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

May 2024

DECLARATION

I/We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date : 11-03-2024

Saumya Singh	Vageesha Rai	Vikas Kumar Verma
2000290120138	2000290120181	2000290120189

CERTIFICATE

This is to certify that Project Report entitled **"NoSQL to RDBMS Converter"** which is submitted by **Saumya Singh, Vageesha Rai, Vikas Kumar Verma** in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Date: 11-03-2024

Supervisor:

Mr. Abhishek Goyal

Assistant Professor

Department of Computer Science

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe a special debt of gratitude to Prof. Abhishek Goyal, Department of Computer Science, KIET, Ghaziabad, for his constant support and guidance throughout our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also acknowledge the contribution of Dr. Ajay Kumar Shrivastava, Head of the Department of Computer Science, KIET, Ghaziabad, for his full support and assistance during the project's development. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

Last but not least, we acknowledge our friends for their contribution to the completion of the project.

Date : 11-03-2024

Saumya Singh	Vageesha Rai	Vikas Kumar Verma
2000290120138	2000290120181	2000290120189

ABSTRACT

NoSQL databases are becoming more and more popular for managing unstructured data for applications because they provide scalability and performance guarantees. However, many organizations choose to migrate data from an operational NoSQL database to a relational database based on SQL in order to utilize the current tools for business intelligence, analytics, decision-making, and reporting. The existing methods for transforming NoSQL databases into relational databases require manual schema mapping, which is laborious and needs subject expertise. An efficient and automatic process is needed to transform an unstructured NoSQL database into a structured database. We proposed and evaluated a practical method in this study for automatically transforming a NoSQL database into a relational database. In our experimental evaluation, we used a variety of NoSQL files, such as ".CSV," ".JSON," and ".XML" files, together with MySQL as a relational database to perform transformation operations for different dataset sizes. As transferring data from a NoSQL database to a relational database, we observed exceptional performance as compared to the state-of-the-art approaches used at the time. Our proposed system can be best described as an ETL (Extraction, Transformation, Loading) tool that basically loads the data from NoSQL files to relational databases. It contains two major elements, Schema analyzer and dynamic data mapper. The schema analyzer analyzes the input file schema thoroughly and then decides and defines the schema of it in the relational database. After this process is complete, the next step is data mapping where the data from the file is mapped with the table in the relational database. By mapping the data from these NoSQL files into relational databases, the user can query the data and extract essential insights from it with Structured Query language. Using relational databases, we can run SQL queries over the data which is mapped from these files. So, our model provides better access and more usability to these files and the crucial information stored in them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SQL	Structured Query Language
NoSQL	Not Only SQL
CSV	Structured Query Language
JSON	JavaScript Object Notation
XML	Extended Markup Language
DB	Database
DS	Dataset
ER	Entity Relationship Diagram
DFD	Data Flow Diagram
RDBMS	Relational Database Management system

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

NoSQL databases are becoming more and more popular for managing unstructured data for applications because they provide scalability and performance guarantees. However, many organizations choose to migrate data from an operational NoSQL database to a relational database based on SQL in order to utilize the current tools for business intelligence, analytics, decision-making, and reporting. The existing methods for transforming NoSQL databases into relational databases require manual schema mapping, which is laborious and needs subject expertise. An efficient and automatic process is needed to transform an unstructured NoSQL database into a structured database. We proposed and evaluated a practical method in this study for automatically transforming a NoSQL database into a relational database. In our experimental evaluation, we used a variety of NoSQL files, such as ".CSV," ".JSON," and ".XML" files, together with MySQL as a relational database to perform transformation operations for different dataset sizes. As transferring data from a NoSQL database to a relational database, we observed exceptional performance as compared to the state-of-the-art approaches used at the time. Our proposed system can be best described as an ETL (Extraction, Transformation, Loading) tool that basically loads the data from NoSQL files to relational databases. It contains two major elements, Schema analyzer and dynamic data mapper. The schema analyzer analyzes the input file schema thoroughly and then decides and defines the schema of it in the relational database. After this process is complete, the next step is data mapping where the data from the file is mapped with the table in the relational database. By mapping the data from these NoSQL files into relational databases, the user can query the data and extract essential insights from it with Structured Query language. Using relational databases, we can run SQL queries over the data which is mapped from these files. So, our model provides better access and more usability to these files and the crucial information stored in them.

1.2 PROJECT CATEGORY:

Creating a project that involves converting NoSQL data to a Relational Database Management System (RDBMS) format can be quite comprehensive and technically challenging. This project category falls under the domain of SYSTEM DEVELOPMENT and RESEARCH BASED. System Development and Research Based are the primary categories because they encapsulate the core objectives and challenges of developing a NoSQL to RDBMS converter. These categories focus on the technical and innovative aspects of building a solution that facilitates data migration across fundamentally different database systems. The main goal of this project would be to develop a tool or system that efficiently migrates data stored in a NoSQL FILES (like MongoDB, Cassandra, or DynamoDB) into a relational database (like MySQL, PostgreSQL, or Oracle).

System Development:

This category involves creating and implementing software solutions to solve specific problems or fulfill identified needs. A NoSQL to RDBMS converter is essentially a software tool designed to bridge the gap between different types of database systems, making it a clear fit for system development. It involves software engineering principles, including requirements analysis, design, coding, testing, and deployment.

Research Based:

The project requires a deep understanding of both NoSQL and relational database models, including their underlying principles, data structures, query languages, and performance characteristics. It may involve exploring new methods for data conversion, transformation, and mapping that haven't been standardized. This exploration and innovation align with research-oriented work, especially if the project aims to address unsolved challenges in data migration or improve upon existing methodologies.

1.3 OBJECTIVES:

The objective of a NoSQL to RDBMS converter project centers on creating a robust, efficient, and user-friendly tool or system that facilitates the migration of data from NoSQL files to relational database management systems (RDBMS). The key goals of this project include:

1. Data Compatibility and Integration

To enable seamless migration of data structures, entities, and relationships from NoSQL databases to an RDBMS format, ensuring that all data is accurately translated and preserved in the process.

2. Schema Translation

To automatically translate or assist in translating NoSQL schema designs to relational schema models, considering the differences in data modeling paradigms between NoSQL and RDBMS.

3. Data Transformation and Mapping

To convert and map NoSQL data types and structures (such as documents, key-value pairs, or wide-column stores) to the relational model, including tables, rows, and columns, while maintaining data integrity and relationships.

4. Performance and Scalability

To provide an efficient migration process that minimizes the impact on system performance, allowing for the migration of large datasets with minimal downtime.

5. Usability and Accessibility:

To offer a user-friendly interface that allows users with varying levels of technical expertise to configure and execute data migrations, potentially including features for custom mapping, conflict resolution, and preview of changes.

6. Interoperability:

To support a wide range of NoSQL and RDBMS platforms, maximizing the tool's applicability across different technological environments and use cases.

7. Data Integrity and Security

To ensure that all migrated data maintains its integrity, with comprehensive support for data validation, error handling, and security measures to protect sensitive information during the migration process.

8. Research and Innovation

To explore new methodologies and technologies for data migration between heterogeneous database systems, contributing to the academic and practical knowledge base in database management, data engineering, and software development.

By achieving these objectives, the project aims to provide a solution that addresses the challenges of migrating from NoSQL to RDBMS, facilitating the transition for organizations seeking the advantages of relational databases, such as robust transaction support, complex query capabilities, and standardized data integrity mechanisms.

1.4 PROBLEM FORMULATION:

Problem Statement

In the contemporary data management landscape, NoSQL databases have become popular due to their flexibility, scalability, and performance in handling large volumes of unstructured or semi-structured data. However, organizations often face the need to migrate their data to a Relational Database Management System (RDBMS) to leverage advantages such as strong consistency, ACID transactions, and complex query capabilities. This migration process presents several challenges due to fundamental differences in data models, schema design, and data integrity constraints between NoSQL and RDBMS systems.

Key Challenges

Schema Translation: NoSQL databases often employ flexible, schema-less models, whereas RDBMS requires a predefined schema. Mapping complex, nested structures of NoSQL databases to the tabular format of relational databases can be challenging.

Data Integrity and Relationships: Ensuring data integrity and accurately representing relationships between data entities during the migration process, including the translation of embedded documents or key-value pairs to foreign key relationships in RDBMS.

Data Type and Format Conversion: Converting various data types and formats from NoSQL to equivalents in RDBMS while preserving data accuracy and consistency.

Performance and Scalability: Managing the migration of potentially massive datasets with minimal impact on the performance of both the source NoSQL database and the target RDBMS.

Tool Usability and Flexibility: Providing a user-friendly, flexible tool that can accommodate various NoSQL and RDBMS platforms, enabling users to configure and execute the migration process according to their specific requirements.

Security and Privacy: Ensuring the security of the data during the migration process, including the handling of sensitive or personal information in compliance with data protection regulations.

1.5 PROPOSED SYSTEM:

Our proposed system can be best described as an ETL (Extraction, Transformation, Loading) tool that basically loads the data from NoSQL files to relational databases. It contains two major elements, Schema analyzer and dynamic data mapper. The schema analyzer analyzes the input file schema thoroughly and then decides and defines the schema of it in the relational database. After this process is complete, the next step is data mapping where the data from the file is mapped with the table in the relational database. By mapping the data from these NoSQL files into relational databases, the user can query the data and extract essential insights from it with Structured Query language. Using relational databases, we can run SQL queries over the data which is mapped from these files. So, our model provides better access and more usability to these files and the crucial information stored in them.

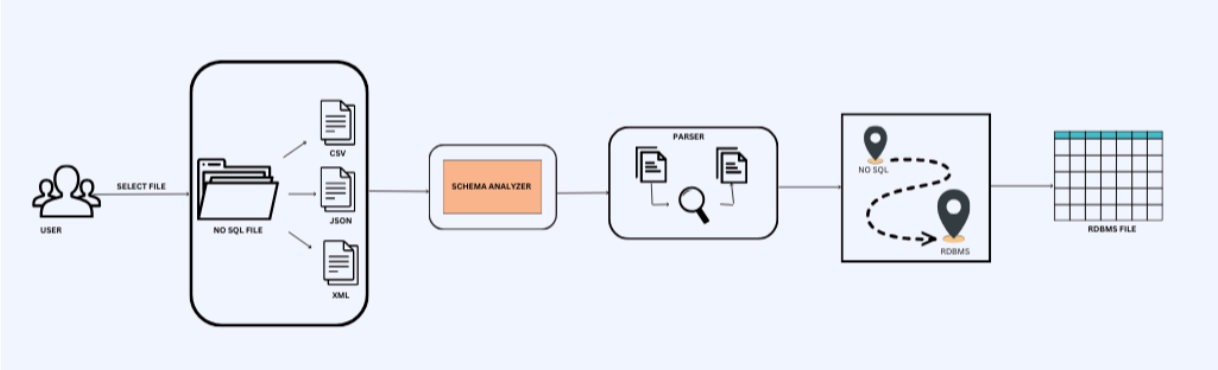


Figure 1.1: Basic representation of proposed system

System Overview

The system will consist of several core components that work together to facilitate the migration from NoSQL to RDBMS:

Schema Translation Module: This component is responsible for analyzing the schema or data structure of the NoSQL database and translating it into an equivalent relational schema that

can be implemented in an RDBMS. It handles the mapping of document-based, key-value, or column family data models to a tabular format.

Data Transformation Engine: This engine converts the data from NoSQL formats (e.g., JSON, BSON) into data formats suitable for RDBMS (e.g., SQL tuples). It includes functionalities for transforming complex, nested structures into a relational format while preserving data integrity and relationships.

Migration Execution Framework: This framework oversees the actual migration process, including data extraction from the NoSQL database, transformation by the Data Transformation Engine, and loading into the RDBMS. It ensures efficient data transfer, minimal downtime, and scalability for large datasets.

User Interface (UI): A graphical user interface that allows users to configure the migration process, including selecting source and target databases, mapping schemas, and initiating the migration. The UI is designed to be intuitive and accessible to users with varying levels of technical expertise.

Validation and Verification Module: This module checks the migrated data for integrity, consistency, and correctness. It includes tools for error detection, reporting, and, if possible, automatic correction of common issues encountered during migration.

Security Layer: This component ensures that all data handled by the system is securely processed and transferred, implementing encryption, access control, and compliance with data protection regulations.

1.6 UNIQUE FEATURES OF THE SYSTEM:

The proposed NoSQL to RDBMS conversion system is designed with several unique features to address the specific challenges of migrating data between fundamentally different database models. These features set it apart from existing solutions by enhancing usability, efficiency, scalability, and data integrity. Here are some of the system's unique features:

1. Efficient Schema Analyzer:

An efficient schema analyzer for converting NoSQL databases to an RDBMS plays a pivotal role by dynamically mapping diverse NoSQL data models to structured relational schemas. It comprehensively understands various NoSQL structures, intelligently translating them into optimized relational formats while preserving data integrity. This tool ensures that the complexities of NoSQL data are seamlessly translated into RDBMS schemas, facilitating a smoother migration process with minimal manual intervention, making it an invaluable component of the NoSQL to RDBMS conversion project.

2. Dynamic Data type mapping:

Features an advanced data type mapping engine capable of automatically identifying and converting complex data types and structures from NoSQL to their closest relational equivalents. This includes handling nested documents, arrays, and custom data types, ensuring a smooth translation with minimal data loss or distortion.

3. Adaptive Parser:

The Proposed system involves an adaptive parser, the main aim of this parser is to parse the data within the given NoSQL files like .csv, Json, .xml, etc. And align these data to the columns of the table generated for migrating these data into the table in the relational database and if there is any missing value or error occurred configure it as per the error handling algorithm applied.

4. Multiple format support:

The system is compatible to process and support multiple formats of NoSQL files over a single format. It supports Comma Separated Values (CSV) files, JavaScript Object Notations (JSON) files and Extensible Markup Language (XML) files. It migrates the data of these files into relational database.

5. Easy Accessibility:

The conversion project aims to make NoSQL data more accessible to users, enabling them to efficiently prepare and execute the migration process with a clear understanding of their data. This approach ensures that the transition from NoSQL to RDBMS preserves the integrity and structure of the data and enhances its usability and accessibility in the new relational environment.

6. Enhanced Security:

The proposed model runs over the local machine which does not expose crucial data over the internet. Enhancing security means implementing robust measures throughout the migration process to protect the data and the systems involved from unauthorized access, data breaches, and potential loss.

7. Data Querying Capability:

By mapping the data from these NoSQL files into relational databases, the user can query the data and extract essential insights from it with Structured Query language. Using relational databases, we can run SQL queries over the data which is mapped from these files. So, our model provides better access and more usability to these files and the crucial information stored in them.

CHAPTER 2

REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

2.1 FEASIBILITY STUDY:

Conducting a feasibility study for a NoSQL to RDBMS conversion project involves assessing the project's viability from technical, economical, and operational perspectives. This comprehensive evaluation ensures that the project can be successfully completed within its constraints and will meet its intended goals. Here's a breakdown of the feasibility study:

TECHNICAL FEASIBILITY:

1. The proposed model provides a better compatibility between NoSQL and RDBMS technologies and the ability to accurately translate and migrate data.
2. All the tools and framework needed for the model's proper functioning are easily available and can be adapted or extended for the user's specific needs.
3. The model is prepared after doing a lot of research work in this field and considering all the scenarios that can happen with this model for providing you with the best deliverables.
4. The model runs on the local machine without any internet requirement which protects the data from exposure to unauthorized people or organizations.
5. Data privacy and integrity are considered as prime parameters to make sure that crucial data of the user should not be misused.

ECONOMIC FEASIBILITY:

The overall project is financially viable. As it mainly involves free open-source resources and the database used is also available without any cost. It utilizes the time most as it involves the major portion of duration in the research part, accessing all the tools and development in this field.

OPERATIONAL FEASIBILITY:

1. The model can be implemented within the organization's operational capabilities and constraints.
2. The model can be implemented over the available organization's IT (Information Technology) infrastructure and personnel to adapt to the new RDBMS environment.

3. Does not require very heavy training, can be easily operated using the manual or short briefing.

4. Efficient model with greater usability and querying power.

2.2 SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT:

2.2.1 INTRODUCTION:

2.2.1.1 PURPOSE:

The purpose of moving from NoSQL data to RDBMS (Relational Database Management System) data depends on the specific needs and requirements of a particular application or organization. NoSQL databases are designed to handle large volumes of unstructured and semi-structured data and can be highly scalable and flexible. They are often used in modern web applications, where data is not necessarily structured in a way that fits well with traditional relational databases. However, RDBMS systems are designed to store structured data in tables with rows and columns and to enforce relationships between different tables. They have been used for decades and have a proven track record of reliability, consistency, and robustness. They are often used in applications that require transactions and data consistency, such as banking, finance, and healthcare.

2.2.1.2 DOCUMENT CONVENTIONS:

DB	Database
DS	Dataset
ER	Entity relationship
SQL	Structured Query Language
NoSQL	Not only SQL
CSV	Comma Separated Values
JSON	JavaScript Object Notation
XML	Extensible Markup Language
RDBMS	Relational Database management system

Table 2.1 : Document conventions

2.2.1.3 INTENDED AUDIENCE:

The intended audience for the NoSQL to RDBMS conversion tool includes database administrators, software developers, and IT professionals responsible for managing and migrating database systems within organizations. These individuals often face the challenge of moving data from NoSQL databases to RDBMS platforms to take advantage of relational database features such as ACID compliance, complex queries, and transaction support. The

tool is also geared towards organizations looking to integrate or consolidate their data management systems for improved analytics, reporting, and data integrity. Additionally, the conversion tool may attract academic researchers and students in computer science or data engineering fields, who are interested in studying database technologies and migration strategies.

2.2.1.4 PROJECT SCOPE:

The project scope for a NoSQL to RDBMS converter encompasses developing a comprehensive software tool designed to automate the migration of databases from various NoSQL formats to a relational database management system (RDBMS). This includes:

1. Schema Translation:

Automatically translating NoSQL database schemas into equivalent RDBMS schemas, considering differences in data models and ensuring integrity in the relational environment.

2. Data Transformation and Mapping:

Converting and mapping NoSQL data types, structures, and relationships into formats compatible with RDBMS, while preserving data accuracy and integrity.

3. User Interface:

Providing an intuitive, user-friendly interface that allows users to configure migration settings, map data fields, and initiate the conversion process with ease.

4. Support for Multiple Databases:

Ensuring compatibility with popular NoSQL databases (e.g., MongoDB, Cassandra, DynamoDB) and RDBMS platforms (e.g., MySQL, PostgreSQL, Oracle) to maximize the tool's utility across different technologies.

5. Performance Optimization:

Implementing efficient data processing algorithms to minimize migration time and system downtime, enabling the handling of large datasets effectively.

6. Security and Compliance:

Incorporating robust security measures, including data encryption and secure data transfer protocols, to protect sensitive information during the migration process. Ensuring compliance with relevant data protection regulations.

7. Validation and Error Handling:

Including mechanisms to validate the migrated data for consistency and completeness and providing comprehensive error reporting and troubleshooting support.

8. Documentation and Support:

Offering detailed documentation, tutorials, and technical support to assist users in the migration process, addressing common challenges and questions.

9. Scalability and Extensibility:

Designing the system to be scalable to handle growing data volumes and extensible to support future NoSQL and RDBMS technologies.

The project aims to streamline the migration process from NoSQL to RDBMS, reducing manual effort, mitigating risks associated with data loss or corruption, and enabling organizations to transition smoothly to a relational database system that meets their evolving data management needs.

2.2.2 OVERALL DESCRIPTION:

2.2.2.1 PRODUCT PERSPECTIVE:

A noSQL to RDBMS file converter model would serve a specific need for users who have noSQL data stored in files and need to convert it to a relational database format for storage and analysis. The model would provide an efficient and user-friendly solution for converting data without requiring the user to have knowledge or expertise in database management or programming. The model would likely have a simple and intuitive interface for users to upload their non-SQL files, select the target RDBMS format, and initiate the conversion process. The model may also offer additional features such as data mapping, data cleansing, and data validation to ensure the accuracy and completeness of the converted data.

2.2.2.2 PRODUCT FEATURES:

A Model that converts noSQL databases to RDBMS (Relational Database Management System) files would typically have the following product features:

1. Input File Format Support: The model should support a wide range of noSQL database file formats as input files, such as CSV, Excel, JSON, MongoDB, or NoSQL databases.

2. Output File Format Support: The model should provide support for various RDBMS file formats as output files, such as SQL Server, Oracle, MySQL, or PostgreSQL.

3. Data Mapping and Conversion: The model should have the capability to map and convert data from the input file to the output file, which includes data types, tables, columns, and relationships.

4. Customization: The model should provide the ability to customize the conversion process by allowing users to select the desired output format, data types, and other parameters.

5. User-Friendly Interface: The model should have a user-friendly interface that simplifies the process of uploading input files, selecting output file formats, and initiating the conversion process.

6. Performance: The model should be designed to handle large input files and convert them efficiently and quickly.

7. Security: The model should be secure and ensure the confidentiality of user data during the conversion process.

8. Error Handling: The model should be able to detect and handle errors during the conversion process and provide appropriate error messages to the user.

9. Technical Support: The model should provide technical support to users who may encounter issues during the conversion process.

10. Cost: The model may offer different pricing models based on the size and complexity of the input files, as well as the desired output format.

2.2.2.3 USER DOCUMENTATION:

Components that will be delivered along with the software:

1. Software Requirement Specification
2. Specification Manual
3. Working Manuals
4. Tutorial Manuals

2.2.2.4 ASSUMPTIONS AND DEPENDENCIES:

The assumptions and dependencies of a noSQL to RDBMS file converter model can be categorized into two broad categories: Technical and Non-Technical.

Technical Assumptions and Dependencies:

1. Input File Formats: The model assumes that the input files are in a format that can be parsed and processed by the conversion algorithm. It depends on the libraries and tools that support the input file format and the ability to extract data from the files.
2. Output File Formats: The model assumes that the output file format is compatible with the RDBMS selected by the user. It depends on the compatibility of the RDBMS and the ability to create tables, columns, and relationships.
4. User Input: The model assumes that the user input is accurate and complete. It depends on the user's ability to provide the correct information about the input file format, output file format, and other parameters.
5. Technical Infrastructure: The model depends on the availability and reliability of the technical infrastructure, including the server, database management system, web server, and internet connection.
Non-Technical Assumptions and Dependencies:
1.Stakeholder Buy-in: It is assumed that all key stakeholders (e.g., management, IT department, end-users) understand the need for and benefits of migrating from NoSQL to RDBMS and are committed to supporting the project.
2. Resource Availability: The project assumes that necessary resources, including personnel with the requisite skills for project management and execution, budget allocation, and access to both the NoSQL and RDBMS platforms, will be available as needed.

3.Training and Adoption: There's an assumption that end-users and IT staff will be willing to undergo training for the new system and that there will be a smooth transition with minimal resistance to change.

4. Data Migration Window: The project assumes that there will be a feasible window of opportunity to conduct the migration with minimal disruption to ongoing operations, especially for businesses that operate 24/7.

5. Vendor Support: If third-party tools or platforms are involved, it is assumed that adequate support and documentation will be available from vendors to facilitate the migration process.

6. Regulatory Compliance: The project assumes that all data handling, storage, and processing within the project scope comply with relevant laws and regulations (e.g., GDPR, HIPAA), without unexpected legal barriers.

Dependencies

1. NoSQL and RDBMS Software Versions: The project's success is dependent on the specific versions of NoSQL and RDBMS software involved, as features and compatibility issues may vary between versions.

2. Data Integrity and Quality: The effectiveness of the conversion tool is dependent on the integrity and quality of the existing NoSQL data. Poorly structured or inconsistent data may require additional preprocessing steps.

3. Infrastructure Readiness: The project depends on the organization's IT infrastructure being ready and capable of supporting the new RDBMS environment, including hardware, network capacity, and security measures.

4. External Consultants or Vendors: If the project relies on external consultants or software vendors for specific expertise or components, timelines and outcomes may depend on their availability and performance.

5. Organizational Policies: The migration process must align with organizational policies regarding data management, IT security, and change management, which can influence project timelines and procedures.

2.2.3 SYSTEM FEATURES:

A noSQL to RDBMS file converter model would typically have the following features:

1. Input format support: The model should support different non-SQL file formats such as CSV, Excel, JSON, and XML.
2. Output format support: The model should support different RDBMS file formats such as SQL Server, MySQL, PostgreSQL, Oracle, and SQLite.
3. Mapping of data types: The model should allow users to map the data types of the non-SQL file to the data types of the RDBMS file to ensure that the data is accurately converted.

4. Data validation: The model should validate the data in the non-SQL file to ensure that it is in the correct format and can be converted to RDBMS format.

5. Data preview: The model should provide a preview of the data in the non-SQL file and the RDBMS file to allow users to check the accuracy of the conversion.

6. Batch conversion: The model should allow users to upload and convert multiple non-SQL files at once.

7. Security: The model should ensure that user data is secure during the conversion process and after the conversion is complete.

8. User-friendly interface: The model should have a user-friendly interface that is easy to navigate and understand.

9. Customization: The model should allow users to customize the conversion process to fit their specific needs.

10. Support: The model should provide support to users in case they encounter any issues during the conversion process.

2.2.3.1 DESCRIPTION AND PRIORITY:

A noSQL to RDBMS file converter model is an ETL tool that allows users to convert data from non-SQL file formats (such as CSV, Excel, JSON, and XML) to RDBMS file formats (such as SQL Server, MySQL, PostgreSQL, Oracle, and SQLite). The priorities of a noSQL to RDBMS file converter website would depend on the specific needs of the users. However, some common priorities could include Accuracy, Speed, User-friendliness, Customization, Security, Support, Cost. Overall, the priorities of a noSQL to RDBMS file converter website should be to provide a reliable, efficient, and user-friendly tool that helps users easily and accurately convert their data between different file formats.

2.2.3.2 FUNCTIONAL REQUIREMENTS:

Functional requirements of a noSQL to RDBMS file converter model could include the following: File upload, File Selection, Target database selection, Data type mapping, Data validation, Data preview, Data export, User authentication, Security, Support.

2.2.4 OTHER NONFUNCTIONAL REQUIREMENTS:

2.2.4.1 PERFORMANCE REQUIREMENTS:

To determine the performance requirements of a noSQL to RDBMS file converter model, several factors should be considered, including:

1. Data volume: The size of the files to be converted will affect the model's performance requirements. Larger files will require more processing power and memory and may take longer to convert.
2. Concurrent users: The number of users accessing the model simultaneously will affect the model's performance. The model should be able to handle multiple users and conversions simultaneously without any delays or errors.
3. Processing speed: The speed at which the model can convert the files is crucial to its performance. The model should be able to process the files quickly and efficiently, without causing any delays or timeouts.
4. Database complexity: The complexity of the database being used will affect the model's performance requirements. More complex databases may require more processing power and memory to handle the conversion.
5. Security: The model should be designed to handle data securely and protect user data from unauthorized access.
7. Error handling: The model should be able to handle errors and exceptions gracefully, without crashing or causing data loss.

Based on these factors, the performance requirements of a noSQL to RDBMS file converter model will vary. The model should be designed to handle the expected data volume, concurrent users, and processing speed, while also providing adequate security and error handling.

2.2.4.2 SAFETY REQUIREMENTS:

When it comes to safety requirements of a noSQL to RDBMS file converter model , there are several things to consider: Security, Privacy, Data Integrity, Testing and Validation, Backup and recovery.

2.3 SDLC MODEL TO BE USED:

In the development phase of this model, we make use of **Agile Software Development Life Cycle (SDLC)**. For a complex and technically challenging project like a NoSQL to RDBMS converter, the Agile SDLC (Software Development Life Cycle) model is highly recommended. Agile is well-suited for projects requiring flexibility, incremental development, and close collaboration with all the developers involved in the project. Agile Software Development Life Cycle (SDLC) is the combination of both iterative and incremental process models. It focuses on process adaptability and customer satisfaction by rapid delivery of working software products. Agile SDLC breaks down the product into small incremental builds. These builds are provided into iterations.

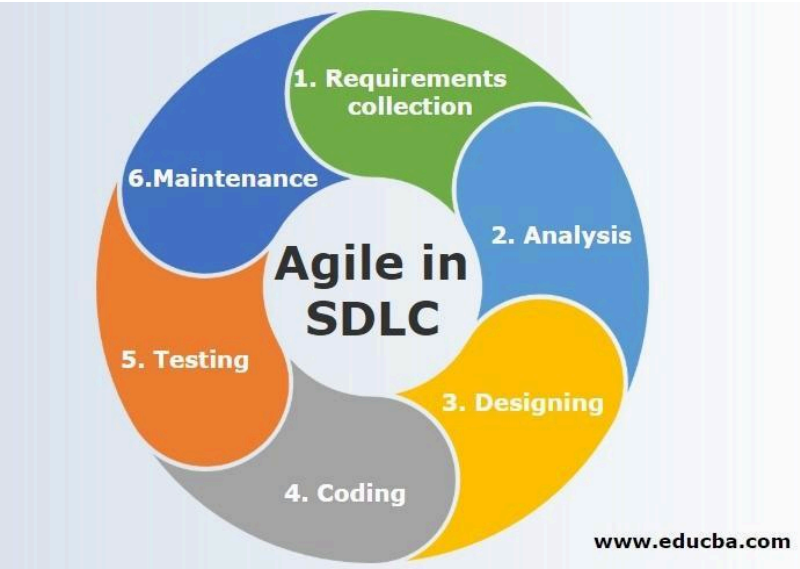


Figure 2.1 : Phases of Agile SDLC

In the agile SDLC development process, the customer can see the result and understand if he/she is satisfied with it. This is one of the advantages of the agile SDLC model. One of its disadvantages is the absence of defined requirements so it is difficult to estimate the resources and development cost.

Here's why Agile is a good fit and how it can be applied to this project:

Why Agile?

1. **Iterative Development**: Agile allows for the development of the converter in small, manageable increments, ensuring that each module (like schema translation, data transformation, etc.) can be developed, tested, and improved upon iteratively. This approach is crucial for handling the complexities and unforeseen challenges of data migration projects.
2. **Adaptability**: Given the potential for changing requirements (such as support for new NoSQL or RDBMS systems) and the learning curve associated with understanding the intricacies of different database models, Agile's adaptability is a significant advantage.
3. **Stakeholder Engagement**: Agile emphasizes regular feedback from users and stakeholders, which is vital for ensuring that the converter meets the real-world needs of those migrating their databases. This continuous feedback loop can help in refining features and usability aspects of the tool.
4. **Risk Management**: By prioritizing work on the most critical features and components first, Agile helps in early identification and mitigation of risks, especially those related to data integrity, security, and performance.

Applying Agile to the Project

1. **Sprint Planning**: Divide the project into sprints, each focusing on a specific set of features or components, such as schema translation in one sprint, followed by data transformation in the next.
2. **Daily Stand-ups**: Implement daily stand-up meetings to discuss progress, identify any obstacles, and ensure alignment of the development team's efforts with the project goals.
3. **Sprint Reviews and Retrospectives**: At the end of each sprint, we conduct a review meeting with all the members to demonstrate the developed features and gather feedback. Follow this with a retrospective meeting within the development team to reflect on what went well, what didn't, and how processes can be improved in the next sprint.
4. **Backlog Refinement**: Maintaining a product backlog that lists all desired features, improvements, and bug fixes. Regularly refine and prioritize this backlog based on

feedback, changing requirements, and project progress to ensure that the team is always working on the most valuable tasks.

5. **Continuous Integration and Deployment (CI/CD)**: Employ CI/CD practices to automate testing and deployment processes, ensuring that new features and fixes are integrated and tested frequently. This helps in maintaining high-quality standards and rapid delivery.

By following the Agile model, the development of the NoSQL to RDBMS converter can be more responsive to the needs of users, adapt to technical challenges as they arise, and ultimately deliver a more effective and reliable tool.

CHAPTER 3

SYSTEM DESIGN

3.1 DETAIL DESIGN

Our proposed system can be best described as an ETL (Extraction, Transformation, Loading) tool that basically loads the data from NoSQL files to relational databases. It contains two major elements, Schema analyzer and dynamic data mapper. The schema analyzer analyzes the input file schema thoroughly and then decides and defines the schema of it in the relational database. After this process is complete, the next step is data mapping where the data from the file is mapped with the table in the relational database. By mapping the data from these NoSQL files into relational databases, the user can query the data and extract essential insights from it with Structured Query language. Using relational databases, we can run SQL queries over the data which is mapped from these files. So, our model provides better access and more usability to these files and the crucial information stored in them.

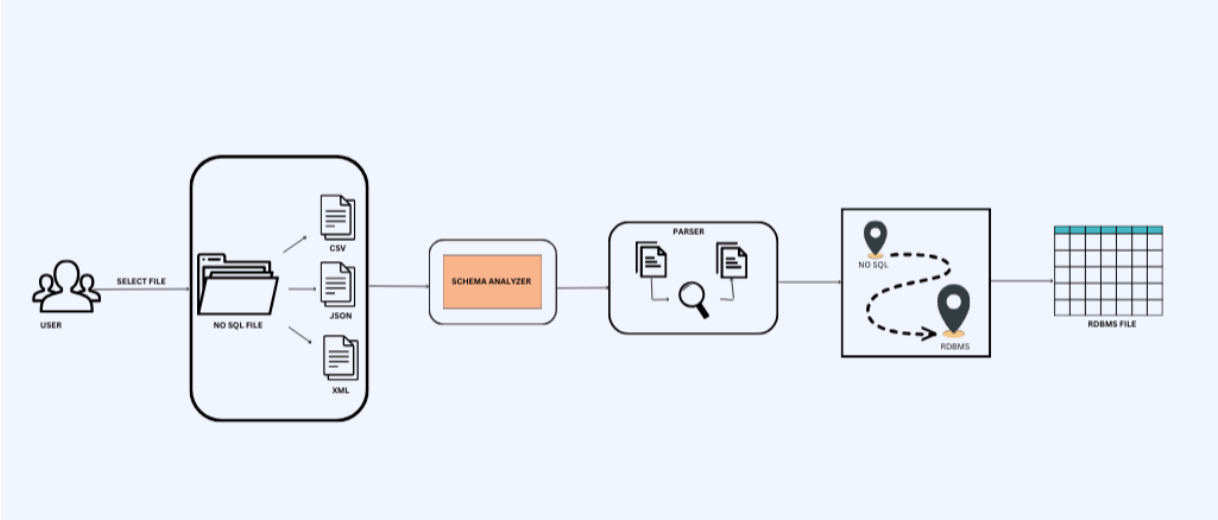


Figure 3.1 : Detail Design

The existing methods for transforming NoSQL databases into relational databases require manual schema mapping, which is laborious and needs subject expertise. An efficient and automatic process is needed to transform an unstructured NoSQL database into a structured database. We proposed and evaluated a practical method in this study for automatically transforming a NoSQL database into a relational database. In our experimental evaluation, we

used a variety of NoSQL files, such as ".CSV," ". JSON," and ".XML" files, together with MySQL as a relational database to perform transformation operations for different dataset sizes. As transferring data from a NoSQL database to a relational database, we observed exceptional performance as compared to the state-of-the-art approaches used at the time.

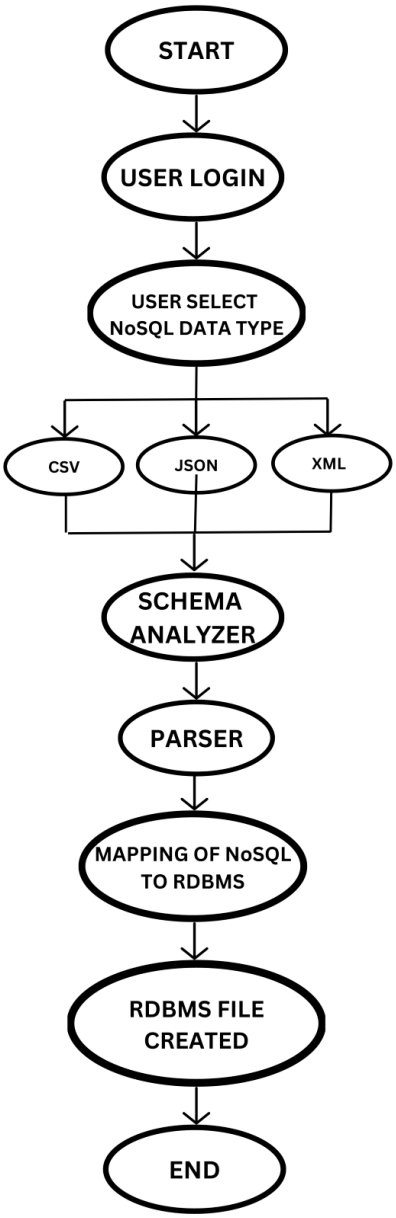


Figure 3.2 : Flow Diagram

3.2 SYSTEM DESIGN USING DFD LEVEL 0 AND DFD LEVEL 1:

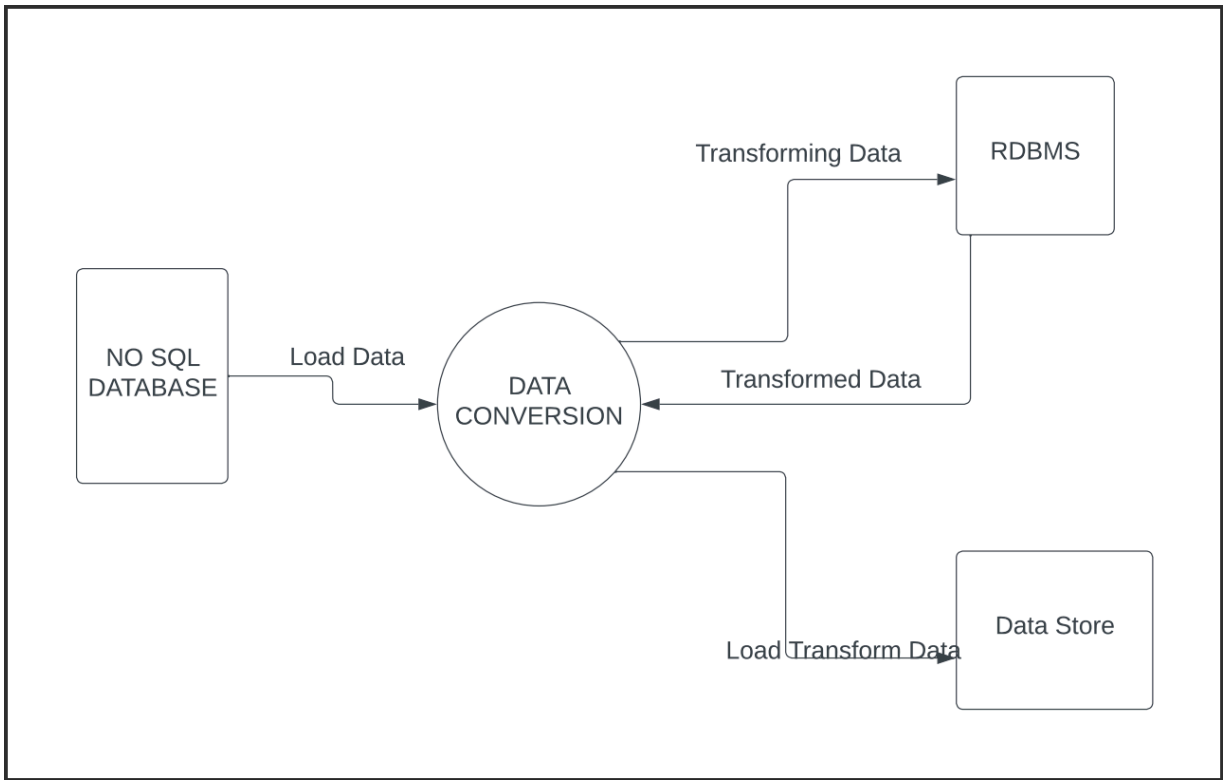


Figure 3.3 : DFD level 0

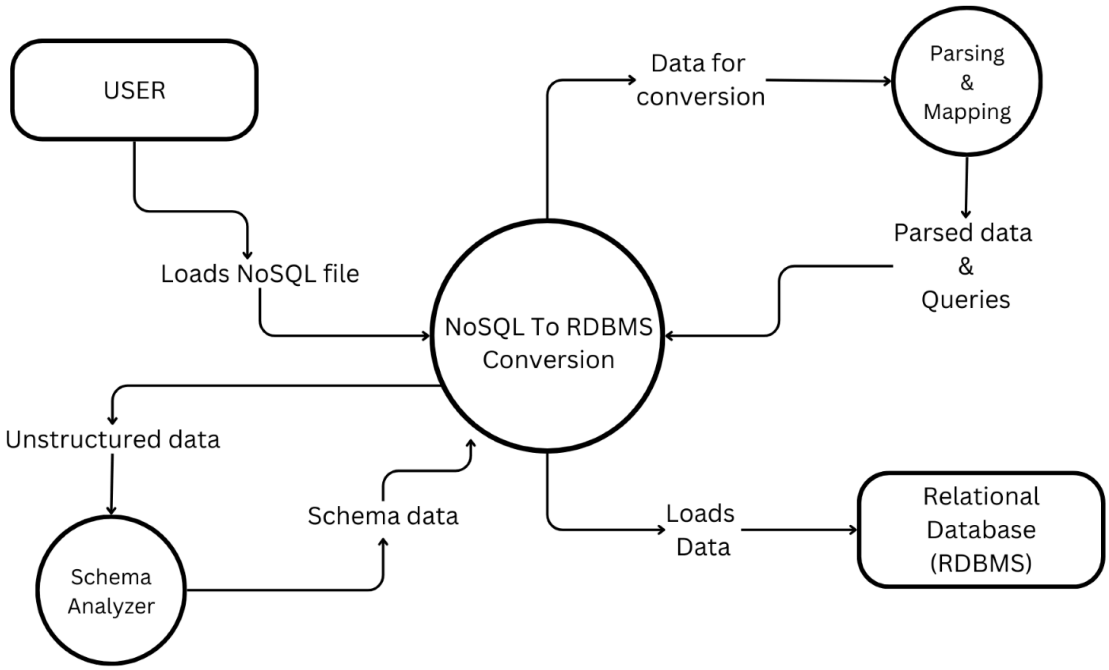


Figure 3.4 : DFD level 1

3.3 USE CASE DIAGRAM

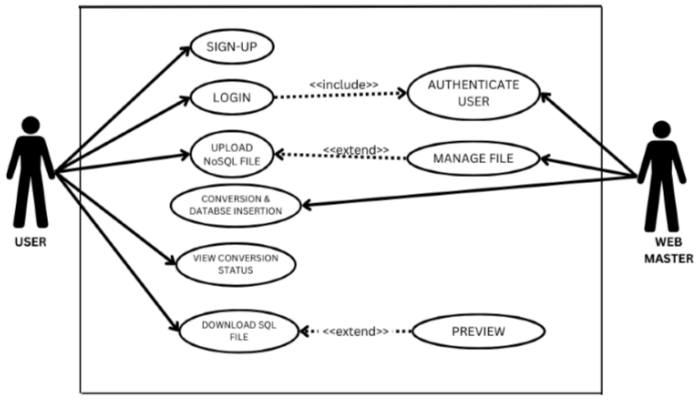


Figure 3.5: Use case diagram

3.4 ER DIAGRAM

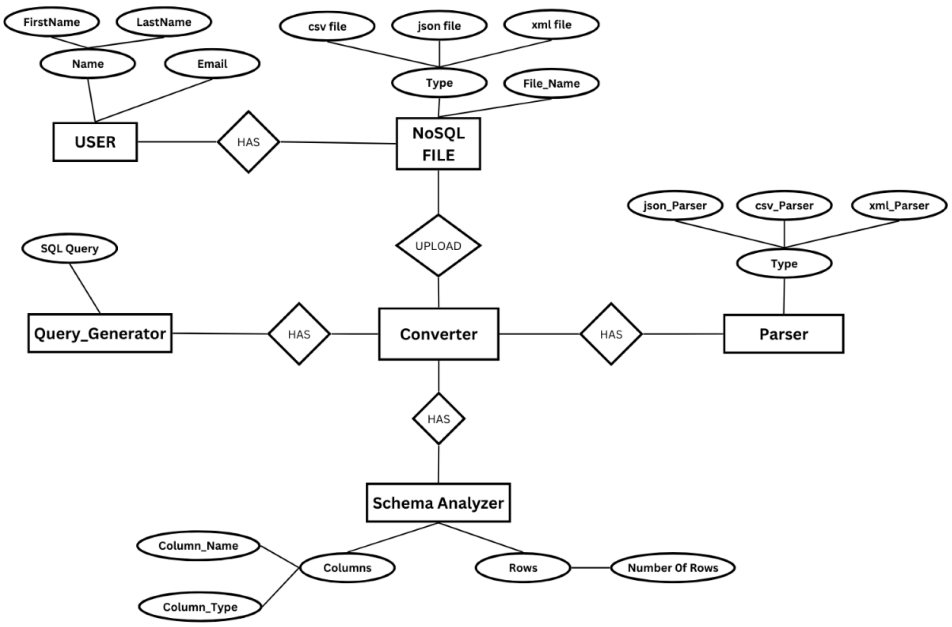


Figure 3.6: ER diagram

CHAPTER 4
IMPLEMENTATION, TESTING AND MAINTENANCE

4.1 INTRODUCTION TO LANGUAGES, TOOLS AND TECHNOLOGIES USED FOR IMPLEMENTATION:

XAMPP:

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. It simplifies the process of installing and configuring these components to create a local web server for testing and development purposes.

Here's a brief overview of its components:

1. Apache: A widely used web server software that is responsible for serving web pages to users' browsers.
2. MariaDB: A database management system, derived from MySQL, used to store and manage data for your site in a relational database.
3. PHP: A popular server-side scripting language used for web development to create dynamic web pages.
4. Perl: Another scripting language that is often used for web development and network programming.

XAMPP is available for Windows, Linux, and macOS. It's designed to be very easy to install and to use, making it a popular choice for developers who want to develop and test their web applications on their local machines before deploying to a live server. The "X" in XAMPP stands for cross-platform, meaning it can run on any operating system that supports these components.

XAMPP also comes with additional tools such as phpMyAdmin for database management, FileZilla FTP Server, Mercury Mail for mail serving, and Tomcat for Java servlet and JSPs. It's a versatile package that can support a wide range of web development needs.

PHP:

PHP, which stands for "PHP: Hypertext Preprocessor," is a popular general-purpose scripting language that is especially suited to web development. It was originally created by Rasmus Lerdorf in 1994, and its open source. PHP scripts are executed on the server, and the result is returned to the browser as plain HTML.

Features of PHP:

1. Server-Side Scripting: PHP is mainly used for server-side scripting. This means that you can create dynamic page content, collect form data, and send and receive cookies, among other things.
2. Easy to Learn: PHP syntax is quite similar to C and Java, which makes it easy for individuals with background in these languages to pick up PHP.
3. Cross-Platform: PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.) and can be integrated with many servers (Apache, IIS, etc.).
4. Database Integration: PHP supports a wide range of databases, including MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and others. This makes it a great choice for web applications that need database interaction.
5. Embedded: PHP code can be easily embedded within HTML code by using the special PHP tags. This allows for the creation of dynamic content without needing to call external files for processing.

Applications of PHP:

1. Web Pages and Web-Based Applications: Creating dynamic and interactive web pages and applications.
2. Content Management Systems (CMS): Many popular CMS, like WordPress, Drupal, and Joomla, are built with PHP.
3. E-commerce Applications: PHP is widely used in developing e-commerce platforms.
4. Web Frameworks: Frameworks like Laravel, Symfony, and CodeIgniter are based on PHP, facilitating rapid application development.
5. Data Analysis and Visualization: With the right libraries, PHP can be used for data analysis and generating visual reports.

PHP Versions:

Over the years, PHP has evolved significantly, with each new version bringing performance improvements, new features, and better security. It's important to keep PHP versions up to date to take advantage of these improvements and to maintain security.

PHP is a pivotal technology in web development, and its widespread use across CMS platforms and web applications underscores its importance. Whether you're building a simple website or a complex web application, PHP offers a rich ecosystem of tools and frameworks to support your development process.

MySQL:

MySQL is an open-source relational database management system (RDBMS) that relies on Structured Query Language (SQL) for processing the data in the database. MySQL is widely used for web applications and acts as the database component of the LAMP, MAMP, and XAMPP web development stacks, among others, with LAMP standing for Linux, Apache, MySQL, and PHP/Perl/Python.

Key Features of MySQL:

1. Cross-Platform Support: MySQL can be run on various platforms, including Linux, Unix, Windows, and macOS, making it versatile for different environments.
2. Reliability and Performance: Known for its reliability and high performance, MySQL is used in demanding production environments. Advanced caching, replication, and batch processing enhance its performance.
3. Scalability: It can handle a wide range of application demands, scaling from small to large enterprise operations with many concurrent users.
4. Security: MySQL provides strong data protection mechanisms, including encrypted connections and authentication processes that ensure secure data transactions.
5. Comprehensive Application Development: One of MySQL's advantages is its architecture, which allows for full flexibility in application development, offering features like stored procedures, triggers, and views.
6. Wide Range of Application: It's used in a wide range of applications, from web databases to logging applications, e-commerce, and data warehousing.

Applications of MySQL:

1. Web Databases: The most common use of MySQL is for web databases, where it stores data for web applications, such as user information, posts, and comments.
2. Content Management Systems (CMS): MySQL serves as the backend database for popular CMS platforms like WordPress, Joomla, and Drupal.
3. E-commerce Applications: Online stores and e-commerce platforms use MySQL to store product inventory, customer information, and transaction histories.
4. Data Warehousing and Business Intelligence (BI): Organizations use MySQL for data warehousing purposes, where they can store large volumes of data and perform complex queries for business insights.

MySQL Versions:

MySQL has undergone significant evolution, with improvements in performance, security, and features over the years. MySQL 8.0 is one of the latest versions, introducing enhanced JSON support, window functions, and CTEs (Common Table Expressions), among other features. Keeping your MySQL version up to date is crucial for benefiting from these improvements and maintaining security.

MySQL competes with other RDBMS systems like PostgreSQL, Microsoft SQL Server, and Oracle Database. However, its simplicity, speed, and compatibility with major web technologies make it a preferred choice for many developers and businesses.

VISUAL STUDIO:

Visual Studio is an integrated development environment (IDE) from Microsoft used for developing computer programs, websites, web apps, web services, and mobile apps. It offers developers a single application in which they can use various development tools, languages, and libraries to build a wide range of applications. Visual Studio supports development in several programming languages including, but not limited to, C#, Visual Basic .NET, C++, JavaScript, Python, and more. Its capabilities can be extended by developers and third-party companies using plugins.

Key Features of Visual Studio

1. Multiple Language Support: Supports development in C#, VB.NET, C++, F#, JavaScript, Python, and more, making it versatile for different types of development projects.

2. Powerful Debugging and Diagnostics: Offers robust debugging features, including the ability to set breakpoints, step through code, inspect variables, and analyze memory and CPU usage.

3. Integrated Development Environment (IDE): Provides a comprehensive environment with features like code editor, debugger, GUI design, database schema design, and version control integration.

4. Extensibility: Visual Studio can be extended with a vast array of extensions available through the Visual Studio Marketplace, allowing developers to add new features and support for different languages and frameworks.

5. Collaboration and Version Control: Integrated support for Git and Team Foundation Version Control (TFVC) enables developers to collaborate on projects and manage code changes efficiently.

6. Mobile Development: With Xamarin integration, developers can build and debug cross-platform mobile apps for Android, iOS, and Windows using C#.

7. Web Development: Offers extensive tools for building web applications, including ASP.NET development, JavaScript editing, and powerful frameworks like .NET Core.

8. Cloud Development: Direct integration with Microsoft Azure allows developers to build, test, manage, and deploy cloud applications directly from the IDE.

Editions of Visual Studio

Visual Studio is available in several editions, catering to different needs and budgets:

1. Visual Studio Community: A free edition for individual developers, open-source projects, academic research, and small teams. It offers many of the features of the more expensive editions.

2. Visual Studio Professional: Offers more advanced features and tools for professional developers and small teams, with subscription-based or standalone licensing.

3. Visual Studio Enterprise: The most comprehensive edition, providing advanced debugging, AI-driven code recommendations, testing tools, and more, for large development teams and enterprises.

Visual Studio Code:

It's worth mentioning Visual Studio Code (VS Code), a free, open-source editor also developed by Microsoft. VS Code is lighter and more customizable than Visual Studio IDE, supports development in multiple languages through extensions, and is available on Windows, Linux, and macOS. While VS Code shares part of its name with Visual Studio, it's a separate product focusing more on being a powerful code editor rather than a full IDE.

4.2 TESTING TECHNIQUES AND TEST CASES USED:

SCOPE:

Scope defines the features, functional or non-functional requirements of the software that will be tested. Features of the Project:

1. File format support: The converter website would need to support various NoSQL file formats, such as JSON, BSON, and XML, and convert them to relational database formats such as MySQL, PostgreSQL, or Oracle.
2. Schema mapping: The converter would need to map the NoSQL data schema to the relational database schema. This may involve detecting data types, relationships, and constraints, and generating the SQL schema definition language (SDL) commands to create the tables, columns, and indexes.
3. Data transformation: The converter would need to transform the NoSQL data to fit the relational database schema. This may involve flattening nested data structures, splitting data across multiple tables, and handling data type conversions.
4. User interface: The converter website would need to provide a user-friendly interface for users to upload their NoSQL files, specify the target RDBMS system, and configure any additional settings, such as data mapping rules.
5. Security: The converter website would need to ensure the security of the user's data and protect against unauthorized access or data breaches.

OUT OF SCOPE:

Out Of Scope defines the features, functional or non-functional requirements of the software that will NOT be tested:

1. Custom Knowledge Database: Ensure the database provides accurate and reliable descriptions and context to users.
2. Scalability: Load testing to ensure the platform can handle increased user loads efficiently.

Quality Objective:

Here makes a mention of the overall objective that you plan to achieve without your testing Some objectives of your testing project could be:

Ensure the Application Under Test conforms to functional and nonfunctional requirements.
Ensure the AUT meets the quality specifications defined by the client.
Bugs/issues are identified and fixed before going live.

Roles and Responsibilities:

Detail description of the Roles and responsibilities of different team members like:

QA Analyst: Vageesha Rai

Test Manager: Prof. Shreela Pareek

Configuration Manager: Prof. Neha Shukla

Developers: Saumya Singh, Vageesha Rai, Vikas Kumar Verma

Installation Team: Prof. Shreela Pareek, Prof. Neha Shukla, Vageesha Rai, Saumya Singh, Vikas Kumar Verma

Test Methodology:

Overview

We are using an iterative testing approach to make sure our project works well. This means we test it in small steps, starting with checking if each part works on its own. Then, we see how different parts work together. We keep testing as we make changes and add new things. This way, we make sure our project is always working well, even after modification.

Test Levels

Test Levels define the Types of Testing to be executed on the Application Under Test (AUT). We aim to test our project at the following levels:

1) Unit Testing: This is the lowest level of testing and focuses on individual components or functions within the software. Developers often perform unit tests to verify that specific parts of the code work correctly.

2) Integration Testing: This level of testing checks how different components or modules of the software work together. It ensures that integrated parts of the software function as intended.

3) System Testing: At this level, the whole system is tested. It verifies that the software meets its specified requirements and functions properly in its intended environment.

Test Completeness:

Here you define the criteria that will deem your testing complete. For instance, a few criteria to check Test Completeness would be

• 100% test coverage

• All Manual & Automated Test cases executed

• All open bugs are fixed or will be fixed in next release

Test Deliverables

Here are the deliverables

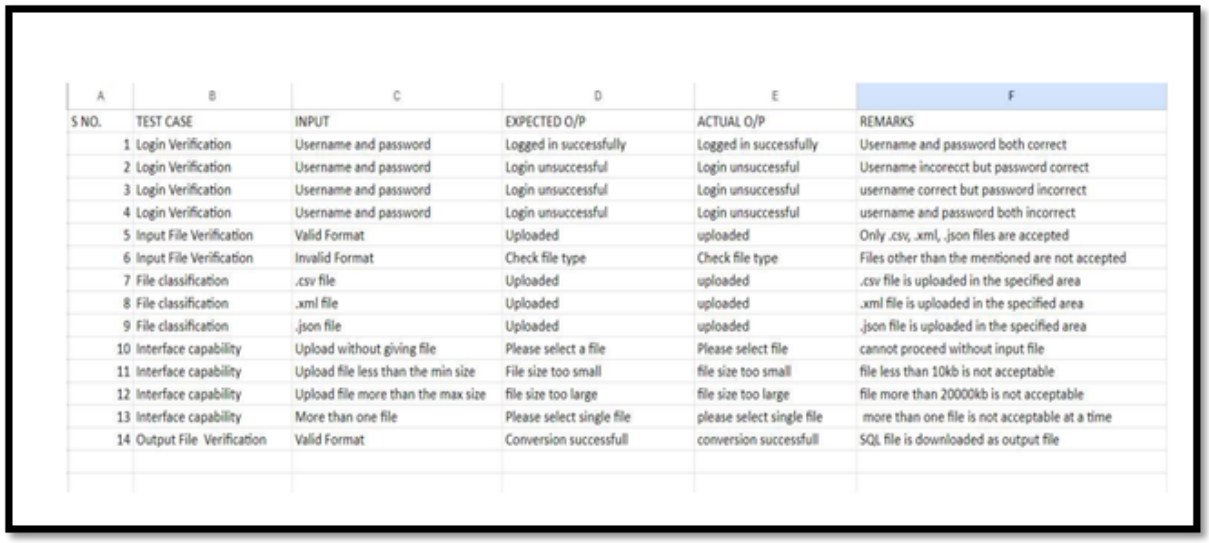
• Test Plan

• Test Cases

• Bug Reports

• Test Strategy

Test Cases:



A	B	C	D	E	F
S NO.	TEST CASE	INPUT	EXPECTED O/P	ACTUAL O/P	REMARKS
1	Login Verification	Username and password	Logged in successfully	Logged in successfully	Username and password both correct
2	Login Verification	Username and password	Login unsuccessful	Login unsuccessful	Username incorecct but password correct
3	Login Verification	Username and password	Login unsuccessful	Login unsuccessful	username correct but password incorrect
4	Login Verification	Username and password	Login unsuccessful	Login unsuccessful	username and password both incorrect
5	Input File Verification	Valid Format	Uploaded	uploaded	Only .csv, .xml, .json files are accepted
6	Input File Verification	Invalid Format	Check file type	Check file type	Files other than the mentioned are not accepted
7	File classification	.csv file	Uploaded	uploaded	.csv file is uploaded in the specified area
8	File classification	.xml file	Uploaded	uploaded	.xml file is uploaded in the specified area
9	File classification	.json file	Uploaded	uploaded	.json file is uploaded in the specified area
10	Interface capability	Upload without giving file	Please select a file	Please select file	cannot proceed without input file
11	Interface capability	Upload file less than the min size	File size too small	file size too small	file less than 10kb is not acceptable
12	Interface capability	Upload file more than the max size	file size too large	file size too large	file more than 20000kb is not acceptable
13	Interface capability	More than one file	Please select single file	please select single file	more than one file is not acceptable at a time
14	Output File Verification	Valid Format	Conversion successfull	conversion successfull	SQL file is downloaded as output file

Figure 4.1 : test cases

BOUNDARY VALUE ANALYSIS:

Input file size must be between 10-20000 kB

Invalid (min-1)	Valid (min, min + 1, nominal, max – 1, max)	Invalid (max + 1)
1KB	10KB, 11KB, 10005KB, 19999KB, 20000KB	20001KB

Table 4.1: BVA table

	A	B	C	D	E	F	G	H
1	S NO.	TEST CASE	INPUT	EXPECTED O/P	ACTUAL O/P	REMARKS	RESULTS	
2	1	.csv file	1	Invalid i/p	Invalid i/p	File .csv but size less than min	Fail	
3	2	.csv file	20	Valid i/p	Valid i/p	File .csv and size is betwwen the valid range	pass	
4	3	.csv file	50000	Invalid i/p	Invalid i/p	file .csv but size greater than max	fail	
5	4	.xml file	9	Invalid i/p	Invalid i/p	File .xml but size less than min	fail	
6	5	.xml file	500	Valid i/p	Valid i/p	file .xml and size is between valid range	pass	
7	6	.xml file	10000	Invalid i/p	Invalid i/p	file .xml but size greater than max	fail	
8	7	.json file	2	Invalid i/p	Invalid i/p	file .json but size less than min	fail	
9	8	.json file	1000	Valid i/p	valid i/p	file .json and size between the valid range	pass	
10	9	.json file	20001	Invalid i/p	Invalid i/p	file .json but size less than max	fail	
11								
12								
13								
14								
15								
16								
17								
18								

Figure 4.2: BVA Test cases

Equivalence Class Testing:

No of files acceptable as input = 1

Invalid	Valid	Invalid
0	1	2, 20, 200,.....

Table 4.2: Equivalence class table

	A	B	C	D	E	F	G
1	TEST CASE	INPUT	EXPECTED O/P	ACTUAL O/P	REMARKS		
2	1	1	valid i/p file	valid i/p file	only one file is accepted as a i/p at a time		
3	2	2	invalid i/p file	invalid i/p file	multiple files cannot be accepted as an i/p		
4	3	0	invalid i/p file	invalid i/p file	no i/p will lead to no result		
5	4	10	invalid i/p file	invalid i/p file	multiple files cannot be accepted as an i/p		
6	5	-20	invalid i/p file	invalid i/p file	no such i/p is possible		
7	6	50	invalid i/p file	invalid i/p file	multiple files cannot be accepted as an i/p		
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							

Figure 4.3: EC Test cases

DECISION TABLE:

Conditions	Input-1	Input-2	Input-3	Input-4
Type	T	T	F	F
Size	T	F	T	F
Result	Accepted	Not Accepted	Not Accepted	Not Accepted

Table 4.3: decision table

Example:

Conditions	Input-1	Input-2	Input-3	Input-4	Input-5	Input-6
Type	.csv	.csv	.xml	.xml	.json	.json
Size	10kb-20000kb	>=20000kb	10kb-20000kb	>=20000kb	10kb-20000kb	>=20000kb
Result	Accepted	Not Accepted	Accepted	Not Accepted	Accepted	Not Accepted

Figure 4.4: Decision test Cases

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 USER INTERFACE REPRESENTATION:

5.1.1 BRIEF DESCRIPTION OF EACH MODULE:

Our module consists of three modules:

CSV converter:

A CSV (Comma Separated Values) to relational table converter is a tool or utility that transforms CSV file data into a format suitable for storage in a relational database. This process typically involves parsing the CSV file to extract data rows, which are then inserted into a database table following the relational database schema. The conversion process may include several steps:

1.Reading the CSV File: The converter reads the CSV file line by line, interpreting each line as a record. CSV files are simple text files where each line represents a data record, and each record consists of fields separated by commas.

2. Parsing and Validation: Each line is parsed to separate the fields based on the delimiter (commonly a comma, but sometimes other characters like semicolons or tabs). During this step, data can also be validated to ensure it conforms to expected types and formats.

3. Mapping to a Relational Schema: The fields from the CSV are mapped to columns in a relational database table. This step may involve transforming the data to fit the data types and constraints of the database schema, such as converting strings to numbers or dates, trimming extra spaces, or applying default values.

4. Insertion into Database: Once the data is parsed and mapped, it's inserted into the relational table. This can be done using SQL INSERT statements, bulk insert operations for efficiency, or through the use of database-specific tools and APIs.

JSON Converter:

A JSON (JavaScript Object Notation) to relational table converter transforms JSON data into a format suitable for storage in a relational database table. JSON is a lightweight data-interchange format that is easy for humans to read and write and for machines to parse

and generate. It's often used in web applications for data exchange and in various systems for data storage because of its flexibility and human-readable format. However, when it comes to storing this data in a relational database, the hierarchical nature of JSON needs to be flattened or transformed into a tabular format. Here are the steps typically involved in converting JSON to a relational table:

1. Reading the JSON Data: The converter reads the JSON data, which might be in a file or a string. JSON data can represent simple key-value pairs, arrays, or nested objects.
2. Parsing JSON Data: The data is parsed using a JSON parser, which converts the JSON format into a structure that can be easily manipulated by the programming language being used (for example, dictionaries in Python or objects in JavaScript).

3. Mapping to a Relational Schema: This is a crucial step where the hierarchical JSON data needs to be mapped to a flat relational schema. This often involves:
   - Identifying primary keys and foreign keys to maintain relationships among data that was nested in the JSON.
   - Deciding how to map arrays or nested objects to relational tables. This might involve creating additional tables to maintain the structure and relationships in the data.
   - Converting data types from JSON to types suitable for the relational database (e.g., converting timestamps to the database's date/time type).
4. Creation of Database Schema (Optional): Based on the mapping, the corresponding database schema (tables with specific columns and data types) may need to be created if it doesn't already exist.
5. Insertion into Database: The mapped data is then inserted into the database tables. This can involve generating and executing SQL INSERT statements or using bulk insert tools provided by the database management system for efficiency.

The complexity of converting JSON to a relational table largely depends on the structure of the JSON data. Simple, flat JSON objects might be straightforward to convert, while deeply nested objects or arrays may require more complex logic to ensure all relationships and data integrity are preserved in the relational model.

XML Converter:

An XML (eXtensible Markup Language) to relational table converter transforms XML data into a format suitable for storage in a relational database table. XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It's widely used for representing complex data structures in a hierarchical format, similar to JSON but with a different syntax and more extensive support for namespaces and schemas. The process of converting XML to a relational table involves several steps, much like converting JSON or CSV, but with some differences due to XML's unique features:

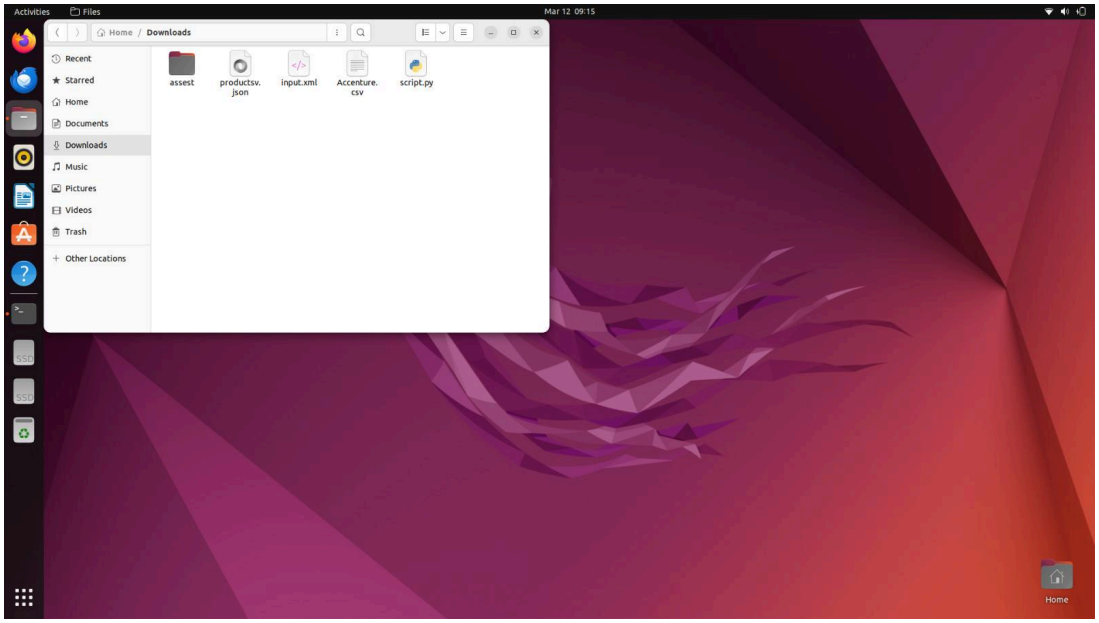
1. Reading the XML Data: The converter reads the XML document, which could be stored in a file, a string, or accessed through a network. XML data consists of elements (tags) that may contain attributes, text content, and nested elements.
2. Parsing XML Data: Using an XML parser, the data is converted into a structure that can be easily manipulated by the programming language being used. This step involves interpreting the tags, attributes, and text content according to the rules defined in the XML document or its associated schema (if available).
3. Mapping to a Relational Schema: This step involves translating the hierarchical structure of XML into a flat relational model. The process includes:
   - Identifying elements and attributes that will become tables and columns in the relational database. Elements often translate to tables, while attributes and text content translate to columns.
   - Determining primary keys for the tables and foreign keys to preserve the relationships between elements that were nested within the XML.
   - Dealing with repeating elements, which may require creating separate tables to maintain a normalized database structure.
   - Converting XML data types to types compatible with the relational database.
4. Creation of Database Schema (Optional): Based on the mapping process, the appropriate database schema might need to be created, including the definition of tables, columns, data types, and relationships if it doesn't already exist.

5. Insertion into Database: Finally, the mapped data is inserted into the database. This could involve generating SQL INSERT statements for each piece of data or utilizing bulk insert operations provided by the database management system for more efficient data loading.

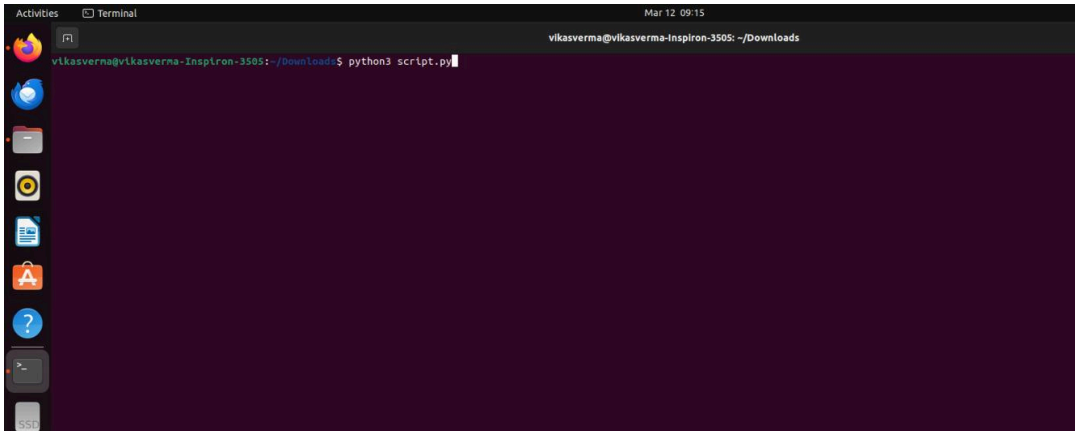
The conversion from XML to a relational table can be complex, especially for XML documents with deep nesting, mixed content (elements containing both text and child elements), or extensive use of attributes. The goal is to preserve the document's structure and semantics within the constraints of a relational model, which may not always be straightforward.

5.2 SNAPSHOTS OF THE SYSTEM WITH BRIEF DETAIL:

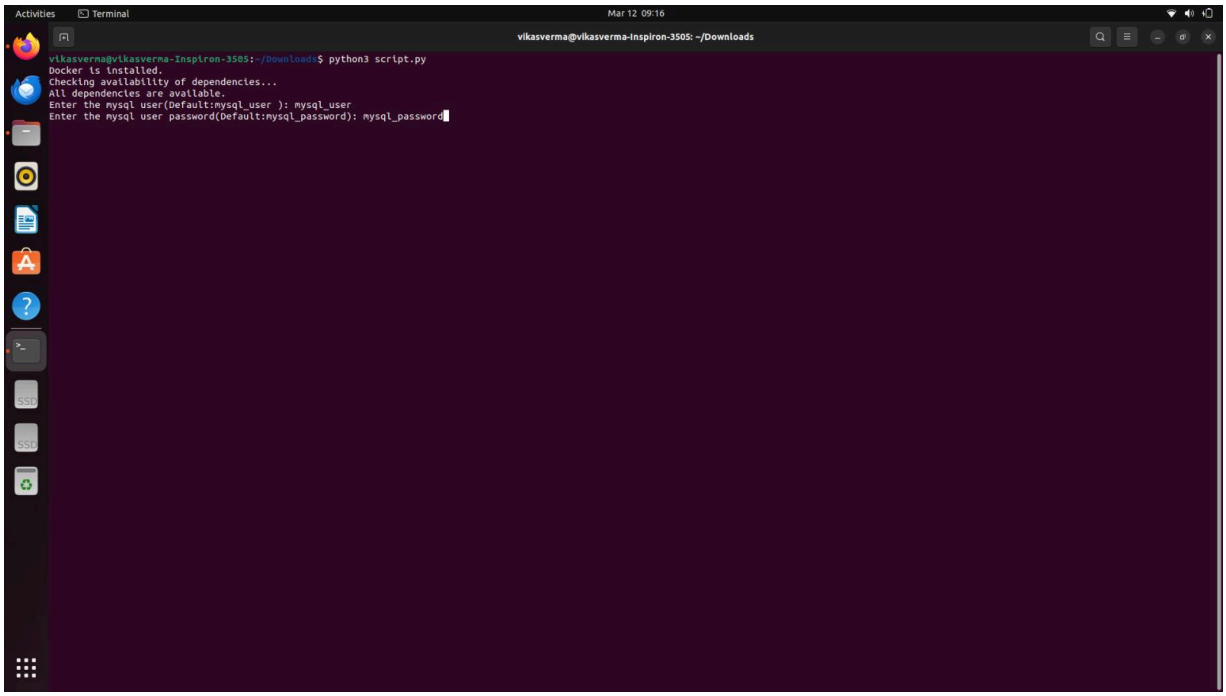
Open the location of software package in the system:

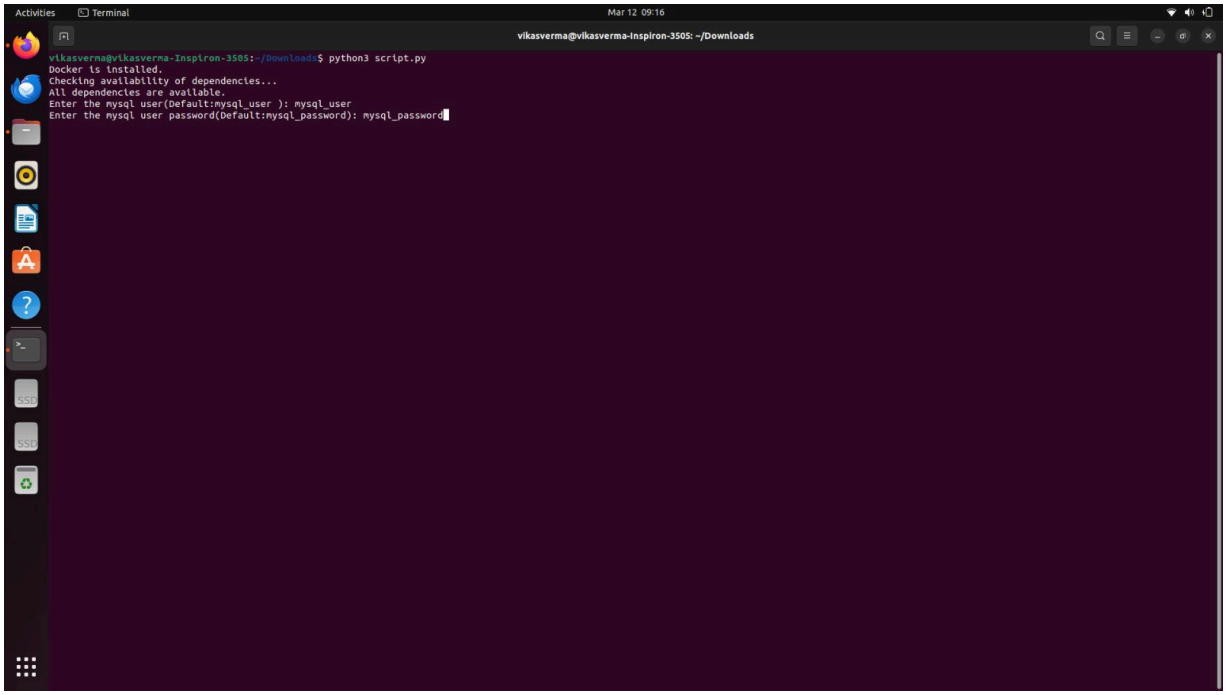


Run the terminal over the location and write command “python3 script.py” (script.py is the name of the package):

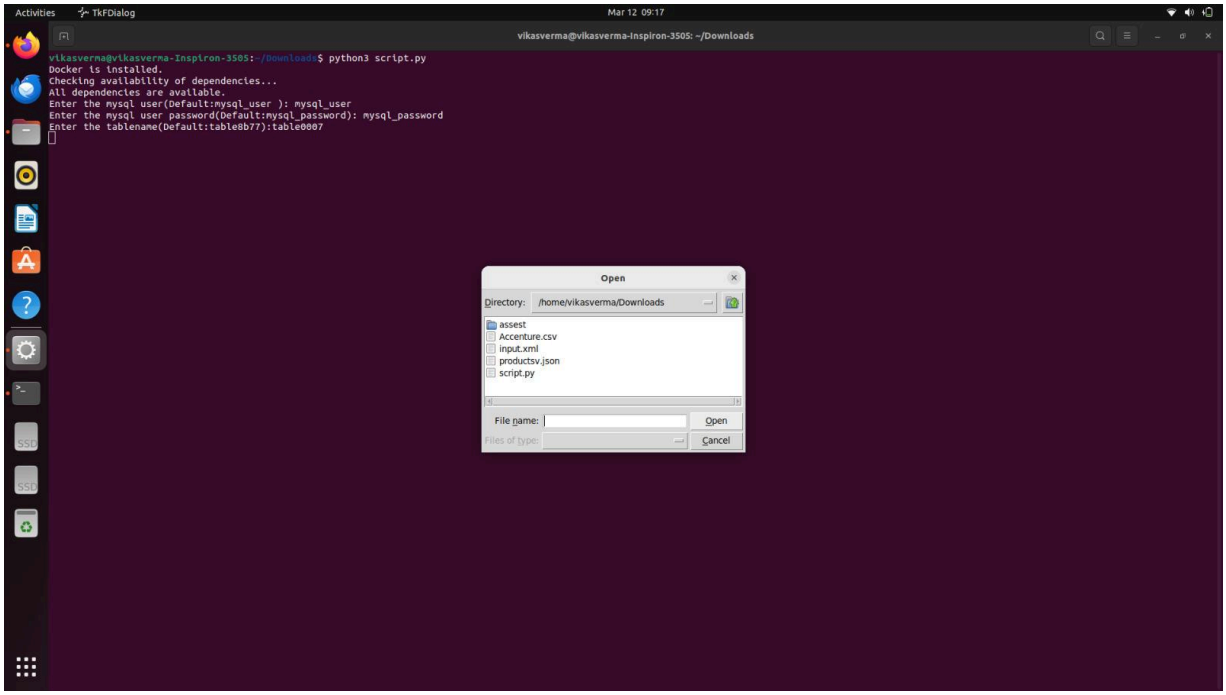


After the above code, the script will ask for ‘username’, ‘password’ and ‘tablename’, id you do not enter these values, it will take default values.

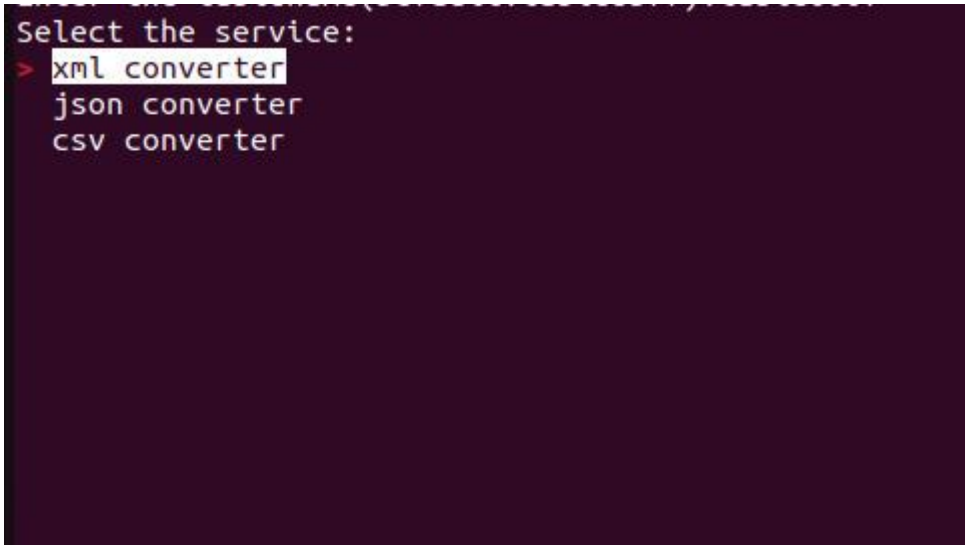




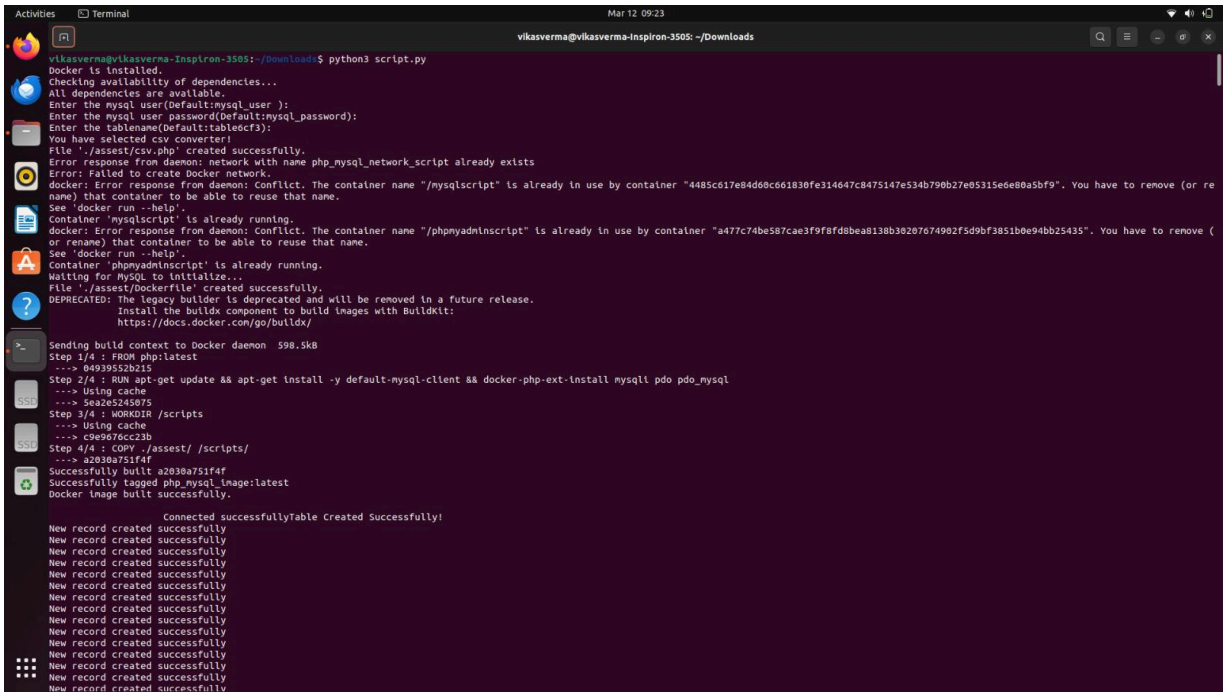
After this it will ask to select the file you want to convert:



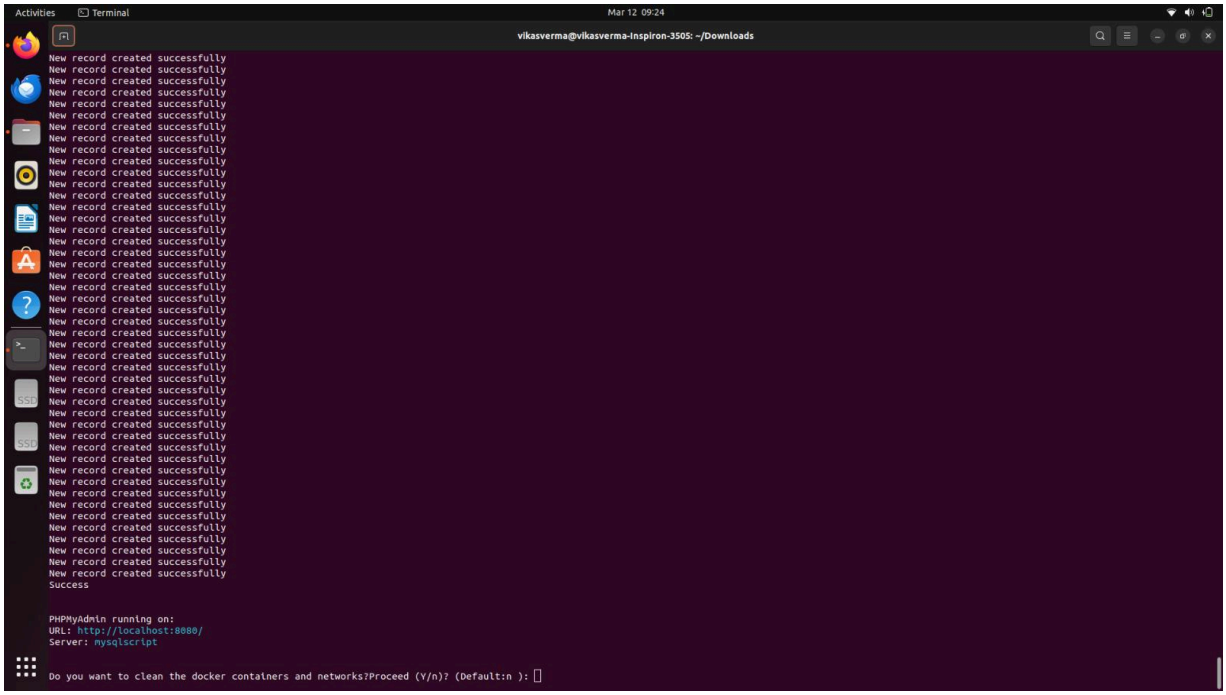
After selecting the file, select the type of converter as per the file.



After that the package will invoke the docker which eventually runs the container and other dependencies required for the script:

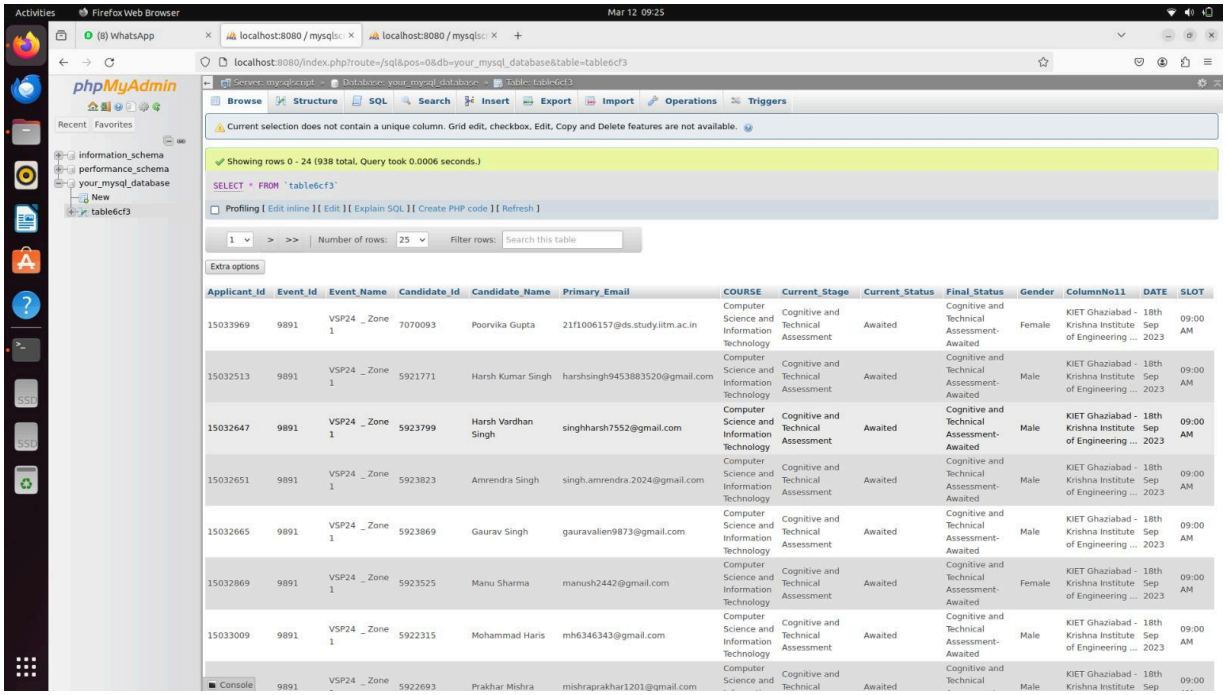


After successful execution of the script, it will show message “new record created” and it will provide the URL and server_name, click on the link:



After that it will ask to enter the servername, username and password , then it will redirect to the phpmyadmin homepage.

The created table will be seen in the select database , in the left corner:



5.3 BACKEND REPRESENTATION:

The backend of the model uses three things:

Docker:

Docker is a platform and tool for developing, shipping, and running applications inside lightweight, portable containers. Containers allow developers to package up an application with all the parts it needs, such as libraries and other dependencies, and ship it as one package. This ensures that the application runs on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

Container:

A container is a runtime instance of an image—what the image becomes in memory when executed (that is, an image with state, or a user process). You can start, stop, move, or delete a container using the Docker API or CLI. Containers are isolated from each other and the host system.

phpmyadmin:

phpMyAdmin is a free and open-source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting services.

Key features and capabilities of phpMyAdmin include:

1. Database Management: phpMyAdmin allows users to create, modify, and delete databases. It provides an interface to manage tables, columns, indexes, and foreign keys.
2. User Interface: It offers a user-friendly web interface that simplifies the process of managing databases, tables, and the data within them. Users can easily execute SQL statements, manage users and permissions, and import/export data in various formats.
3. Data Manipulation: Users can insert, update, or delete records within tables. It supports browsing data in tables, executing SQL queries, and searching within databases.
4. Import/Export: phpMyAdmin supports a wide range of formats for importing and exporting data, including CSV, SQL, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, LATEX, and others. This makes it easy to migrate data between different systems or to backup and restore databases.

5. Security Features: It includes capabilities for managing MySQL user accounts and privileges, ensuring database operations are performed securely. phpMyAdmin can also be configured to use SSL connections, enhancing security when managing databases over the internet.

6. Customization and Configuration: It offers various settings that can be customized, such as the appearance (themes), and the ability to configure servers, including connection settings and extensions.

7. Support for SQL: phpMyAdmin provides a robust interface for executing SQL queries, with features like syntax highlighting and query history. It can also generate graphical representations of the database structure.

8. Documentation and Support: phpMyAdmin is well-documented, offering a comprehensive manual and FAQs for its users. Being open-source, it also has a large community that contributes to its development and provides support through forums and other channels.

phpMyAdmin is typically installed on a web server, and users can access it through a web browser. Its installation requires a web server (like Apache, Nginx), PHP, and a MySQL or MariaDB database. It's widely used by web developers, database administrators, and hosting services for managing databases easily without needing to use command-line tools.

Python script:

Python is a high-level, interpreted programming language known for its clear syntax, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python has become one of the most popular programming languages in the world, used for a wide range of applications from simple scripts to complex machine learning algorithms.

Here are some key features and aspects of Python:

1. Ease of Learning and Use: Python's syntax is designed to be intuitive and similar to the English language, with a strong emphasis on readability. This makes it an excellent choice for beginners in programming.

2. Versatility: Python can be used for various types of programming, including web development, data analysis, artificial intelligence, scientific computing, and more. It's also used in system scripting and software development.

3. Open Source: Python is an open-source language with a vibrant community contributing to its development and a vast ecosystem of libraries and frameworks. This extensive support helps in solving a wide array of problems in different domains.

4. Cross-platform Compatibility: Python programs can run on multiple operating systems without requiring any changes to the code. It is compatible with major platforms like Windows, Linux, and macOS.

5. Extensive Standard Library: Python comes with a large standard library that includes modules and functions for various tasks, such as file I/O, system calls, and even internet protocols like HTTP and FTP.

6. Support for Multiple Programming Paradigms: Python supports different programming paradigms, including procedural, object-oriented, and functional programming, making it highly flexible in approach.

7. Community and Documentation: Python has a large and active community, which means it's easy to find help, tutorials, and documentation. The community also contributes to an expansive collection of libraries and frameworks, such as Flask and Django for web development, NumPy and Pandas for data analysis, and TensorFlow and PyTorch for machine learning.

8. Integrated Development Environments (IDEs): There are numerous IDEs and code editors available that support Python, making code development easier. Popular ones include PyCharm, Visual Studio Code, Jupyter Notebooks, and Spyder.

Python's simplicity and power have led to its widespread adoption across industries and academia. It is often used as a first programming language because it allows newcomers to quickly grasp the fundamentals of programming and move on to complex tasks.

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

Conclusion:

The NoSQL to RDBMS conversion project represents a significant step towards bridging the gap between flexible, schema-less NoSQL databases and the structured, transaction-oriented world of relational databases. By developing a comprehensive tool that automates the migration process, this project addresses a crucial need for organizations looking to leverage the benefits of RDBMS for certain applications, such as enhanced data integrity, complex query capabilities, and robust transaction support. The emphasis on user accessibility, security, performance optimization, and support for a wide range of database technologies ensures that the solution is practical, efficient, and adaptable to various use cases and requirements.

Future Scope

1. Advanced AI-Driven Optimization: Incorporating more advanced AI and machine learning algorithms to further refine schema translation and data mapping processes, potentially automating more complex decision-making aspects of the migration.

2. Cross-Database Synchronization: Expanding the tool's capabilities to support real-time data synchronization between NoSQL and RDBMS systems, facilitating hybrid database environments.

3. Broader Database Support: Continually updating the tool to support additional NoSQL and RDBMS platforms, responding to technological advancements and market trends.

4. Cloud Integration: Enhancing the tool for seamless integration with cloud-based database services, addressing the growing trend of cloud migration and distributed database systems.

5. Community-Driven Features: Developing an open-source version of the tool or incorporating community feedback into the development process to introduce features that address a broader range of user needs and scenarios.

In conclusion, the NoSQL to RDBMS conversion project can become an indispensable tool for organizations navigating the complexities of modern database technologies. By continuously evolving in response to technological advancements and user feedback, the project can expand its scope to offer more sophisticated, efficient, and versatile data migration solutions, catering to the ever-changing landscape of database management and data-driven decision-making.

REFERENCES

[1] Strauch, C., Martens, A., & Rumpe, B. (2017). "A Classification and Survey of NoSQL Databases". ACM Computing Surveys, 50(2), 1-38.

[2] Kaur, M., & Kumar, S. (2015). "Challenges in Data Migration from NoSQL Databases to RDBMS". International Journal of Computer Applications, 116(10), 1-7.

[3] Alomari, M. M., Lechner, G., & Gal, A. (2019). "Challenges of Data Migration Between NoSQL and SQL Databases." In Proceedings of the 2019 International Conference on Management of Data (COMAD), 145-152.

[4] Yu, H., Varghese, B., & DeWitt, D. J. (2016). "Efficient Data Conversion and Loading for NoSQL Systems". In Proceedings of the VLDB Endowment, 9(10), 684-695.

[5] Shah, S. H., & Patil, M. R. (2017). "NoSQL to SQL: A Review". International Journal of Computer Applications, 159(4), 23-29.

[6] Chen, J., & DeWitt, D. J. (2017). "Migrating to a NoSQL Database: An Empirical Study of Cassandra." In Proceedings of the VLDB Endowment, 10(12), 1879-1890.

[7] Zain Aftab, Waheed Iqbal, Khaled Mohamad Almustafa, Faisal Bukhari and Muhammad Abdullah (2020).” Automatic NoSQL to Relational Database Transformation with Dynamic Schema Mapping.” Hindawi Scientific Programming.

[8] V. Gour, D. S. S. Sarangdevot, J. R. N. R. Vidyapeeth, G. S. Tanwar, and A. Sharma (2010). “Improve performance of extract, transform and load (ETL) in data warehouse,” International Journal on Computer Science and Engineering, vol. 2, no. 3, pp.786–789.

[9] D. Skoutas and A. Simitsis(2006). “Designing ETL processes using semantic web technologies,” in Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP, DOLAP 06, ACM, New York, NY, USA, pp. 67– 74.

[10] M. Casters, R. Bouman, and J. Van Dongen, Pentaho Kettle Solutions (2010). Building Open Source ETL Solutions with Pentaho Data Integration, John Wiley & Sons, Hoboken, NJ, USA.

[11] S. Ramzan, I. S. Bajwa, B. Ramzan, and W. Anwar (2019). “Intelligent data engineering for migration to NoSQL based secure environments,” IEEE Access, vol. 7, pp. 69042–69057.

[12] A. K. Bhattacharjee, A. Mallick, A. Dey, and S. Bandyopadhyay (2013). “Enhanced technique for data cleaning in text file,” International Journal of Computer Science Issues (IJCSI), vol. 10, no. 5, p. 229.

[13] H. Mohamed, T. Leong Kheng, C. Collin, and O. Siong Lee (2011). “E-clean: a data cleaning framework for patient data,” in Proceedings of the 2011 First International Conference on Informatics and Computational Intelligence, pp. 63–68, IEEE, Bandung, Indonesia.

[14] B. Iqbal, W. Iqbal, N. Khan, A. Mahmood, and A. Erradi, “Canny edge detection and Hough transform for high resolution video streams using Hadoop and Spark,” Cluster Computing, vol. 23, no. 1, pp. 397–408, 202.

[15] A. Gupta, S. Tyagi, N. Panwar, S. Sachdeva and U. Saxena, "NoSQL databases: Critical analysis and comparison," 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), Gurgaon, India, 2017, pp. 293-299, doi: 10.1109/IC3TSN.2017.8284494.

[16] LI Jun-shan, LI Jian-jun(20, "Research on NoSQL Database Technology," 2nd International Conference on Management, Education and Social Science (ICMESS 2018).

[17] Ying-Ti Liao, Jiazheng Zhou, Chia-Hung Lu, Shih-Chang Chen, Ching-Hsien Hsu, Wenguang Chen, Mon-Fong Jiang, Yeh-Ching Chung, "Data adapter for querying and transformation between SQL and NoSQL database," 2016, Vol 65, pp. 111-121.

[18] Jing Han, Haihong E, Guan Le, Jian Du, "Survey on NoSQL database," 2011 6th International Conference on Pervasive Computing and Applications, 2011.

[19] Ágnes Vathy-Fogarassy, Tamás Hugyák, "Uniform data access platform for SQL and NoSQL database systems," 2017, Vol 69, pp. 93-105.

[20] Srdja Bjeladinovic, "A fresh approach for hybrid SQL/NoSQL database design based on data structuredness," Enterprise Information Systems, 2018, Vol 12(8-9), pp. 1202-1220.

[21] Sharvari Rautmare,D. M. Bhalerao, "MySQL and NoSQL database comparison for IoT application," 2016 IEEE International Conference on Advances in Computer Applications (ICACA), 2016