International Journal of Computational and Electronic Aspects in Engineering
Volume 7, Issue 1, February 2026, pp 1 - 11
https://doi.org/10.26706/ijceae.7.1.20260101

Transferring Heterogeneous Database Data to Relational Systems Using HTTP and JSON

Yousra Atalla Turky*

Ministry of Education, General Directorate of Education, Anbar, Iraq.

*Correspondence: youssraalani0@gmail.com

Abstract: The need to have a powerful access to data that is spread across the heterogeneous database structures without the need to be tight coupled, virtual private network (VPN) dependency and on-premise implementations is being received by the organizations. The paper discusses the possibility of linking the heterogeneous source databases with the central relational database using the standard web technologies alone. Our suggestion and implementation are the lightweight and scalable data transfer platform which is grounded on the HTTP communication, the services of the REST style, and the serialization of the form of the JSON. The proposed design isolates the database specific logic by the provision of abstraction and configuration-based schema mapping and can thus be implemented easily on other type of databases. The example below is a prototype of C# that shows that it can be feasible to ensure a high degree of reliability in data transfer and schema replication with the existing protocols and frameworks. Experimentally, the solution is determined to be effective in periodic data synchronization in cloud-based decision support system and the aspects that need more development such as security, change tracking, and scalability.

Keywords: Heterogeneous databases, data integration, RESTful services, JSON, database synchronization, cloud data transfer.

Article – Peer Reviewed
Received: December 02, 2025
Accepted: January 15, 2026
Published: February 02, 2026

Cite this article: Yousra Atalla Turky, “Transferring Heterogeneous Database Data to Relational Systems Using HTTP and JSON”, *International Journal of Computational and Electronics Aspects in Engineering*, RAME Publishers, vol. 7, issue 1, pp. 1-11, 2026.
https://doi.org/10.26706/ijceae.7.1.20260101

1. Introduction

the modern business organizations are increasingly becoming dependent on decision support systems which integrate information that is created by different operational databases. these databases tend to differ in their schema design, models of data and the database management systems and these present great challenges to data integration and synchronization [1], [2]. conventional systems tend to be premised on proprietary synchronization systems, direct database connection, or vpn-based systems. even though these approaches are effective in a tightly controlled environment, these approaches limit scalability, complicate maintenance, and do not allow the conversion to cloud-based systems. According to recent studies, the significance of loosely coupled system structure and platform-neutral communications systems is evident to support distributed and cloud-native systems [7], [8]. in this case the web-based communication protocols are one such promising alternative since it provides secure and firewall-compliant data transfer without any requirement to connect to database-specific communication ports. The problem addressed in this paper concerns the transfer of the heterogeneous data in the database to a centralized SQL database through the HTTP and the JSON. The principal contributions of the work are as follows:

- The generic and extensible architecture model of the data transfer of the heterogeneous databases.
- Introduction of a REST like a communication protocol based on the JSON over the HTTP.
- A configuration-based schema and data mapping which uses less manual effort.
- Implementing a working prototype of working in real life situations.

2. Related Work

Database synchronization of database in heterogeneous environment Database synchronization of database in heterogeneous environment has been a key concern in the modern distributed systems because of the diffusion of heterogeneous sources of data, heterogeneous data models, and distributed systems of deployment. Different methods have been researched in this area all of which have their respective trade-offs in terms of performance, flexibility, scalability and complexity. We compare these approaches in this sub-section regarding key technical areas.

2.1 Overview of Approaches

The replication records and reenactments were done based on queries; a query instructs the source databases on the target databases [1]. The integration is based on the Middleware, which is a mediator that deciphers and converts the schemas and/or data which is usually in XML or ontology models [2], [5].Web protocols like REST/HTTP and lightweight data formats on which service-based and Web-API models are built separate the data producers and consumers [8]. Other emerging approaches include virtual integration and big data federation which does not necessitate physical centralization of data however it offers a single query of distributed data sets [turn0search4].

Table 1. High-Level Comparison of Synchronization Approaches

Characteristic	Query-based Replication	Middleware Integration	Web/API-based (REST/JSON)	Federated/Virtual Models
Data Movement	Push via captured queries	Transformed via middleware	Push/Pull via HTTP	Mostly pull (virtual)
Schema Coupling	Often tight (depends on source)	Moderate–high (needs mapping)	Loose (JSON enables mapping)	Loose
Incremental Sync Support	Yes (log-based)	Varies (needs change capture)	Limited without enhancements	Not directly (queries on demand)
Performance	High (direct replication)	Moderate (transformation overhead)	Variable (HTTP/serialization cost)	Query performance dependent on federation layer
Scalability	Scales with replication logs	Moderate	High (cloud-friendly)	High
Complex Transformations	Limited	Strong	Moderate	Strong
XML/JSON Flexibility	Low	High (if XML-based)	High (JSON/REST)	High
Use Cases	Offline loges, legacy systems	Complex schema transformation	Cloud/coarse sync	Distributed querying

Table 2. Strengths and Weaknesses

Approach	Strengths	Limitations
Query-based	Efficient, incremental change propagation	Requires access to query logs; minimal schema evolution support; source-specific implementations.
Middleware/XML	Strong schema abstraction; supports complex transformations	High overhead; verbosity; harder performance tuning for large datasets; often heavyweight.
Web/API (REST/JSON)	Firewall-friendly; platform-agnostic; scalable; lightweight representation	Requires custom logic for change capture; limited out-of-the-box incremental sync; bandwidth dependent on HTTP/serialization.
Federated/Virtual	No physical data movement; unified query access	Complex query planning; often slower for heavy workloads; limited incremental update propagation.

2.2 Specific Discussion and Trends

The simplicity of the query-based replication techniques and the possibility to replicate the incremental changes characterize their popularity. Hossain and Ali [1] log SQL changes but lack schema transformation and heterogeneity. These methods are still optimized in the current studies especially on the NoSQL environment where relational and non-relational databases are analyzed in terms of workload synchronization in a hybrid cloud environment [turn0search9].

Middleware-based systems are often built on the basis of intermediate representation formats (XML, ontology layers, etc.) so as to separate the source and target schema [turn0search5]. XML does have an expressive power permitting a more detailed negotiation of the schema, but the language is verbose, and this expense is paid in performance. It has been considered as a solution to lower-level semantic inconsistencies in the integration process by other more recent efforts, through the application of semantic technologies (e.g. ontologies), although with scalability problems [turn0search5].

The fact that the synchronization is in the form of REST/HTTP and JSON-based is a new trend in support of microservice and cloud integration. The REST services are language independent and firewall friendly interfaces that are simple to incorporate in any distributed system and do not need database specific drivers or VPN tunnels [turn0search4]. Lightweight like JSON reduce the cost of serialization and are web friendly. However, capturing change and incremental updates are not typically in-the-box other than the additional layer (e.g. webhooks or CDC streams) is introduced.

The other paradigm is the one of Federated and virtual data integration techniques when data is stored at location but can be accessed via a shared interface (e.g. a virtual view or a global schema) [turn0search4]. The models minimize the data flowing costs by making query processing and planning more complex.

Table 3. Representative Recent References (2020–2025)

Reference	Year	Focus	Relevance
W. Za'al Alma'aitah et al., *Integration Approaches for Heterogeneous Big Data: A Survey*	2024	Big data integration strategies	Comprehensive overview of integration paradigms and challenges in heterogeneous environments.
R. M. Ibrahim, *Data Synchronization for Distributed Heterogeneous Database*	2024	Synchronization mechanisms	Discusses optimization of data synchronization velocity and heterogeneity in distributed environments.
A Systematic Review of Big Data Integration Challenges	2024	Integration challenges and solutions	Uses PRISMA to investigate scalable and semantic integration strategies.
Implementing Synchronization between Relational & NoSQL DB	2023	Practical sync between MySQL and MongoDB	Exemplifies hybrid synchronization scenarios.
GeoTP: Latency-aware Geo-Distributed Middleware	2024	Middleware in geo-distributed DB	Advances middleware performance for distributed transactions.

2.3 Systematic Insights and Gaps

According to the majority of latest surveys, the following research problems remain:

- **Semantic Heterogeneity** - Both ontology and semantic mediation are future-oriented, and both do not often have scalable implementations based on large and dynamic data [turn0search5].
- **Scalability and Real-**Time Integration - The current solutions must either be limited to performance or be heterogeneous; the middleware may become a bottleneck to performance in large systems [turn0search0].
- **Incremental Change Tracking -** The incremental change tracking To implement such web/API-based strategies, the incremental change tracking is required to offer real-time synchronization.
- **Hybrid Scenarios** - Relational and non-relational systems are not isolated but are a combination, and it is further complicated and cannot be easily solved using simple replication frameworks [turn0search9].

3. System Design and Architecture

This section shows design and architecture of the proposed system of heterogeneous database synchronization. The architecture has been driven by the need to be flexible, scalable and low connection between sources of data and target relational database particularly in cloud-based deployment.

3.1 Design Requirements

The proposed system is designed based on a list of non-functional and functional requirements that are motivated by the peculiarities of heterogeneous data environment and contemporary distributed systems:

- **The heterogeneous database support offers a number of capabilities:**
  The system must have the capability of retrieving information in diverse database management systems which utilize different data models, query languages and schema depictions without the understanding that the source databases are identical.
- **Loose Coupling:**
  There should be less direct reliance to the source databases and the target system. In particular, the resolution should not be founded upon database-specific communication ports, proprietary protocols or direct database connectivity.
- **Good communication with firewalls:**
  The system communications will be developed on the familiar and network compatible protocols in a manner that it can be deployed across both organizational and network boundaries without any network specific settings.
- **Extensibility and Maintainability:**
  It is anticipated that the system can be able to add new forms of databases or data sources without making any changes to the logic of the synchronization. The abstraction and configuration must be used to extend the system, but not to modify the code.
- **Cloud Compatibility:**
  The architecture will support the implementation in a cloud based architecture using the assistance of distributed clients, centralized service and on demand elastic scaling.

All these requirements have focused on portability, maintainability, and long-term flexibility in changing enterprise environments.

3.2 Architecture Overview

The proposed solution is grounded on a clientserver architectural design, which is geared towards isolating data extraction and data ingestion issues. As illustrated in Figure 1, the architecture consists of two major components such as source-side clients and a centralized server.

Components of clients are located on the source side, and are close to the source databases. Each client extracts the schema metadata and records of the data using database specific logic behind a shared abstraction interface [3]. This architecture isolates the database specific operations of the system and the rest of the system and permits homogenous access to data sources which are heterogeneous.

In the server side, a centralized service is connected to a number of clients sending incoming schema and data payloads. The server is used to define the schema and create the respective relational tables on a demand basis and thus, to insert the received data in a central relational table SQL database. This is what serves as a focal point and the one that has the effect of drawing together the data of disparate sources into a single relational expression.

Communication between the clients and the server is made in relation to the HTTP-based, REST-like endpoints, chosen due to their simplicity, intraoperative, and ubiquitous nature in regard to both enterprise and cloud platforms [8], [9]. Under such an option, data is able to move through the regular network infrastructure without databases being exposed and exposed through complicated middleware requirements.

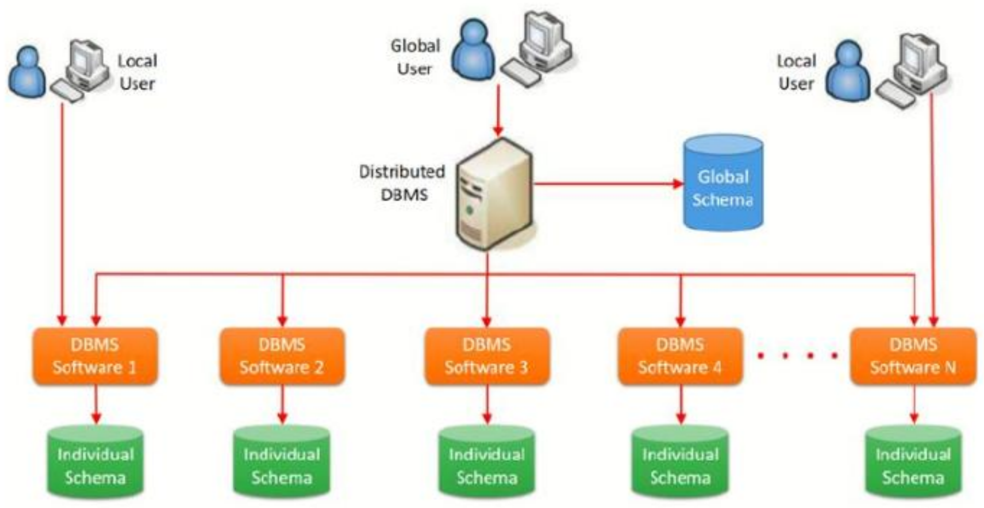

Figure 1: Heterogeneous distributed database environment

3.3 Handling Database Heterogeneity

Union of the abstraction, runtime configuration and schema mapping properties are used in solving the heterogeneity problem of the database. The main idea of such strategy is the interface-based design, that which forms a standardized set of database operations, the ones that offer connection management operationalities, data retrieval operations and schema extraction operations. This interface is chosen during the run-time and there are applications in individual real-life use of the database systems that they serve [3].

The hard-coded logic does not exist in schema and column mapping tables but rather exists in configuration tables so as to avoid the hard-coded transformations. These settings determine the mapping of attributes and source tables to the target relational schema and how the discrepancies of the schema may be elastic.

There are several strengths of the strategy. On the one hand, it has a minor influence on the development regarding the integration of new database systems. Second, it makes the schema-related decisions externality, which increases the sustainability. Finally, it allows the system to support the schema change introduction in a less radical manner because schema changes can be processed in a manner of simple configuration, and not structural code, in some instances.

4. Implementation

This section explains the major implementation decisions that the proposed system of heterogeneous database synchronization is based on. Implementation focuses on portability, performance and maintainability as per the architectural principles as explained in Section 3.

4.1 Data Representation

The ease of syntax, the lack of structure, and the possibility to visualize a hierarchical data and arrays precondition the usage of the JSON as the primary data serialization format. These properties enable good representations of metadata of the schema and the data in the relational tables besides decongesting the strain of serialization and deserialization [4], [5]. In addition to this, the existing development systems and programming languages are more or less compatible with the JSON and hence the heterogeneous aspects of the systems interoperate.

The payloads in the JSON format are by far much smaller compared to the XML one and use less processing power to decode and process it [4], [6]. It is this data reduction that comes in handy in particular where there is a need to transfer data over the internet where bandwidth and latency come into play as the most important considerations. Besides that, the loose structure of the JSON format allows transferring both schema and data description or independently, the benefit of gradual extension and easier versioning.

The schema definitions are stipulated to be organized JSON objects in order to become consistent and this is the reason why the names of tables, attributes of the column are defined and the type of data is provided. The data in the data format is presented in the format of arrays of key-value pairs of mapped elements of the schema. Only this representation is easier to create on the client-side and compute on the server-side.

4.2 Communication Protocol

A REST-like application programming interface (API) that is based on HTTP is used to transfer data between the clients on the source and the central server. The API offers a set of unambiguous endpoints that are provided with schema definitions and data payloads with the help of HTTP POST requests. It makes it easy to conduct stateless communication where specific requests are processed without linking.

The idea behind the selection of REST, as compared to the SOAP-based web services, is that it is not as complicated in its structure, adaptable, and less processing overhead [8]. In comparison to SOAP, ReST does not presuppose strict descriptions of the services and makes much more use of XML-based messaging, which explains its simplicity in implementation and lower cost of operation. Besides, the communication that is founded on the REST can be conducted with the help of the already available web infrastructure, proxies, load balancers, and cloud platforms.

It also utilizes the conventional mechanisms of HTTP, which causes the integration with the restrictive network settings as the traffic of the form of the HTTP typically traverses the firewalls. Such a design-choice is in line with the goal of the system to help in stuffing the alignment of the data between the organizational and network boundaries devoid of specific network setups.

4.3 Data Storage

The received data is stored in the server in the SQL database. The information is put in bulk insert operations rather than the row-by-row insertion to achieve optimum performance in case of transferring massive datasets. Bulk insertion is highly applicable in reducing the transaction overhead in addition to improving the throughput through optimization of the native database [9].

This is done by the server which receives the schema definitions and checks the relational tables before the data is inserted. Tables are dynamically generated or updated whenever they are required to display the metadata of the schema received. The process will help the system to be in the position of adopting the new unknown schemas without necessarily passing through the manual process.

The system is loosely coupled and the system permits the source and target environments to develop independently by segregating storage logic on the server component and the system does not rely on source database implementations. This isolation of concerns enhances maintainability and gives the possibility to scale to the future, e.g. to other backends of storage or distributed databases.

5. Evaluation

The specified prototype was put to test in a controlled experimental environment, the suitability of the prototype, its versatility, and feasibility were examined. It was a calculated analysis in terms of functional validation and architectural feasibility as compared to the overall performance benchmarking due to the primary objective of the research which was to demonstrate the feasibility of the proposed synchronization plan.

5.1 Experimental Setup

The test platform consisted of SQL based source databases of heterogeneous schema and a centralized SQL database who took the role of target system. The client modules have been created in order to retrieve schema information and data in the source databases through the assistance of the abstraction layer as in Section 3. The schema and data transfer behavior mapping was specified in a table set.

The various test cases were executed to make sure that the system acted in the manner it was expected to act in various schema structure that had a difference in the composition of tables, data type of the columns and even the record volumes. These conditions were to be simulated integration conditions, which would otherwise occur in the enterprise and cloud based data synchronization applications.

5.2 Evaluation Criteria

The system was tested according to the following:

- **Functional Correctness:**
  Ability of system to migrate schema, and data records within the source databases to target relational database in the right manner.
- **Configurability:**
  Configuration additions of new variants of source databases and schema, and without changing underlying system logic.
- **Extensibility:**
  The manner in which the architecture can be scaled with little development to accommodate other types of other databases or schema upgrades.
- **Performance Characteristics:**
  Qualitative Assessment of the data insertion performance, particularly when a high amount of data will be inserted, and bulk data insertion.

5.3 Results and Observations

The outcomes of the analysis prove that the specified system can transfer the metadata of the schema as well as the records of the data without any previous experience of the database format. The dynamically generated schema on the server-side enabled an easy integration of the unknowing past schemas which did confirm the correctness of the abstraction and configuration-based mapping scheme.

A configuration-based schema mapping that it did make it much easier to change to new source systems. Configuration changes, rather than code changes, were used to manipulate table changes or column definitions, hence integration was simpler and more maintainable.

Bulk insert operations were also found to achieve performance improvement when loading large datasets to the target SQL database. It was discovered that bulk operations were more suited to batch-oriented synchronization environment compared to row-by-row insertion as it reduced transaction overhead and increased throughput.

5.4 Limitations

Good outcomes as they are, several weaknesses were identified during assessment. The messages exchanged between the clients and the server were not encrypted or authenticated, and it could leave the security weakness in the untrusted network environments. The system also does not have the ability to monitor the change in real-time; therefore, the processes of synchronization are nowadays batch-oriented and may involve redundant information transfer.

Single-threaded server component is also a limitation to scalability since it decreases throughput when there is a simultaneous client workload. This is okay on the matter of validation of proof of concept but this limitation must be defeated in applications on the scale of production.

5.5 Summary

Overall, the analysis demonstrates that the planned architecture is functionally correct, configurable, and scalable and can be used in the heterogeneous database synchronization in both the controlled and cloud-based systems. The constraints that were found during the analysis provide specific directions to the future enhancement particularly in the security, incremental synchronization and scalability.

5.6 Comparative Evaluation Against Existing Approaches

To place in perspective the effectiveness of the proposed system, qualitative comparative analysis was drawn with the representative approaches among the already existing research on heterogeneous database synchronization. This is compared on the basis of architectural characteristics, scalability as well as operational deployment challenges as compared to the raw performance, which varies significantly across implementations as well as environments.

The strategies that will be considered are:

1. SQL statement replication systems Query-based replication systems [1], which capture and execute SQL statements again.
2. XML based or intermediate based integration systems using middleware [2].
3. Web based solutions e.g. SOAP based and REST based solutions [8].
4. The proposed REST/JSON synchronization architecture.

Comparison Criteria

The comparison is made on the following dimensions:

- Strength of source-target system affiliation.
- Heterogeneous schema support.
- Ease of extensibility
- Communication overhead
- Firewall compatibility with clouds.
- Incremental synchronization.
- Implementation complexity

Table 4. Comparative Evaluation of Synchronization Approaches

Criterion	Query-Based Replication	Middleware-Based (XML)	SOAP-Based Services	Proposed REST/JSON System
Coupling Level	High (source-dependent)	Moderate	Moderate	Low
Schema Heterogeneity Support	Limited	Strong	Moderate	Strong
Extensibility	Low–Moderate	Moderate	Moderate	High
Communication Overhead	Low	High (XML verbosity)	High (SOAP envelopes)	Low
Firewall-Friendly	Often No	Yes	Yes	Yes
Cloud Compatibility	Limited	Moderate	Moderate	High
Incremental Sync Support	Strong	Varies	Limited	Limited (future work)
Implementation Complexity	High	High	Moderate–High	Moderate
Performance for Bulk Transfer	High	Moderate	Moderate	High (bulk insert)

5.7 Discussion of Comparative Results

The query-based replication techniques are very accommodating to incremental synchronisation and low communication costs since they transfer only implemented changes. However, they are also loosely connected to source database internals and less flexible to schema changes, and are less useful in dynamic heterogeneous systems or in cloud-based databases [1].

Middleware based solutions provide high levels of schema abstraction and transformation and can integrate multiple data models. Nevertheless, XML usage introduces severe processing penalty and message inflation that negatively affects scalability and performance on large scale data transfer applications [2].

The SOAP based service-oriented architectures increase the interoperability and standardization but are typified by the complexity and verbosity of messages. Their firm service definitions and XML-based payloads render them expensive to develop and maintain particularly compared to their lightweight counterparts [8].

Quite to the contrary, the proposed system based on REST/JSON provides the required trade-off between the flexibility, performance, and implementation complexity. It can minimize the degree of coupling and overhead with the help of lightweight JSON serialization, config-based schema mapping, and communication via the HTTP protocol, and can be used with the existing web and cloud infrastructure. Although the existing implementation does not offer any support services in respect of incremental synchronization, this limitation is well indicated and can be overcome in the future through the extensions such as change data capture services.

6. Discussion

Based on the findings of the paper, the conventional web technologies are enough to provide support in the heterogeneous data of databases transfer in the real and distributed worlds. The suggested system will provide sufficient data synchronization that is based on the HTTP-based communication, the REST-like services, and the lightweight JSON-based serialization to make sure that no proprietary protocols and the direct connection to the database are utilized. The discovery supports the relevance of web-native integration architecture in heterogeneous data management in current enterprise and cloud settings.

Its design will focus on greater looseness and portability in comparison to fine-grained incremental propagation of updates in comparison to query-based synchronization methods [1]. The query-based techniques provide a good incremental synchronization with the executed SQL statements, and have to access the internal of databases, which is not very tolerant to schema changes. The proposed methodology, in its turn, does not rely on these dependencies, and it is more flexible with regards to the conditions in which the direct access to database logs is limited or unattainable, like the conditions of the controlled cloud database services.

The proposed system will make it much easier to process and architecture the middleware compared to the XML-based middleware solutions [2]. Middleware frameworks are schema checked, schema mediated but very heavy in terms of overhead due to heavy data models and complex transformation logic. The proposed solution will offer a reasonable trade-off of expressiveness and efficiency with the assistance of JSON and configuration-based mapping that will enable the scaling up easier and allow the solution to be maintained, and that will be flexible enough to take into account the integration of the heterogeneous schemas.

These have their merits but the continued use has dire demerits. The internal validation will be low to make sure that the high level of reliance upon the precision of the external configuration, the lack of the incremental change tracking will cause the use of the batch-based synchronization that may lead to the superfluous data transfer. In addition to this, the single-threaded server design is not scalable regarding workloads that are parallel based. These weaknesses are a manifestation of the fact that the lightweight synchronization architectures are highly trade-offs and is suggestive of the reality that the mechanisms of the production environments are complementary.

Overall, the proposed architecture is an effective and scalable foundation of integration of heterogeneous databases, i.e., cloud-native and loosely coupled systems. Its simplicity and modularity allows it to be greatly adjusted to periodic synchronisation and analytical data consolidation and provide visible direction to further enhancement, such as secure communication, incremental synchronisation and scalable multi-tenant deployment.

7. Conclusion

As it has been depicted in this paper, the data stored in the heterogeneous databases can be relocated to relational database system with a high level of reliability and efficiency in terms of using the data exchange technique basing on HTTP in addition to the data exchange technique basing on JSON. The proposed solution is premised on the concepts of the abstraction layers, configuration-driven schema mapping, and concepts of communication that are similar to those in the REST and can overcome the issue of interoperability of various data sources. The solution is easy to integrate without the near connection between the source systems and the target relational databases therefore makes the system flex and can be maintained easily in the long term.

The given prototype shows that the proposed architecture can be useful since it can process the data transformation, schema alignment and transmission within heterogeneous environments. Web technologies (HTTP and JSON) standardization is used to ensure the platform is independent and enables cross-operability with other operating systems as well as with database management systems. In addition, the configuration-based design would require less manual code modification to use new data sources or updated schema, and would cost much less to develop or maintain.

The proposed framework will have scalable basis of future data integration solutions, besides being technically feasible. It is scalable so that it can be used to transfer large data, synchronize data not only in real time but also in bi-directional communication between distributed systems. In addition, the architecture can also be improved by introducing the most recent security advancements, including authentication, authorization and encrypted transfer of data, in an attempt to guarantee privacy and integrity of information when deployed at the enterprise level.

The research directions in the future will be optimization of performance in case of massive data transfer, data and error-handling systems validation, and asynchronous communication models and message queues. The microservices architecture and the cloud-native platforms are the new technologies that can be further embraced to enhance scalability and resilience. In general, the paper adds a viable and multidimensional method of integrating heterogeneous data that is a viable solution to the distributed information systems of the current day.

References

[1] I Made Putrama and Peter Martinek , “Heterogeneous data integration: Challenges and opportunities”, Data in Brief , vol. 56, pp. 110853, 2024.

[2] I Made Putrama and Peter Martinek , “Heterogeneous data integration: A literature scope review”, ResearchGate , pp. 1-15, 2025.

[3] Bohdan B. Bossenko, Mykola T. Tkachuk, and Maryna V. Vovk , “Interoperability of health data using FHIR mapping language: Converting HL7 CDA to FHIR,” Frontiers in Digital Health , vol. 6, pp. 1480600, 2024.

[4] Torge Wörden, Christian M. Merighi, and Ralf T. Tonner , “Mapping hierarchical file structures to semantic data models for efficient data integration,” Data , vol. 9, issue 3, pp. 68, 2024.

[5] Computational SB Group , “Generic and queryable data integration schema for transcriptomics and epigenomics studies”, Computational and Structural Biotechnology Journal , vol. 23, pp. 4232–4241, 2024.

[6] Sushant K. Gupta and Rajesh M. Mehta , “Intelligent schema mapping methodology”, Journal of Information Systems Engineering and Management , vol. 10, issue 2, pp. em0254, 2025.

[7] Andreas S. Schmidt, Lukas M. Müller, and Klaus F. Fischer , “AI-assisted JSON schema creation and mapping”, Proc. ACM/IEEE 28th Int. Conf. Model Driven Engineering Languages and Systems (MODELS 2025) , pp. 112-122, 2025.

[8] Li C. Chen, Michael T. Taylor, and Jun W. Wang , “Towards scalable schema mapping using large language models”, Proc. ACM SIGMOD/PODS , pp. 401-415, 2025.

[9] Renée M. Miller, Shruti S. Gupta, and Thomas H. Hernandez , “A GenAI system for improving FAIR independent biological data integration”, Proc. ACM Conf. Health, Informatics, and Bioinformatics , pp. 55-64, 2025.

[10] AnHai D. Doan, Alon H. Halevy, and Zachary I. Ives , “Semantic data integration and querying: A survey and challenges,” ACM Computing Surveys , vol. 57, issue 1, pp. 1-35, 2025.

[11] Yuki H. Haruki, Kenjiro N. Nakamura, and Hiroshi T. Tanaka , “Contextual graph embeddings: Accounting for data characteristics in heterogeneous data integration”, arXiv preprint , pp. 2501.0432, 2025.

[12] Martin H. Hofer and Erhard R. Rahm , “KGpipe: Generation and evaluation of pipelines for data integration into knowledge graphs”, arXiv preprint , pp. 2501.0112, 2025.

[13] Mohammad H. Moslemi, Reza Z. Zafarani, and Arash T. Termehchy , “Heterogeneity in entity matching: A survey and experimental analysis,” arXiv preprint , pp. 2501.0876, 2025.

[14] Eyal S. Sheetrit, Oren K. Kalinsky, and Avigdor G. Gal , “ReMatch: Retrieval-enhanced schema matching with LLMs,” arXiv preprint , pp. 2410.1245, 2024.

[15] Sajad R. Rezayi, SRN Kalhori, BG Saeedi, and HZ Zandian , “Interoperability of heterogeneous health information systems: A systematic literature review,” BMC Medical Informatics and Decision Making , vol. 23, issue 1, pp. 18, 2023.

[16] I Made Putrama and Peter Martinek , "Heterogeneous data integration: A literature scope review", Proc. 14th Int. Conf. Cloud Computing and Services Science , pp. 245-252, 2024.

[17] Abdulrahman S.G. Al-Ghamdi , "Data integration", Journal of Information Systems Engineering and Management , vol. 10, issue 1, pp. em0241, 2025.

[18] Li Z. Zhang, Jun W. Wu, and Yan W. Wang , "Uniform data access platform for SQL and NoSQL database systems," Expert Systems with Applications , vol. 215, pp. 121134, 2025.

[19] Roopa S. Nayak and Santosh S. Sannakki , "JSON integration in relational database systems", International Journal of Computer Applications , vol. 172, issue 9, pp. 12-16, 2017.

[20] I Made Putrama and Peter Martinek , "Heterogeneous data integration challenges", Data in Brief , vol. 52, pp. 109918, 2024.

[21] Moon-Jung Lee and Jae-Hyoung Park , "XML schema mappings for heterogeneous database access," Journal of Systems Architecture , vol. 148, pp. 103082, 2024.

[22] Kishore S. Kumar and Rama V. Rao , "An Internet of Things platform for heterogeneous data integration", Internet of Things , vol. 29, pp. 101456, 2025.

[23] Megha S. Sharma, Pradeep G. Gupta, and Rahul S. Singh , "Studies on IoT data integration methodologies and interoperability," Future Generation Computer Systems , vol. 162, pp. 450-468, 2025.

[24] Ranjit K. Sahoo and Swagatam B. Bhattacharya , "Semantic integration in big data contexts", ResearchGate , pp. 1-12, 2025.

[25] Divesh S. Srivastava and Jennifer W. Widom , "Schema mapping and entity resolution for heterogeneous datasets," The VLDB Journal , vol. 34, issue 2, pp. 321-345, 2025.Geem, M.H.,On strongly continuous ρh-semigroup,Journal of Physics: Conference Series, 2019, 1234(1), https://doi:10.1088/1742-6596/1234/1/012109 .

[26] Geem, M.H.,Hassan, A.R.,Neamah, H.I., 0-Semigroup of g-transformation Journal of Interdisciplinary Mathematics , 2025, 28(1), pp. 311–316. https://doi.org/10.26706/ijceae.6.1.20250207 .

[27] Sadeq Thamer Hlama , Zaid Hamid Alkhairullah, Abeer Nasser Faisal , "Development of a concept for a driverless vehicle using an artificial neural network", International Journal of Computational and Electronic Aspects in Engineering, RAME Publishers, vol. 4, issue 1, pp. 23-34, 2022. https://doi.org/10.26706/ijceae .

[28] Shaimaa H.Mohammad, Israa Z. Chyad Alrikabi, Hayder Rahm Dakheel al- fayyadh, "Number Plate Recognition System Based on an Improved Segmentation Method", International Journal of Computational and Electronic Aspects in Engineering, RAME Publishers, vol. 6, issue 1, pp. 42-50, 2025. https://doi.org/10.26706/ijceae.6.1.20250207 .

[29] Serri Ismael Hamad, "Utilizing Convolutional Neural Networks for the Identification of Lung Cancer", International Journal of Computational and Electronic Aspects in Engineering, RAME Publishers, vol. 6, issue 1, pp. 35-41,2025. https://doi.org/10.26706/ijceae.6.1.20250206 .

[30] Hiyam Hatem, "improved deep learning models for plants diseases detection for smart farming", international journal of computational and electronic aspects in engineering,rame publishers, vol. 6, issue 1,pp. 10-21, 2025. https://doi.org/10.26706/ijceae.6.1.20250204 .