International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Transformation-Based Data Migration Method for Public Sector's through Relational Database

Varun Varma Sangaraju

Submitted:15/05/2024 **Revised**: 28/06/2024 **Accepted**: 07/07/2024

Abstract: In this work, a procedure for converting a relational database (RDB) into an XML document is proposed. Database migration is the process of moving schema and data from a source RDB towards the destination database of XML script, which makes to achieved and moved through new context. The home schema is semantically enhanced and translated into a target schema, and the data in the source database is transformed into a target database based on the new schema. The semantic enrichment procedure is required to create an improved metadata model from the source database and captures key elements of the destination XML schema, making it appropriate for turning RDB data into an XML document. Algorithms are designed for constructing the target database based on a set of migration rules to translate all RDB constructions into an XML Schema, from which RDB data is subsequently transformed. A prototype system has been developed and experimentally assessed by testing its outcomes, looking at our accomplishments, and commenting on the findings. The recommended remedy is found to be effective and accurate after the review of the outcomes.

Keywords: *relational database, XML document, database, metadata, data transformation.*

I. Introduction

Since XML has established itself as a norm for sharing and disseminating RDB data via the Web, it is often utilised for e-technologies and non-traditional applications, such as multimedia, GIS, and CAD software, among others [1]. Additionally, this new technology has dominated the field of information systems because to its productivity, adaptability, and support for a wide range of unique ideas, which has allowed it to meet the needs of sophisticated applications that need a variety of rich data kinds. However, a significant amount of data is kept in RDBs, despite their limits in supporting sophisticated structures and user-defined data types given by the relatively new XML. These days, XML serves as both a content-level database and a hypertext-level standard [2]. As a result, rather than discarding the enormous quantity of data held in RDBs, it is preferable to enhance and transform such data so that it may be utilised by modern systems.

Because it extends simple user-defined tags to additional layers with intricate structures and connections like aggregation and inheritance, XML is a potent paradigm. In addition, XML Schema language is a standard that offers a complex way to describe the structures and restrictions of XML schema and instance documents [3]. It incorporates ideas from object-based models, like inheritance, references, data collections, and user-defined data types, as well as RDB models' characteristics like integrity restrictions. The limits and issues with XML-enabled RDBs in managing XML documents have resulted in the emergence of native XML databases to manage XML documents, such as XML-Spy and eXist-db. This is due to the growing relevance of XML.

The majority of RDB to XML conversion techniques now in use concentrate on creating a document type definition (DTD) schema. Because the XML Schema standard has acquired widespread adoption in recent years, database migration solutions must produce target databases in accordance with this standard.

We provide a method in this article for creating an XML document from an existing RDB. The initial step in this conversion is to enrich a source RDB schema semantically by acquiring as much metadata information as possible and to produce an enhanced metadata model known as the Canonical Data Model (CDM) [4], which encapsulates essential characteristics of the target XML schema. The CDM then directs the conversion of RDB schema and data into the target XML database, ensuring that the conversion process is carried out with data integrity and consistency. In more depth, we show a set of translation rules that are built into algorithms. These rules translate all of an RDB's structures into an

Independent researcher and senior engineer, Dallas, TX, USA.
varunvarma93@yahoo.com

XML Schema, which is then used to turn the data. We created a prototype to test the algorithms and prove the idea, which outputs an XML document. A source RDB and a target XML document produced by the prototype were compared in two studies to see how they varied. The outcomes reveal that the source and destination databases are identical, and they also prove that the suggested approach is technically sound and accurate conceptually [5]. Our technique is more efficient than previous alternatives because it generates XML schema and data depending on user input and takes use of the sophisticated capabilities given by the XML Schema standard. The resulting XML documents might be useful for disseminating and exchanging business data across disparate platforms.

The paper is organized as follows, the existing literature is discussed in section 2. The enhancement of relational database is discussed in section 3 and the performance analysis based on the proposed one is detailed in section 4. Finally the proposed system is being concluded in section 5.

II. Literature Survey

Because of the widespread use of XML in online commerce, the conversion of RDBs to XML databases has become increasingly significant [6]. There are two methods for converting RDB to XML. The first method handles data stored in RDBs through XML interfaces, while the second method migrates an RDB into an XML database. The first method focuses on schema translation, while the second method entirely migrates both the schema and the data into the destination database.

Existing work on converting RDBs into XML documents imposes certain requirements and makes specific assumptions to simplify the conversion process, which may be a source of constraints or vulnerability. Various approaches, for example, employ data dictionaries and presume well-designed RDB [7], whilst others use older RDB for migration into XML documents [19]. Additionally, the generated XML schemas might be a DTD [7], XML Schema [19], or another standalone XML language [4]. Nevertheless, a number of scholars have suggested approaches for converting UML class diagrams to XML [8]. We have provided a comprehensive analysis of translation options for different directions, as they pertain to database conversion, in [9].

[11] Describes a suggested way to change an RDB to an XML flat file. However, the conversion process doesn't take into account the RDB's logical limits. Data semantics are abstracted from an RDB schema into an EER model and then mapped into an XSD graph according to a technique developed by Fong and Cheung [12]. In this method, foreign keys are put into a structure of elements and sub-elements, which can lead to duplication when an element is related to more than one other element. [13] Provided a technique for creating an XML document from historical RDBs utilising the ER model as an intermediary step; however, inheritance and aggregation relationships are not taken into account. A mapping technique between RDB tables and DTD components was proposed [14]. However, the technique does not make use of XML model characteristics or take into account integrity restrictions.

RDBs may also be published as XML documents using certain declarative languages so they can be transferred over the Web. SEML [10], XPERANTO [3], and XTABLES [11] are examples of systems that use this technique. Transforming a relational database into XML exposes views that can be queried using XML query languages [15]. The outcomes of these apps are completely materialized in XML, but the data inside them is not. Furthermore, modifying the object view for expressing XML data in an RDB encounters constraints such as data collection representations and tag naming. The SEML is an interpreter with a markup language for mapping RDBs into XML documents [16]. When used with (object-) RDBs, the XPERANTO converts XML-based queries into SQL [17]. In order to produce XML documents, the system deconstructs SQL queries. However, inconsistencies between XML and SQL query syntax exist, and integrity constraints are not considered precisely [18]. XTABLES offers a query language that may be used to query and store XML documents in RDBs [19].

Research into the conversion of RDBs into XML is still in its infancy, as shown by our review of the literature, and as a result, there are a number of areas that need to be prioritised for future work. We have observed that the majority of XML model migration studies have employed source-to-conceptual-to-target methodologies, concentrating on producing a DTD schema and/or data. Certain semantics, such as inheritance and aggregation relationships, are not

taken into account in certain works. This is primarily due to their absence of support for such semantics in source or target data models, such as ER model and DTD's lack of inheritance support. The XML Schema supports a far wider range of data types than DTD and gives strong methods for attributes and elements' referencing, nesting, and inheritance.

III. Enhancement Of Relational Database

Beginning with the extraction of RDB's metadata, which includes relation names, attribute characteristics (such as attribute names, data types, and whether or not the attribute allows null values), Primary Keys (PKs), Foreign Keys (FKs), and Unique Keys (UKs), semantic enrichment of RDB begins. For this work, we used our technique [20] for enriching RDBs. We assume that data dependencies are represented by PKs and FKs, and that for each FK value, there is an existing, matching PK value.

The procedure' last stage is locating and creating the CDM and its associated constructs using a categorization of the relatedness, traits, and relationships.

Based on CDM, According to our categorization system, classes fall into the following groups:

- Regular Strong Class (RST): a class without any FKs in its PK.
- Secondary Strong Class (SST) is an inherited RST.
- Subclass (SUB): A class that derives from a super-class but is not descended from by other subclasses.
- Secondary Sub-class (SSC): A sub-class that other sub-classes inherit from.
- Secondary Relationship Class (SRC): a referred RRC class, a M:N relationship class with characteristics, or n-ary relationships, where n is greater than 2.
- A regular Component Class (RCC) is a weak class that interacts with other classes as opposed to its parent class.
- Multi-valued Attribute Class (MAC): A class that symbolises a multi-valued attribute.
- The Composite Attribute Class (CAC) is a class that denotes a composite attribute.
- Regular Relationship Class (RRC): a M:N relationship class without characteristics.

Based on the following procedure of data abstraction, data attribute & relationship their unique keys, Construction of CDM are performed.

A. Building the CDM from Relational Database:

Key matching is used to categorise relations and their properties, identify links between relations, and calculate their cardinalities. All of them are converted into equivalents in the CDM. Following that, the semantically enhanced CDM may be translated into the target schema. Each relation R is classified based on a comparison of its primary key (PK) with the PKs of other relations and mapped to one of the nine CDM classifications listed above. If C.cls:= ("XXX" | "YYY"), it is crucial to determine whether class C is concrete or abstract after it has been classed. C is a concrete class (i.e., abs: = false) when some of its corresponding RDB table rows are not members of other sub-tables, and abstract otherwise. The detected qualities of R are mapped to the recognised attributes of C. To create the Rel of C relationships, R's keys are utilised. Using this data, the connections are located, their cardinalities are established, and these connections are then mapped into Rel as association, inheritance, or aggregation. Using matching data, every relationship that R is a part of is found, mapped into a comparable relationship rel, and added to Rel.

For example, PKs are italicised, while "*" is used to denote FKs. Fig. 2 displays (in part) the final CDM that was derived from the RDB while concealing its very intricate structure. Each RDB relation is translated into a CDM class. For instance, the relation EMP is mapped into the CDM class EMP, an abstract SST class with the following attributes: ename, eno, bdate, address, spreno, and dno. Other characteristics (such as attribute types and default values) are not shown for space reasons. The EMP class is 'connected with' the classes DEPT (twice), WORKS_ON, and with itself (twice). Additionally, it 'aggregates' the KIDS class and 'inherited by' the SALARIED_EMP and HOURLY_EMP classes. Also provided for each class are the cardinality c and unique keys. Relationships are specified in each class as RelType (where invAs and dirAs denote association and aggregation, respectively).

B. Transforming XML Schema from CDM:

The conversion of a CDM into an XML Schema file (.xsd) is described in this section. A collection of mapping rules is used to convert CDM constructs into XML Schema annotations during the translation process. The XML target schema is defined initially,

and the algorithmic procedures for converting CDM constructs into their target schema counterparts are then described. [21] has all the information.

XML Schema - Annotations, element declarations, and type definitions (XML documents):

Additionally, the document could include additional elements such as attribute and model groups [21]. The XML Schema language standard defines identity constraints, which allow relationships and integrity restrictions to be expressed among related items. In the root element of the schema, you may use the key element, the refkey element, and the unique element to establish unique key constraints, primary key constraints, and foreign key constraints, respectively. The XML Schema has a number of tools for dealing with inheritance relationships, including derived types, replacement groups, and abstract type mechanisms. Complex types may be formed from other types by utilising the extension, limitation, and choice keywords, which allow a sub-class complex type to expand its super-class type complex foundation. Furthermore, a super-class complex type may be termed abstract if all instances of its sub-classes inherit all of its instances. With respect to the number of elements, minOccurs and maxOccurs are used.

The prospective target XML Schema may be constructed as two components as a result of this. One is a global element, which represents the root of the XML tree as a complicated type encompassing schema components and restrictions. The second component is a set containing all global complex types. Each complex type may be used as a type of one (or more) elements stated in the root or in other complex types. Using the extension and base keywords, an inheritance is expressed.

Algorithm 1: TransXML Schema Algorithm

Input: CDM

Output: XML Schema

Defining the Schema as Root → aR

Setting the GT → aG to denote the complex types

Defining the database Namespace and annotation

Assign aR as Rn

Tuples nm, mx, Rel_{mn}

For Class CDM_c belongs to CDM do

If (CDM_c class not belongs ('NAME' | 'XXX') then

C_{type} → Complex Type {ct_n, base, abst, LE }

$C_{type}.(n) \rightarrow C_{cdm}.n$

$C_{type}.abst \rightarrow C_{cdm}.abs$

For Each attribute ATT belongs to $C_{cdm}.att_{cdm}$ do

Nm, mx, → '1'.

If AT.n = 'y' then

Mn = '0'

end if

if (CDM_c class not belongs ('YYY' | 'ZZZ' | 'AAA') then

if AT.t not belongs 'pf'then

CT.le → CT.le U {ATT.a, mapping AT(attribute), mn, mx}

End if

Else

CT.le → CT.le U {ATT.a, mapping AT(attribute), mn, mx}

End if

End for

// if C_{CDM}.Class not belongs && C_{CDM}.abs then

a.root {Primary Keys (PKs), Foreign Keys (FKs), and Unique Keys (UKs)} → a.R U {Primary Keys (PKs), Foreign Keys (FKs), and Unique Keys (UKs)}$_{CDM}$

For relationship belongs to $Class_{CDM}$.Relationship do

Get the master class based on the relationship for $Class_{CDM}$.Relationship

If (relationship → associated && class = 'XXX' then

CT.le → relationship

Else if relationship type → Aggregates then

Relatioship {mn} → Get relationship {Attributes}

Nm, mx → min and max cardinality rel.c

If {Class.C = 'XXX'} then

// NFK Type

Else

// CT.le → Union

End if

End if Relationship type → inherits then

Return

End Algorithm

C. Declare the schema Roots:

According to a tree data model, which comprises two primary parts, an XML Schema is developed. The root element is initially declared as a complicated type that includes a series of elements and integrity requirements. The definitions of the complex types of the items defined beneath the root are included in the second component. The choice of whether to specify the schema's components locally or globally is accessible in three typical ways. Salami-slice, Russian-doll, and Venetian-blind designs are examples of these strategies [22]. Depending on the needs of the application, any technique may be used. This article uses the Venetian Blind design to create the target XML Schema, which specifies complicated types globally and elements locally. This allows for the flexible reuse of components and the nesting of element declarations inside of type definitions.

The root element of the schema document, aRoot, is constructed and given the name rootn with the same name as an existing RDB schema (or an alternative specified by the user), once the namespace and annotations have been defined. The collection of components that make up the root aRoot is established by the algorithm's future phases. Describe the set of global complex types aGT after defining LE and its identity constraints PKx, FKx, and UKx. From aRoot and aGT, the target XML Schema document is produced.

D. Converting Attribute relationship and Constraints:

An XML Schema specifies layered complicated kinds or restrictions utilising the key/keyref to indicate connections between items. On the one hand, the construction of nested types employs the parent-child confinement strategy, which often results in data redundancy even if it may speed up query processing by eliminating join operations. Furthermore, layering the parts needs user assistance during translation. However, creating associations with key/keyref may lead to a flat document, even if the documents produced using this method have less duplication. So, we use both of these methods to reduce the amount of duplicate information in a stacked text. Thus, links between core CDM classes are mapped into identity constraints using the key/keyref, whilst the MAC, CAC, and RRC classes are translated as nested elements beneath their parent elements.

IV. Performance Analysis

In this Trans XML Schema proposed process, effective Trans XML Schema proposed is design and objective of this performance analysis helps to obtain data reliability with appropriate perception. Here we have considered the throughput based on records execution time and searching time based on Activities and Data source. The existing methods such as, base line [23] and semi-automated [24] are compared and analysis is made to represent how the proposed method outperforms the existing one.

In Fig. 7, we have calculated the throughput, which gets improved based on the variation in the record size as it grows gradually. While considering the fig. 7, existing method of baseline has very high variation in the values compared to other existing method of semi-automated and proposed one. Here we will find a slightly different between the exiting semi-automated and Trans XML Schema proposed. Finally, the proposed one will outperform the existing ones and it give improved throughput based on the variation in the record by considering the MySQL database.

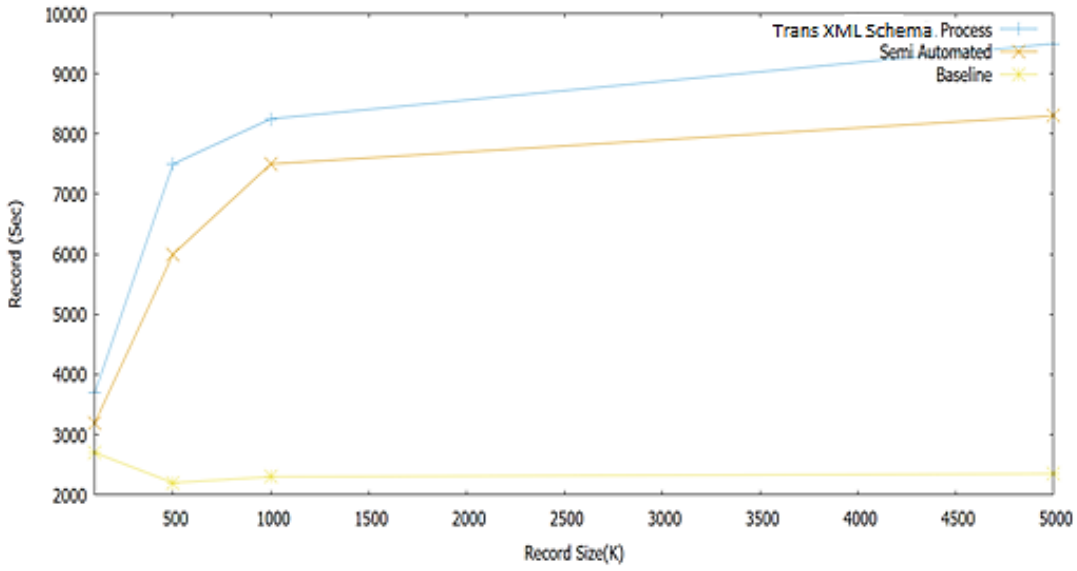

Fig. 7. Throughput improvement based on Increase in the record size Via MySQL

In Fig. 8, we have calculated the throughput based on the variation in the record size as it gradually increases. While considering the fig. 8, existing method of baseline has very high variation values gets record below 2500 per Sec values as compared to other existing method of semi-automated and proposed one. Here we will find a different between the exiting semi-automated and Trans XML Schema proposed as the proposed one gets the record of 30000 per sec values. Finally, the proposed one will outperform the existing ones and it give enhanced throughput based on the variation in the record by considering the PostgreSQL database.

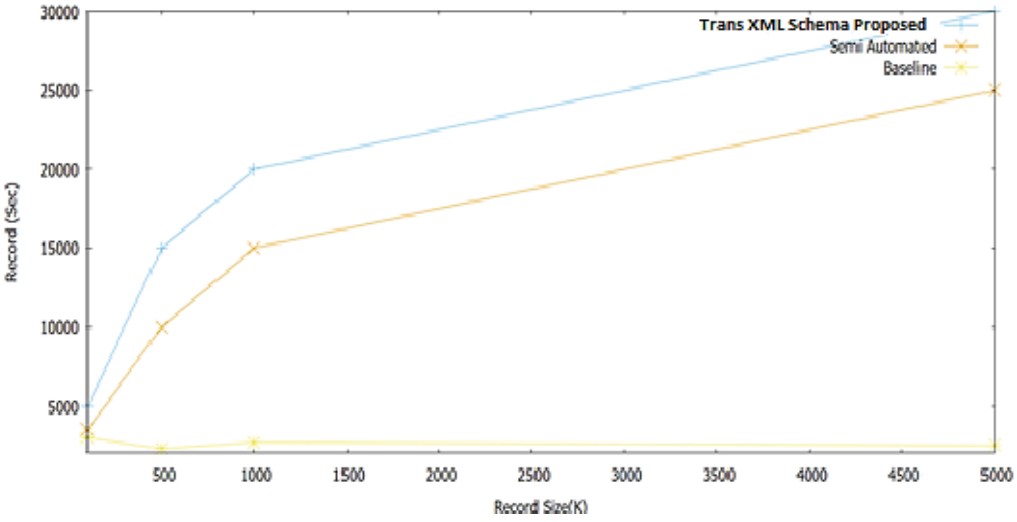

Fig. 8. Throughput improvement based on Increase in the record size Via PostgreSQL

Based on the number of data source, new issue of search time occurs and there will be increase in the data source is considered. From the fig. 9, we infer that there is an increase in the data source where there will be an increase in the search time. Here we have considered the data source varies from 0 to 10 to calculate the searching time in millisecond. Here we have compared the existing method of baseline and semi-automated which is compared with the Trans XML Schema proposed. From the fig, 9, we infer that the proposed system gets variation and gets better search time even though there is a variation in the data source and the proposed system outperforms the existing ones.

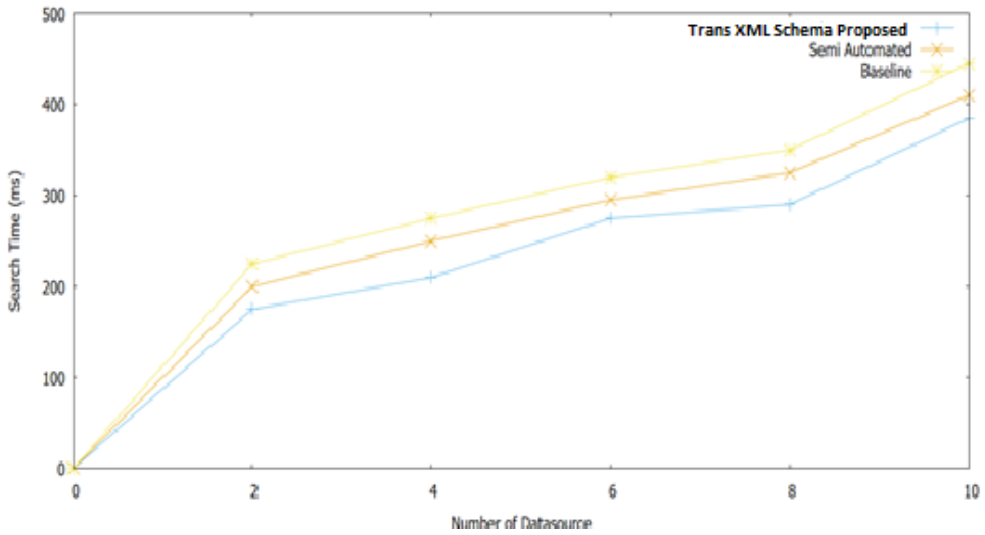

Fig. 9. Search Time based on the variation in the data source

In the fig. 9, we have considered the ETL activities which varies from 20 to 140 and the search time is calculated. When there is an increase in the number of ETL activities, then the search time also gets varies. Here we have compared the proposed DW ETL process with existing baseline and semi-Automated method and from, which we infer that the proposed method gets effective search time and gives some variation compared to the existing ones.

V. Conclusion

The issue of converting RDBs into XML documents is addressed in this study. The technique is useful since it creates not just the XML schema but also data instance documents. The solution also makes use of the extensive set of potent capabilities made available by the XML Schema standard. A prototype has been created to realise and verify the solution's algorithms, which are also tested by comparing query results from the input and output databases. By comparing the variations between the XML documents produced by the prototype and the original RDB, we have carried out two trials to assess our strategy. The tests assess the prototype's inputs and outputs in terms of schema structures, data semantics, integrity requirements, and data instance equivalents. After analysing the query results from both databases, it was discovered that they were identical. As a result, we infer that the source and target databases are comparable. Also, the results show that the answer is possible, effective, and right, both in theory and in practise.

References:

[1] Abdelhafz BM, Elhadef M (2021) January. Sharding Database for Fault Tolerance and Scalability of Data. In 2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM) (pp. 17–24). IEEE.

[2] Abourezq M, Idrissi A (2016) Database-as-a-service for big data: An overview. International Journal of Advanced Computer Science and Applications (IJACSA), 7(1).

[3] Agarwal S, Rajan KS (2017) Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries. In Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings (Vol. 17, No. 1, p. 4).

[4] Singh et al. Journal of Cloud Computing (2022) 11:53 Page 16 of 17 4. Agarwal T, Quelle H, Ryan C (2020) Stock Trend Evolution. University of Arizona.

[5] Ahmad AAS, Andras P (2019) Scalability analysis comparisons of cloudbased software services. Journal of Cloud Computing 8(1):1–17

[6] Ahmad K, Alam MS, Udzir NI (2019) Security of NoSQL database against intruders. Recent Patents on Engineering 13(1):5–12

[7] Compose, An IBM Company. Alba, L., November 2016. Building OHLC Data in PostgreSQL. Available from https://www.compose.com/articles/building-ohlc-data-in-postgresql/. Accessed 26 Oct 2021.

[8] Antas J, Rocha Silva R, Bernardino J (2022) Assessment of SQL and NoSQL Systems to Store and Mine COVID-19 Data. Computers 11(2):29

[9] Bagui S, Nguyen LT (2015) Database sharding: to provide fault tolerance and scalability of big data on the cloud. International Journal of Cloud Applications and Computing (IJCAC) 5(2):36–52

[10] BalaMurali A, Sravanthi PS, Rupa B (2020) January. Smart and Secure Voting Machine using Biometrics. In 2020 Fourth International Conference on Inventive Systems and Control (ICISC) (pp. 127–132). IEEE.

[11] Gartner Bala R, Gill B (2021) Magic Quadrant for Cloud Infrastructure and Platform Services. Available from https://www.gartner.com/doc/reprints?id=1-271OE4VR&ct=210802&st=sb. Accessed 26 Oct 2021.

[12] Balusamy B, Kadry S, Gandomi AH (2021) NoSQL Database. Big Data: Concepts, Technology, and Architecture, Wiley, pp. 53–81.

[13] Beaulieu A (2009) Mary E Treseler (ed.). Learning SQL (2nd ed.).Sebastopol, O'Reilly. ISBN 978–0–596–52083–0.

[14] GitHub Singh B (2021) Cloud based evaluation of databases. Available from https://github.com/handabaldeep/cloud-based-evaluation-of-databases. Accessed 26 Oct 2021.

[15] Bhatti HJ, Rad BB (2017) Databases in cloud computing. Int J Inf Technol Comput Sci 9(4):9–17

[16] Cao Z, Dong S, Vemuri S, Du DH (2020) Characterizing, modeling, and benchmarking rocksdb key-value workloads at facebook. In 18th {USENIX} Conference on File and Storage Technologies ({FAST} 20) (pp. 209–223).

[17] Chakraborty S, Paul S, Hasan KA (2021) January. Performance Comparison for Data Retrieval from NoSQL and SQL Databases: A Case Study for COVID-19 Genome Sequence Dataset. In 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST) (pp. 324–328). IEEE.

[18] Chauhan VP (2019) Google Big Table: A Change to Data Analytics. International Journal of Information Security and Software Engineering 5(1):5–9

[19] Chawathe SS (2019) September. Cost-Based Query-Rewriting for DynamoDB: Work in Progress. In 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA) (pp. 1–3). IEEE.

[20] Chen JK, Lee WZ (2019) An Introduction of NoSQL Databases based on their categories and application industries. Algorithms 12(5):106

[21] Codd EF (1970) A Relational Model of Data for Large Shared Data Banks. Commun ACM 13(6):377–387

[22] Cooper BF, Silberstein A, Tam E, Ramakrishnan R, Sears R (2010) Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM symposium on Cloud computing (pp. 143–154).

[23] DB-Engines. DB-Engines Ranking 2021. Available from https://db-engines.com/en/ranking. Accessed 8 Oct 2021.

[24] Dean J, Ghemawat S (2008) MapReduce: simplifed data processing on large clusters. Commun ACM 51(1):107–113