Optimized Model of Ledger Database Management to handle Vehicle Registration

Neha Gaonkar
School of Computing and Digital Technology
Birmingham City University
Birmingham, B4 7XG, UK
neha.gaonkar@mail.bcu.ac.uk

Parnia Samimi
School of Computing and Digital Technology
Birmingham City University
Birmingham, B4 7XG, UK
parnia.samimi@mail.bcu.ac.uk

Jagdev bhogal
School of Computing and Digital Technology
Birmingham City University
Birmingham, B4 7XG, UK
jagdev.bhogal@bcu.ac.uk

Luke Porter
Ld8a Ltd, https://ld8a.com
Greater London
luke@ld8a.com

Rob Squire
Ltd, https://ld8a.com
Greater London
rob@ld8a.com

Abstract— **In recent years, versioning of business data has become increasingly important in enterprise solutions. In the context of big data, where the 5Vs (Volume, Velocity, Variety, Value, and Veracity) play a pivotal role, the management of data versioning gains even greater significance. As enterprises grapple with massive volumes of data generated at varying velocities and exhibiting diverse formats, ensuring the accuracy, completeness, and consistency of data throughout its lifecycle becomes paramount. System of record solutions, which cover most enterprise solutions, require the management of data life history in both system and business time. This means that information must be stored in a way that covers past, present, and future states, such as a contract with a start date in the past and an end date in the future that may require correction at any point during its lifetime. While some systems offer transaction time rollback features, they do not address the business life history dimension of a contract or asset, which requires the developer to code the business rules of the requirements. The relational data model is unable to inherently use relational constraints where a business time dimension of the data is required, as it is a "current view" and not designed for this purpose. Therefore, there is a need for better autonomous capabilities for version control of data, which will bring new functionality and cost reduction in application development and maintenance, reduce coding complexity, and increase productivity. This paper presents an approach to relational data management that relieves the developer from the need to code the business rules for versioning. The framework, called Ld8a, works with a standard Oracle database that keeps the developer in the "current view" paradigm but allows them to specify the point in time that logical insert, update, and delete events take place with the infrastructure autonomously maintaining relational correctness of the dataset across time. The Ld8a framework has been used to address the vehicle registration scenario used by AWS in presenting the capabilities of the Quantum Ledger Database product. This approach offers a solution that maintains referential integrity by the infrastructure across time, making version control of data easier and more efficient for developers.**

Keywords—LD8a framework (Ledger Data framework), Amazon Quantum Ledger Database model (QLDB), Vehicle Registration, Relational database management systems (RDBMSs)

I. INTRODUCTION

Vehicle sales transactions are increasing globally, hence, new techniques are being created to change the current model of vehicle ownership transfer systems. The fact that all vehicles in the market are sold and resold, it becomes a cumbersome task to maintain a legitimate record of the history of each vehicle and make it available when needed. Also, to prevent various types of fraud, the integrity of the data in this type of operation must be given the highest priority. Researchers have worked hard over the years to develop new techniques for effectively managing data [1]. However, the current vehicle registration process stores all the registration-related information in a central relational database [2]. Relational database management systems (RDBMSs) are extremely successful, but they face challenges when most data description is not in their current state. However, the properties of objects and states change over time - vehicle ownership or a customer address can change, and vehicles can also be modified and updated [3].

When the data changes, it is important to ensure its corresponding description is up to date. Though all these things have a history, and many of them will have a future, information about the vehicle's past or future is often useful [4]. Restoring and collecting historical information is usually possible with enough time and effort. However, as vehicle registrations have increased, businesses are recognizing the value of having access to both past and future ownership of the vehicle without the associated delays and costs. To address this issue, we propose a ledger database system for vehicle registration, an information management system distinguished by decentralization, security, and information transparency. A ledger database has a time dimension and stores time-varying data, as opposed to a conventional database, which only stores current data [5]. It maintains a sequenced record of all the changes to the data, which cannot be modified or deleted, giving the ability to query and analyze the full history of the vehicle and its corresponding owner [6]. It also offers exceptional technology for centralized immutable data management [7]. In terms of time, data values have various aspects. The time when a data value becomes effective is referred to as valid time. On the other hand, the time at which a data value is recorded in the database is referred to as transaction time. When a data model supports both times, it is referred to as bitemporal [8].

However, the main reason for the need for a ledger database provided by the LD8a framework is to overcome the limitations of a relational database by eliminating the need for "DATE" fields in table design and the complex coding involved in retrieving the past, present, and future data which requires additional cost. The knowledge base for this research is the Amazon QLDB. A centrally trusted

authority owns the transaction log in the fully managed ledger database known as Amazon QLDB. In Amazon QLDB, since the data is stored in the form of JSON and appended one below the other in the form of journals, incorrect or invalid data can be added to these journals [6]. This challenge has been addressed with the ledger database. It ensures that invalid data is not added to the database. The rest of this paper is organized as follows: Section II briefly highlights the related work on the ledger database. Section III discusses the relational management database system. Section IV describes the enhancements of the database. In Section V, the overview of Amazon QLDB is discussed. Section VI describes the proposed ledger database model. In this section, the architecture of the proposed model is also discussed. Finally, in Section VII, we conclude by briefing the contributions of this paper.

II. RELATED WORK

In recent years, we have seen an increase in several new ledger technologies. Several time-varying database extensions have been proposed and implemented by extending relational data models. Few previous works examined some modern ledger data models.

Vehicle registration has always been a complex process. There are several parties involved in this lengthy process, and there is a chance that information will be manipulated, data will be duplicated, and various errors will occur. In this case, critical information may be susceptible to fraud or data deception, or it may even be trackable. Many of these flaws can be easily addressed by incorporating the power of distributed ledger technology, also known as the blockchain, and moving the entire process of registering a vehicle onto a blockchain [9].

Jiang and Sun in 2021 [10], proposed a blockchain architecture to record the information on vehicle condition. Due to blockchain's transparency and tamper-proofing capabilities, customers can use the system suggested in this paper to check the actual vehicle condition without worrying about being misinformed by used car dealers. A secure distributed ledger with a variety of quality information, asset information, logistic information, and transaction information is provided by blockchain technology. The evaluation result of the suggested system demonstrates that the throughput of transactions is adequate for daily transactions.

In 2010, Snodgrass, R.T. [11] proposed a method for implementing a time-varying application in SQL. The research focused on three concepts: temporal data types, different types of time, and temporal statements. Each of these components is made up of three orthogonal paradigms. The three primary temporal data types are instant, interval, and period. An instant is defined as something that happened at a specific moment in time (an anchored duration of time). Furthermore, there are three fundamental types of time: user-defined time (a time value that cannot be interpreted), valid time (when a fact is true in reality), and transaction time (when an event was recorded in the database). A table may relate to none, one, two, or even all three different types of time. Finally, there are three different categories of temporal statements: current, sequenced (at every point in time), and non-sequential (ignoring time). The distinction holds true for queries, modification statements, views, and integrity constraints.

The prominent feature to consider when designing time variant databases is the time dimension as proposed by Galante et al., in 2005 [12]. There are three different times: application or valid time, system or transaction time, and bitemporal. Kvet, Matiaško, and Kvet [3] proposed temporal data modeling (mostly modeled using uni-temporal and bi-temporal tables) in comparison with conventional tables that can process and retain the information in the past. Transformation of the conventional table to a temporal model is not a trivial problem and needs special structures and resources. In addition, the requirement of the users is to provide compatibility and easy manipulation. The temporal database concept offers new opportunities by adding additional time attributes limiting the validity of the object. Two different states of the object cannot be valid at the same time; the correct attribute values can sometimes be undefined. The aim of temporal database management discussed in this paper was to have information about the whole life cycle of the object, even after logical deletion. Thus, it offers the opportunity to create prognoses and analyses.

The valid time includes future instances, the system time is restricted to the records' most recent and earlier instances [13]. There are two ways to add a timestamp to a relation: tuple time stamping, which utilizes relations of the first normal form and attributes time stamping, which utilizes relations of the non-first normal form. While the valid time includes future instances as well, the system time is restricted to the records' most recent and earlier instances. Appending a timestamp to a relationship can be done in two ways: tuple time stamping (first normal form relations) and attributes time stamping (non-first normal form relations). The time stamps for the start and end of the time-varying records must be inserted by the database user. To ensure the accuracy and efficiency of temporal data, integrity constraints are imposed [13].

Teradata Database versions 13 and 14 include numerous enhancements for SQL features specifically designed for temporal support. Teradata supports all three time zones: valid, transaction, and bitemporal dimensions. The “transaction time” and “valid time” are contained in bitemporal tables [14]. Teradata does not support data coalescing for missing information [13]. Coalescing is the process of joining two overlapping or adjacent value-equivalent tuples to maintain the integrity constraint. It is necessary because when the UPDATE statement is executed, the non-temporal variables of tuples are the same, also overlapping, or sequential timestamps.

Columns are inevitably added and removed from tables over time (a process known as "schema mutation"), but most database implementations only keep the most recent state of the schema. Historical data "breaks" when schemas change as explained by Renwick. A whole class of problems involving time intervals is simply unreasonably complicated in databases. To figure out the logic behind "Did person A and person B work for the company at any point?" (Did the time intervals they worked overlap). Ld8a technology is a solution that is built on Oracle technology using the “roots of Codd” which has made time “a first-class citizen” is fully compatible with immutable/blockchain tables and fully supports the meaning of “time travel”.

III. RELATIONAL DATABASE MANAGEMENT SYSTEM

A relational database management system (RDBMS) is a database management system (DBMS) that stores data in a row-based table structure that connects related data elements as shown in Fig.1. The RDBMS's built-in functions ensure the security, accuracy, integrity, and consistency of data. The CRUD (creating, reading, updating, and deleting) operations are collectively referred to as the most fundamental RDBMS functions. They provide the framework for a well-structured system that encourages uniform data handling. ACID (atomicity, consistency, isolation, and durability) transactions are also supported by RDBMS.

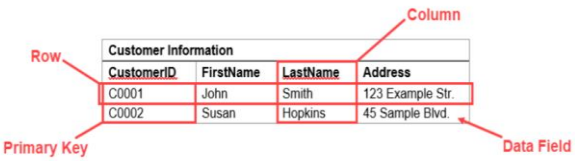

Fig.1. RDBMS current view [15]

A. *RDBMS Operations*

Fig.2 shows the primary high-level built-in operational features of a traditional RDBMS.

RDBMS Operation	RDBMS Operation Definition
Creating tables	All column names and data types are defined in the CREATE TABLE statement.
Creating indexes	For high-performance queries CREATE INDEX statement can be used
Inserting data	INSERT statement that specifies values for a new row or tuple in line with the schema established by the tables.
Querying data	SELECT-FROM-WHERE statement
Updating data	UPDATE-SET-WHERE statement
Deleting data	DELETE-FROM-WHERE statement

Fig.2. RDBMS CRUD (create, read, update, delete) operations.

B. *RDBMS Database Date Limitations*

Relational database theory is in a current view. Time is a prominent aspect of the real world. Events have a time of occurrence or a period of effect. Therefore, the current view of relational theory is compromised, and the development requires coping mechanisms. However, to track the events, it is decided to add a “DATE” column in one of the tables, recording when the row is valid. During testing, it is discovered that the primary key is no longer sufficient, hence, a “DATE” column is added to the primary key. The referential integrity check does not work hence a “DATE” column is added to another table when the row was no longer valid [11].

There are numerous off-on-bugs that have been encountered, in some cases, less-than comparisons should have been made, and in other cases, the code should have "+1 DAY" added. The database modification code becomes complicated. Every modification since then must take the "DATE" columns into account. It has been observed that it is unclear how to approach such changes in a systematic manner. The “DATE” columns do not allow the creation of reliable and accurate business keys to undermine data integrity. Designing, querying, and modifying time-varying tables effectively requires a different set of approaches and techniques than conventional ones [11].

IV. ENHANCEMENTS OF DATABASE

Database technologies take information and store, and process it in a way that enables users to go back easily and intuitively and find the details they are searching for.

A. *Conventional table*

The conventional database system is currently the most popular. It is made up of tables that only store actual data as depicted in Fig.3. As a result, each actual existing object is represented by a single row in the table. Because the database does not reflect attribute changes, it is impossible to predict when (or if) the data will be changed, or the object deleted [4]. As a result, the user has no knowledge of the existence of an object in the database following a deletion request. Monitoring archives can only solve part of the problem [8].

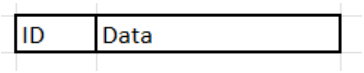

Fig. 3. Conventional database model [3]

B. *Versioned table*

A versioned database is an enhancement of a conventional database. The advantages of the standard approach are used in the versioned table, but the problems with the objects' time validity is eliminated. It not only allows for change monitoring, but it also allows for the creation of future forecasts. In a conventional database, one row represents each state of an object over time. However, a versioned table uses a different number of rows for each object throughout its life cycle. The most basic versioning solution for a versioned table is to add a beginning date to the composite primary key [3].

C. *Ledger Database*

The ledger is a key component of blockchain technology and is a database that, unlike standard databases, keeps a complete history of past transactions in memory as in a notarial archive for the benefit of any future test [7]. The database ledger incrementally captures the state of a database as it evolves over time, while ledger tables are updated. A ledger database can store the current and historical value of a company's data. Ledger databases are a simple solution for applications that require the integrity of all data to be protected for the duration of the database's life when a row in the database is updated, its previous value is saved and protected in a history table. Fig.4 shows the ledger database overview.

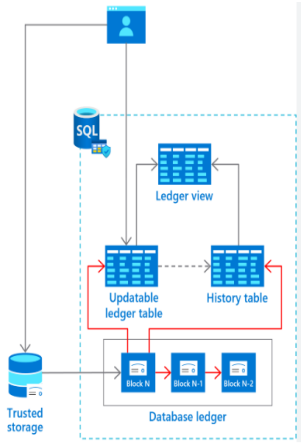

Fig.4. Ledger Database Overview [7]

V. OVERVIEW OF AMAZON QLDB

A new database called Amazon QLDB eliminates the need for complicated development work to create ledger-like applications. In Amazon QLDB, the data's revision history is immutable—it cannot be altered, changed, or deleted. In addition, cryptography makes it effortless to confirm that no unauthorized changes to the application's data have taken place. An immutable transactional log called a journal is used by QLDB. The append-only journal is composed of blocks that have been hash-chained together and contain 16 committed data [6]. The vehicle registration model is implemented in Amazon QLDB which tracks the complete historical information of the vehicle data.

A. Journal First

The journal is the database's core in Amazon QLDB. The journal is an immutable, append-only data structure that is like a transaction log and is used to store application data and its corresponding metadata. The journal maintains a record of each write operation, including updates and deletions [6]. By materializing the ledger data into query-able, user-defined tables, the journal is used by QLDB to determine the current state of the ledger data. These tables also include a history of all transactional information, including metadata and document revisions as shown in Fig.5. Concurrency, sequencing, cryptographic assurance, and ledger data accessibility are other issues that the journal addresses [16]. Journal blocks, like blockchains, are sequenced and chained together using cryptographic hashing techniques. The journal can provide transactional data integrity using this feature's cryptographic verification technique.

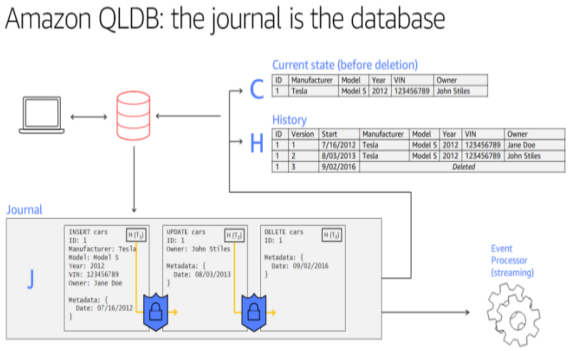

Fig.5. Amazon QLDB journal architecture [6]

B. Immutable

A complete record of all data changes is kept in the append-only QLDB journal, which is immutable and cannot be altered, modified, or overwritten. Any committed data that is already present cannot be changed using APIs or any other methods. It is possible to view and search the entire history of the ledger. The journal is updated once by QLDB during a transaction. The documents that are added, changed, and removed as well as the statements used to commit the documents are represented by entry objects in each block. These blocks are sequenced, and hash chained to ensure data integrity [6].

C. SQL-like and document-friendly

PartiQL is Amazon QLDB's query language, and Amazon Ion is its document-oriented data model [6]. Ion compatibility has been added to PartiQL, an open-source SQL-compatible query language. PartiQL enables the use of well-known SQL operators for data management, insertion, and querying. The syntax for SQL queries on relational tables and queries on flat documents is the same.

D. Verifiable cryptographically

Using cryptographic hashing methods, journal blocks are chained together in a similar manner to blockchains. This feature enables the journal to deliver transactional data integrity using a cryptographic verification method. Using a digest (a hash value that represents a journal's full hash chain as of a particular point in time) and a Merkle audit proof, to confirm that there haven't been any unintended changes to the data at any time (a mechanism that proves the validity of any node within a binary hash tree) [6].

E. Amazon QLDB Limitations

The QLDB journal is append-only, as a result, it maintains an extensive record of all data changes that cannot be altered, modified, or data should be corrected in the present or future to ensure that invalid data is not added to the database, overwritten [6]. In QLDB, the data is appended one after the other, hence, incorrect, or invalid data can be added to the journal. For instance, if a license plate number for a vehicle is valid from the year 2013 and is related to a VIN, and the vehicle corresponding to the VIN is manufactured in the year 2019. Amazon QLDB does not provide the referential integrity concept as the majority of data is not stored or accessed in the current view. The append-only concept does not allow for logical insertion, updating, or deletion, and it also does not preserve referential integrity.

The combination of a database and a document file system known as Amazon QLDB is ineffective and further restricts the range of potential functionality. System errors or failures may result in data inconsistencies between the database and the journal file system, which is one of two sources of truth for Amazon QLDB. Additional checks and maintenance are required when there are multiple sources of truth. For example, the QLDB database was updated, but the journal entry was incomplete due to system issues. The data in QLDB is stored as Amazon ION, a superset of JSON that is document-based [6]. Hence the concept of referential integrity is not supported to maintain data integrity.

VI. PROPOSED MODEL

LD8a technology offers “time travel” relational database solutions and makes complex requirements much easier to handle by simplifying the development process. Ld8a additionally provides ACID compliance by utilizing the power of the relational database management system [17]. LD8a framework is a new kind of database that is built upon the concept of table versioning and eliminates the need to undertake the complex development effort of creating ledger-like applications. In a single relational database, the Ld8a framework keeps track of all application data changes and maintains a complete and sequential history of changes over time. Ledger Database implemented using the Ld8a framework combines the business advantages of ledger database capabilities and relational data management into a single solution that significantly eases application development, introducing a new class of relational database functionality that directly

challenges the limitations of Amazon QLDB. [17]. It also provides a cryptographic audit for all attributes in the tables in a fully relational database system. It delivers a new level of functionality and productivity.

LD8a provides full relational and referential capabilities by default, resulting in a significantly simplified approach to application development. Appending data that is cryptographically locked is used to create and amend database records. As a result, all data in Ld8a is certain to be immutable, hence, it cannot be altered or deleted, making it easy to ascertain whether application data has undergone unintentional changes. Ld8a uses the physical append-only concept, which ensures that all application data operations including inserts, deletes, updates, and corrections to all these database actions are never lost and are easily retrievable. The Ld8a framework maintains referential integrity and importantly keeps the developer in the current view paradigm which is how the relational database is designed to function. The framework is a great way of ensuring that designers or developers learn to model relational data solutions correctly and have the framework do the heavy lifting of ensuring referential integrity is maintained across time.

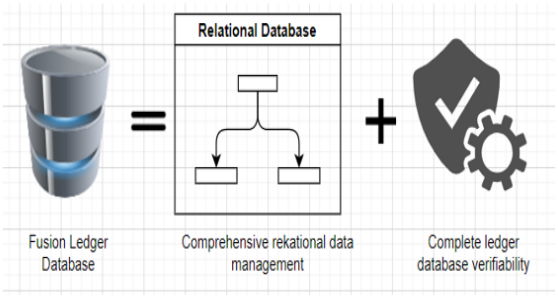

Fig.6. Ledger Database

A. Important LD8a Framework Concepts

The following are the most important concepts of the LD8a framework which need to be examined:

- A table is a collection of related data stored in table format in a database. Tables are like spreadsheets, logically organizing data in a row-and-column format. Each row represents a separate record, and each column is a record field. The LD8a framework includes tables with full relational capabilities [17]. The RDBMS environment is fully accessible to tables, enabling the database to be tuned using conventional relational database techniques. All application data changes are recorded in the tables as an immutable data history.
- Structured Query Language (SQL) is a database communication language. It is the standard language for relational database management systems, according to ANSI (American National Standards Institute). SQL statements are used to perform tasks such as updating or retrieving data from a database. Standard SQL and unrestricted SQL are both used by Ld8a. This indicates that all SQL functions and procedures are supported, and the environment is similar to a conventional relational database. The relational database environment that SQL is designed for is exclusively used by Ld8a technology [17]. It enables full data manipulation and querying via SQL and SQL-like commands.
- A collection of records that have been cryptographically related together to form a journal can be used to verify changes. A crucial task of Ledger DB is its ability to interpret logical INSERT, UPDATE, and DELETE events (and their corrections) into physical INSERT journal entries. Ledger DB allows users to navigate to any journal entry point in the database and view the complete state of the cryptographically locked relationally consistent database. The select statement can be utilized to execute for any point in the journal entry time and effective time without the use of complicated "where" clauses.
- An immutable, transparent, and cryptographically verifiable transaction log that is owned by a central authority is provided by a ledger database, which can be either a SQL or a NoSQL database. Ledgers are typically used to monitor an organization's financial and economic activity [7]. Many businesses develop programs that include ledger-like functionality to maintain an accurate history of the data in their applications. Ld8a's relational database tables, which serve as the system's core for application and journal entry records, provide the system's data structure.
- Ld8a framework enables developers to create and use a standard, current-view relational data model. As a result, the modeling process is made simpler because audit tables are not necessary. The Ld8a environment by default offers complete system audit trails [17].
- In ledger databases, the dimension is the most important concept. There are three types of time: valid time, transaction time, and bitemporal time. Valid time is defined as the time when a row accurately reflects reality by the database user [18]. Transaction time refers to the instant when a row is saved to the database. Bitemporal is the union of valid and transaction time. Most ledger data models only support valid time [13]. All three dimensions are supported by the Ledger Database proposed in this project. An essential part of the modeled reality is the data maintained in databases. Various structures known as entities serve as representations for the mini-facilities worlds. Valid time records the changing states of the modeled reality. Each entity can be assigned a valid time. A fact's valid time indicates when it is true in the real world, which makes it important. Transaction time refers to the moment at which a fact becomes actual. It describes the period over which a database fact is or was stored in the database and is consequently connected to a database entity. It exhibits the evolution of the database's state over time [3].
- In ledger databases, there are three kinds of key constraints: one for the primary key, one for referential integrity, and unique key constraint. Different values for the primary key are used depending on the time dimension that the database uses. If the time dimension is a valid time, the ledger database primary

key, in addition to the relational primary key, must include the temporal attribute where the timestamps are attached to the attribute or apply to a table considering the timestamp as just another column named as non-sequence integrity constraint [14]. When it comes to transaction time, however, the ledger primary key is the same as the relational primary key [3]. The integrity of the data must be confirmed before it is created. Entity, domain, referential, and user-defined integrity are a few of the specific tests used to ensure data integrity. A primary key ensures that no rows in the table are duplicated at different points in time whereas a Unique key ensures no duplication at the same time. Domain integrity makes sure that information is entered into the table in accordance with predetermined criteria, such as file format or value range. Due to referential integrity depending on the different dimensions of time, any row that has been re-linked to a different table cannot be deleted. Finally, user-defined integrity ensures that the table meets all user-defined criteria [19].

B. A database solution based on LD8a Technology

The main purpose of Ld8a technology is to provide a database solution [17]. With the help of Ld8a technology, all data is stored in a database and data manipulation also occurs. For instance, all logical inserts, updates, and deletes are recorded within database table records, which are protected by cryptographically locking all records in the database. The database can be fully encrypted because all insert, update, and delete operations result in records being appended to the database (depending on the needs of the enterprise system, cryptographic data locking is an option). Ld8a framework technology is an exceptional option as a ledger database, providing unprecedented benefits for meeting modern enterprise requirements.

Ld8a framework employs a physical append-only concept, which ensures that all application data inserts, deletes, updates, and corrections to all these database actions are never lost and are easily retrievable. In relational database systems, history and audit tables are frequently created and managed to capture the past and future states of business data; Ld8a, however, simplifies the task for developers by moving the complexity of design and development into the database infrastructure. Due to the centralized environment of Ld8a, transactions can be carried out without the need for multi-party consensus. The full range of referential rules controlling real-time data updates and corrections are supported. Data in Ld8a is accessible to third-party tools like Power BI or Excel reporting because it is organized in a "time-friendly" and relationally structured pattern.

C. Date Effectivity Management in Ledger Database

The provision of Effective Date management is the primary central technical benefit of Ledger DB. Ledger DB provides Autonomous Date Effectivity Management (ADEM) by eliminating the need for “DATE” fields in table designs during the data modeling design process. It allows the application to be set in a "current view" perspective, with all database’s referential integrity enabled and the Ld8a framework handling the time dimensions of the data without complex coding thus overcoming the limitations of relational database. The operating system allows for simple data input, update, and deletion as well as corrections to the data, even though the fact that none of the tables contain a date or time field. This is what is meant by Autonomous Date Effectivity Management (ADEM) [17].

D. The Architecture of the Model

This section includes a thorough overview of the architecture and implementation of the project. Since the primary goal of this dissertation is to analyze the data model and data of vehicle registration provided by Amazon QLDB and implement it using the Ledger DB, there is a requirement of the system that enables the setup of Ledger DB which is implemented in the LD8A framework. Fig.7 shows the overall architecture of the Ledger DB.

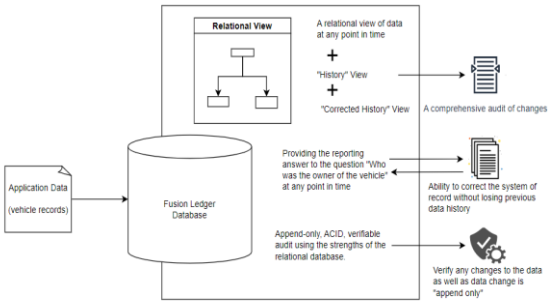

Fig.7. Database design using the Ld8a framework

E. Database Design using the Ld8a framework

The first step in the database design process is data modeling. There are four layers in data modeling: the conceptual data model, the logical data model, the current view model, and the physical data model.

1) Defining entities, their attributes, and their relationships is the goal of creating a conceptual data model. There is not much information available about the actual database structure at this level of data modeling.
2) The structure and relationships of data elements are specified using the logical data model. The logical data model also provides additional information to the conceptual data model elements. The benefit of using a logical data model is that it serves as a foundation for the physical model.
3) A database-specific application of the data model is described in the physical data model. It facilitates the creation of the schema and provides database abstraction. The physical data model replicates database column keys, constraints, indexes, triggers, and other RDBMS features to aid in visualizing database structure.

In Ledger DB, the physical model starts with the source schema (REF_SRC) which is the current view physical model. In LD8a, the current view model does not hold any data. However, it is used as an input to the “exec gen”. The generator creates all the code and creates views for all the tables in SQL developer which simplifies the development and management of the Oracle database. The “exec gen” determines the shape of the model from the Oracle Data Dictionary and generates another physical model underneath which is held in the TPL layer

(REF_TPL) and is the underlying ledger physical model. The user schema (REF_USR) stores the LD8a-generated views and application code e.g., ref_pkg, types, scripts, and API, etc. The "exec gen" command also checks if there is any attribute added or the design is changed in the SRC schema and accordingly resets all the information from the TPL schema and generates a new physical model. The diagram in Fig. 8 is sourced from Amazon Web Services and consists of five tables in the model.

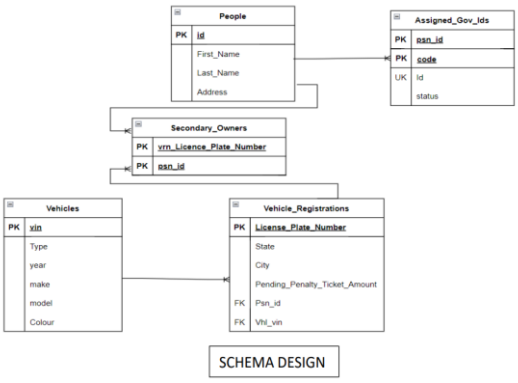

Fig.8. Schema design for vehicle database

F. Database Implementation using LD8A Framework

The tables about vehicles, their owners, and vehicle registration information along with attributes are created in the "carprj_src" schema. The creation of the following different entities can be done with a basic "CREATE TABLE" statement. A primary key constraint specifies two guidelines for the related table. First, none of the key columns in any row may have a null value for their respective values. Second, no two rows can have the same values in the key columns. Every time a change is made, a primary key constraint with ledger support is independently applied. When a row with that key value exists in the referenced table at a specific valid time in the referenced table, the key in the ledger database is a foreign key. The ramifications of ensuring the uniqueness and referential integrity of a current modification impact the period from "now" to "forever".

Events always have a time of happening or a period of effect. Time is a continuum that extends from the past to the future. Data is inserted at a specific point and exists indefinitely until it is acted upon by another event. Data may be updated at one or more points in time with no overlaps, that is, only one

version of the truth at any given time. At some point in time, data is optionally deleted with no gaps, that is, does not reincarnate. The current actions affect the future state. Inserting data in the ledger database using the LD8a framework is always in the given context of time. When a row of data is first added to a table, it remains there indefinitely or only for a predetermined amount of time before being automatically deleted. The time can be changed by the package DBMS, as a result, the database time context can move forward in time or back in time. The DBMS "get valid time" and "set valid time" refers to getting and setting the database time respectively. To ensure that the database and the framework time are synced. A persistent component of the framework is the ability to retrieve the time from it to check the current time context. However, DBMS provides the granularity of the transaction time, whereas the user specifies the granularity of the valid time. An update is logically a delete followed by an insert. The value for the validity period has been updated from "now" to "forever," per the most recent update. The update must occur after the insertion. LD8a framework creates the following different views in the "carprj_tpl" schema for all the tables.

- Vehicle_Registration – stores the information of the vehicle registration in the current transaction time.
- Vehicle_Registration _L – stores all the rows' last known state before updating.
- Vehicle_Registration_Z – stores the current information about vehicle registration with additional metadata and gives a vertical cut of the row.
- Vehicle_Registration _Y – stores the horizontal cut of the versions of the vehicle registration row.
- Vehicle_Registration _X – stores the transaction time history of the vehicle registration model.
- Vehicle_Registration _E – stores all the event (insert, update, delete) history of the vehicle registration model.
- Vehicle_Registration _A – stores all the history of the row and its corresponding values changed over a period for the vehicle registration model.

The start of existence (SOE) is when that vehicle registration row was created, the start of valid (SOV) is the beginning of the time that each of these different versions was created and the end of existence (EOE) is when the vehicle registration is deleted. If the end of existence is null, the row lasts forever in the database.

G. CORRECTIONS IN LD8A FRAMEWORK

A correction is defined as going back to the history of the row in valid time and updating the row which will affect the entire future value of the row. Correction is like an update that happens in the middle of the history of the row. When a correction is made, it is persistent to ask the system the following:

- How it needs to cope with all the future values?
- Does it last forever or update only until the next change?
- Does it update until the end of existence or does it update to a specified time?

An update rule needs to be set while making a correction. For example, if the owner of the vehicle was updated on 14th Jan 2020 instead of 12th Feb 2020, the valid time should be set to perform the update and inform the system if the owner should last till the end of the existence of the row or until the next owner.

VII. CONCLUSION

The primary aim of this project was to explore, design, and implement a solution to address the limitations of incorporating "Date" columns into a relational database. Additionally, it involved an analysis of different aspects of the Ld8a framework and Amazon QLDB. The project delved into the constraints related to QLDB within vehicle registration models and explored potential solutions for these challenges. To pinpoint the most effective features aligning with the intended outcome, an extensive evaluation of currently available tools and methods was initially conducted. This led to the creation of a solution using the Ld8a framework, which has the capability to

extract historical data. Furthermore, this endeavor manages inaccurate data across various time instances while preserving data integrity.

In the context of big data and the 5Vs (Volume, Velocity, Variety, Veracity, and Value), this project highlights the need for managing time-varying data within a relational database, emphasizing the significance of the "Variety" aspect of big data. The utilization of the Ld8a framework and Amazon QLDB demonstrates how these technologies can handle the complexities of data changes over time, thereby addressing the "Veracity" aspect by ensuring accurate and trustworthy data records. The project's focus on historical data extraction and preservation aligns with the "Value" aspect of big data, as it enables the creation of analyses, reports, auditing, and other purposes for vehicle registration data, contributing to the overall value derived from the data.

The inherent capabilities of relational database systems must be expanded to align with the business's need for data management within the business timeline. The approach offered by the Ld8a framework deserves deeper examination from researchers who are focused on devising solutions for handling data that evolves over time. The following list outlines potential avenues for further investigation: (1) The creation of API using Oracle Rest Data Services (ORDS) and Procedural language for SQL (PL/SQL) to manage ledger resources and non-transactional operations to create, delete, describe, list, and update the data (2) Verify data cryptographically and to run data transactions on the ledger (3) Creating indexes to optimize high-performance queries on data retrieval operations (4) The extension of use into JSON document database handling methodologies. For instance, the compatibility with Oracle's recent release of JSON Relational duality.

References

[1] Künzner, F. and D. Petković. *A comparison of different forms of temporal data management*. in *Beyond Databases, Architectures and Structures: 11th International Conference, BDAS 2015, Ustroń, Poland, May 26-29, 2015, Proceedings 11*. 2015. Springer.

[2] Hossain, M.P., et al. *Vehicle registration and information management using blockchain based distributed ledger from bangladesh perspective*. in *2020 IEEE region 10 symposium (TENSYMP)*. 2020. IEEE.

[3] Kvet, M., K. Matiaško, and M. Kvet, *Complex time management in databases.* Central European Journal of Computer Science, 2014. **4**: p. 269-284.

[4] Johnston, T. and R. Weis, *Managing time in relational databases: how to design, update and query temporal data*. 2010: Morgan Kaufmann.

[5] Tansel, A.U., *On handling time-varying data in the relational data model.* Information and Software Technology, 2004. **46**(2): p. 119-126.

[6] *Amazon Web Services, Amazon QLDB.* . [cited 2023 6 January]; Available from: https://aws.amazon.com/qldb.

[7] *Ledger overview*. 2023; Available from: https://learn.microsoft.com/en-us/sql/relational-databases/security/ledger/ledger-overview?view=sql-server-ver16.

[8] Jensen, C.S. and R.T. Snodgrass, *Temporal data management.* IEEE Transactions on knowledge and data engineering, 1999. **11**(1): p. 36-44.

[9] Gupta, K.D., *Recent Advances in IoT and Blockchain Technology*. 2022.

[10] Jiang, Y.-T. and H.-M. Sun, *A blockchain-based vehicle condition recording system for second-hand vehicle market.* Wireless Communications and Mobile Computing, 2021. **2021**: p. 1-10.

[11] Snodgrass, R.T., *Developing time-oriented database applications in SQL*. 1999: Morgan Kaufmann Publishers Inc.

[12] de Matos Galante, R., et al., *Temporal and versioning model for schema evolution in object-oriented databases.* Data & Knowledge Engineering, 2005. **53**(2): p. 99-128.

[13] Kumar, S., *Study of Time-varying Data Models.*

[14] Petkovic, D. *Temporal data in relational database systems: A comparison*. in *New advances in information systems and technologies*. 2016. Springer.

[15] Kovačević, A. *What Is a Relational Database?* 2021; Available from: https://phoenixnap.com/kb/what-is-a-relational-database.

[16] Fekete, D.L. and A. Kiss, *A survey of ledger technology-based databases.* Future Internet, 2021. **13**(8): p. 197.

[17] Ltd., L.a. *Product overview*. Available from: https://ld8a.com/product-overview/.

[18] Kulkarni, K. and J.-E. Michels, *Temporal features in SQL: 2011.* ACM Sigmod Record, 2012. **41**(3): p. 34-43.

[19] Brush, K. *RDBMS (relational database management system)*. 2019; Available from: https://www.techtarget.com/searchdatamanagement/definition/RDBMS-relational-database-management-system.