ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

# RFID/Sensor Big Data to Implement MongoDB-based Repository Design using ACID

*P.Sudhakaran*
M.Phil Research Scholar,
Department of Computer Science and Engineering,
Alagappa University, Karaikudi, Tamilnadu
sudhakarpanneer@gmail.com

*Dr.E.Ramaraj*
Professor,
Department of Computer Science and Engineering,
Alagappa University, Karaikudi, Tamilnadu
eramaraj@rediffmail.com

***Abstract*— Data are characterized by its continuous generation, large amount, and unstructured format. Existing relational database technologies are inadequate to handle such data due to the limited processing speed and the significant storage-expansion cost. The continuous information growth in current organizations has created a need for adaptation and innovation in the field of data storage Current benchmarks evaluate database performance by executing specific queries over mostly synthetic data. Firstly, we devise a data repository schema that can effectively integrate and store the heterogeneous data sources such as RFID, sensor, and GPS, by extending the event data types in Electronic Product Code Information Services (EPCIS) standard, a de facto standard for the information exchange services for RFID-based traceability. Secondly, we propose an effective shard key to maximize query speed and uniform data distribution over data servers. So we introduced ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably. It's avoid issues in transaction process and given more effective and efficient for the RFID/sensor big data.**

***Keywords*— *Big Data, EPCIS, MongoDB, RFID, Sensor, Supply Chain***

## I.INTRODUCTION

In the last couple of decades, radio-frequency identification (RFID) technology has been widely used in logistics,manufacturing, defense, environment, health care, agriculture, retail, aviation, and information technology. Moreover, the use Internet enabling technologies, including RFID, sensors, global positioning systems (GPSs), and automated actuators has lately been expanded to production management, factory management, quality management, logistic management, utility management, and inventory management in various industries[5][8]. In the manufacturing industry, for example, once an RFID tag is attached to individual parts or products, their location information can be collected in real time, thereby enabling flexible production planning and shipment order placement. Furthermore, if quality issues arise, their causes can be analyzed using corresponding sensor data collected from the relevant manufacturing facility to pinpoint the source of the quality problems. In the food industry, safety management of perishable food has been extended from production to disposal. Data such as RFID and sensor data, is not only constantly generated in real time as the supply chain and the manufacturing processes continue, but also provided in a variety of data formats. However, we have many limitations on processing large amounts of unstructured data, such as big data, using existing relational database (RDB) technologies [7][9].

NoSQL technologies can store unstructured data because of their easy data schema modification capability, and require lower server expansion cost than relational databases because of their "scale-out" scheme compared to the "scale-up" scheme of RDB's. Besides, NoSQL database systems can process massive input and output data efficiently by virtue of the distributed storage and processing approach over the multiple data nodes. In this study, we propose a sensor-integrated RFID data repository-implementation model that can integrate and store a large amount of RFID/sensor data collected from supply chain processes, and can quickly process and query them.

To achieve this goal, firstly we design a MongoDB-based RFID/sensor data repository that can integrate and store RFID and sensor data by referencing event types of Electronic Product Code Information Services (EPCIS) in the EPC global network, which is a de-facto standard for RFID information exchange. Secondly, we suggest an effective shard key. A shard is a horizontal partition of data in a database, or the database server in which the partition is located. A shard key is a set of fields to which the partitioning is carried out with respect.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

## II.BACKGROUND

*NoSQL*

The term 'NoSQL' collectively refers to the database technologies which do not abide by the strict data model of relational databases. By sacrificing some of the properties (such as ACID transactional properties) of relational database model,NoSQL databases can achieve higher availability and scalability, which are essential requirements for big data processing. Unlike relational databases NoSQL databases do not need to have a fixed schema with pre-defined data structures and constraints to be finalized in an early stage of database design.classified NoSQL databases into three types – key-value stores, document databases, and column oriented databases depending on their data models. In a key value store, all the data instances are stored in the form of key value pair. Document databases are based on the same key -value structure as key-value stores, but the values are in the form of a more complex data structures called "documents" such as XML documents.

TABLE I
CLASSIFICATION OF NOSQL DATABASES [12]

| Key-value stores | Document databases | Column-oriented databases |
|---|---|---|
| Voldemort, | SimpleDB, | BigTable, |
| Riak, | CouchDB, | HBase, |
| Redis, | MongoDB, | HyperTable, |
| Scalaris, | Terrastore | Cassandra |
| Tokyo Cabinet | | |

Unlike the key-value stores, document databases generally support secondary indexes and multiple-type/nested documents in a database. Column oriented databases store data tables as sections of columns of data, rather than as rows of data. This type of database is efficient when the majority of the database operations are of OLAP, which requires intensive column-oriented calculations like aggregation. Unlike row-oriented databases, column oriented databases can easily add and delete columns.

*MongoDB*

MongoDB, developed by 10gen, is a document-oriented NoSQL database that offers high performance and scalability. Unlike other NoSQL databases, its data structure is designed independently as a document unit so that a schema definition is not needed. Moreover, MongoDB uses a scale-out scheme,which is flexible against hardware expansion, and supports auto-sharding. Thus, the automatic distribution of data over a number of servers can be conveniently carried out. Fig. 1 shows the configuration of horizontal data partitioning, called sharding, and replica sets to ensure high availability, safety, and data consistency. They also enable distributed expansion for data processing involving large amounts of data.

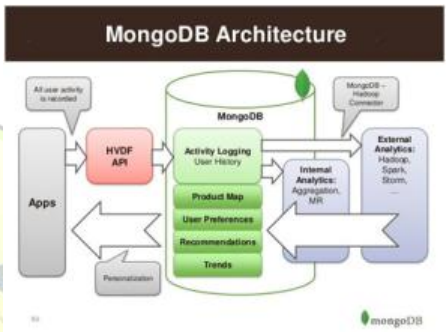


Fig. 1. MongoDB Process Architecture

*EPC Information Services (EPCIS)*

EPCIS is an RFID event repository, which is one of the core components of the EPC global Architecture Framework. It helps store RFID event information and share the information among supply chain partners. Electronic Product Code (EPC) refers to a coding scheme for unambiguous code for the designation of physical goods [1]. It can assign codes, according to an appropriate coding scheme such as Serialized Global Trade Item Number (SGTIN), a Shipment Container Code (SSCC), and a Global Returnable Asset Identifier (GRAI), to objects depending on the purposes. The EPCIS solutions should provide predefined vendor-independent capture/query interfaces to receive/supply RFID event data, although the vendors can freely implement the services by their own ways.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

● : mandatory ○ : optional

| | Object Event | Aggregation Event | Transaction Event | Quantity Event | Description |
|---|---|---|---|---|---|
| eventTime | ● | ● | ● | ● | The date and time at which event occurred. |
| recordTime | ○ | ○ | ○ | ○ | The date and time of the event were recorded by a repository. |
| epcList | ● | | ● | ● | A list of observed of EPCs naming the physical objects. |
| childEPCs | | ● | | | A list of the EPCs of the contained objects. |
| parentID | | ○ | ○ | | The identifier of the parent of the association. |
| epcClass | | | | ● | The identifier specifying the object class to which the event pertains |
| action | ● | ● | ● | ● | How an event relates to the lifecycle of the entity being described. {ADD,OBSERVE,DELETE}* |
| quantity | | | | ● | The number of objects within the class described by this event. |
| readPoint | ○ | ○ | ○ | ○ | The read point at which the event took place. |
| bizLocation | ○ | ○ | ○ | ○ | The business location where the objects may be found. |
| bizStep | ○ | ○ | ○ | ○ | The business step of which the event was a part. |
| disposition | ○ | ○ | ○ | ○ | The business condition of the objects. |
| bizTransactionList | ○ | ○ | ● | ○ | A list of business transactions that define the context of the event. {bizTransactionType, bizTransaction}** |
| extensions | ○ | ○ | ○ | ○ | This identifies the addition of new data members. |

* *ObjectEvent - ADD*: commissioned; *DELETE*: decommissioned; *OBSERVE*: simple observation
*AggregationEvent - ADD*: physically aggregated; *DELETE*: physically disaggregated; *OBSERVE*: simple observation
*TransactionEvent - ADD*: associated to business transactions; *DELETE*: disassociated from business transactions; *OBSERVE*: simple observation
** *bizTransactionType*: An identifier that indicates what kind of business transaction (i.e. purchase order, invoice number)
*bizTransaction* : An identifier that denotes business transaction ID

Fig. 2. Four event types of EPCIS (edited from [29]).

## III. ACID

ACID properties of transactions In the context of transaction processing, the acronym ACID refers to the four key properties of a transaction: atomicity, consistency, isolation, and durability. Atomicity All changes to data are performed as if they are a single operation. That is, all the changes are performed, or none of them are. For example, in an application that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account. Consistency Data is in a consistent state when a transaction starts and when it ends. For example, in an application that transfers funds from one account to another, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction. Isolation The intermediate state of a transaction is invisible to other transactions.

As a result, transactions that run concurrently appear to be serialized. For example, in an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither. Durability After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure. For example, in an application that transfers funds from one account to another, the durability property ensures that the changes made to each account will not be reversed.

## IV. MONGODB-BASED RFID/SENSOR DATA REPOSITORY

In this section, we describe the design of our RFID/sensor data repository in consideration of the MongoDB design pattern as well as the process for selecting an optimal shard key. With reference to the database schema of a well-known open-source EPCIS, we propose a MongoDB schema, which combines RFID and sensor data. Furthermore, we select an optimal compound shard key according to the theoretical guidelines suggested in previous literature.

### *Relational Database Schema for EPCIS Event Types*

which is an open-source RFID software platform project, provides an EPCIS implementation based on MySQL, which is one of the popular relational database systems. Fig. 3 shows the ObjectEvent schema of the Fosstrak EPCIS. Since EPCIS event fields may have multiple values, such as epcList, childEPCs,bizTransactionList, and extensions, a field table with multiple values and event tables are normalized with oneto-many or many-to-many relationships to overcome addition/deletion/update anomalies. The AggregationEvent, TransactionEvent, and QuantityEvent are also designed in the same manner. According to the EPCIS standard, every event has an extension point, to which additional data members can be attached. Therefore, we can easily store additional information from the various data sources, such as sensors, GPS's, and other intelligent devices.

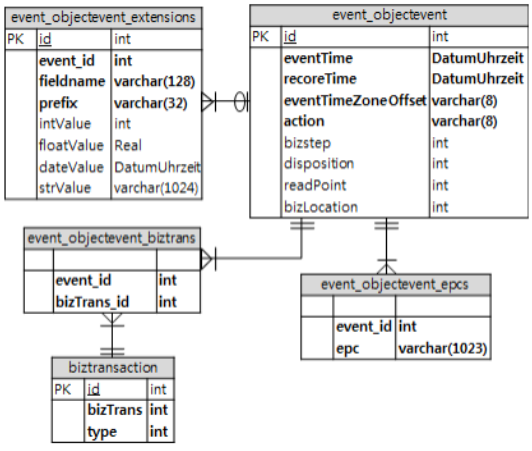


Fig. 3. ObjectEvent schema of Fosstrak EPCIS (revised from [30])

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

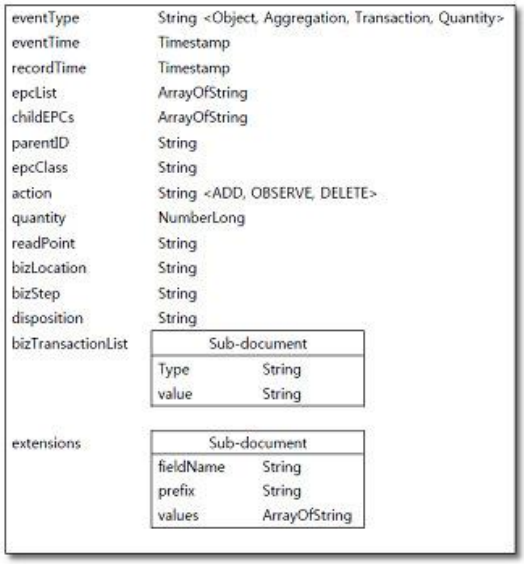


Fig. 4 MongoDB-based RFID/sensor data repository schema.


Fig. 7. Example of XML format for RFID/sensor event.

*Selection of Shard Key*

A shard key should be determined to divide input data into chunks effectively, and to distribute data through the shard evenly. An ideal shard key is a compound key of two fields, in which the first should have a moderately coarse-grained cardinality and the other should be a more fine-grained field with higher cardinality [2], [3]. Once selected as a shard key, the field is to be indexed too. Thus, the choice of a field as a shard key should also involve considering whether it is frequently used in queries as criteria.

Two fields - eventType and readPoint - are the only ones with finite cardinality among MongoDB-based RFID/sensor event member fields. The readPoint field is appropriate as the first part of the compound shard key, in the sense that each readPoint, which represents a point where an object is recognized, is identified by a unique ID. Once a logistic process in business is defined, all the objects will pass through almost the same (finite) set of readPoints. Note that readPoint is optional in the EPCIS event schema. In order to use readPoint as a shard key, implementers should define readPoint as a Non-Null field having a finite set of values in developing RFID/sensor systems.

*Experimental Evaluation*

In order to validate our proposed design, we carried out a series of experiments using sample data set, which was generated based on a realistic supply chain. The first experiment compares data distribution levels and query performance of different shard key choices in order to validate our choice of shard key using with ACID. The second experiment compares query performance of MongoDB-based repository with MySQL-based repository on a single machine in order to check if our choice of database (i.e., NoSQL) outperforms a representative relational database (like MySQL).The last experiment checks whether an increase in the number of MongoDB shards improves query performance.

## V. CONCLUSION

RFID and sensor technologies are undoubtedly the core

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

technologies. Many researchers [5-7] have studied various research issues on integrating RFID and sensor technologies. At present, efforts are being made to integrate these two technologies on the same platform in different fields. Unlike conventional studies that provide platforms at the architecture level only, this study proposed an implementation model of an RFID/sensor data repository on the basis of MongoDB. Furthermore, based on logistic process simulation of automotive parts, the proposed RFID/sensor data repository was empirically validated in terms of even distribution of data and query speed. And could be used ACID for enhanced transaction performance model in Bigdata process. It also given good output for who could used these structure in their industries.

## *References*

[1] F. Thiesse, C. Floerkemeier, M. Harrison, F. Michahelles, and C. Roduner, "Technology, standards, and 4eal-world deployments of the EPC network," *IEEE Internet Comput.*, vol. 13, no. 2, pp. 36-43, 2009.

[2] K. Banker, *MongoDB in Action*, Manning Publications Co, 2011.

[3] R. Copeland, *MongoDB Applied Design Patterns*, O'Reilly Media, Inc., 2013.

[4] L. Zhang and Z. Wang, "Integration of RFID into Wireless Sensor Networks: Architectures, Opportunities and Challenging Problems," In *Proc. the 5th International Conference on Grid and Cooperative Computing Workshops* (*GCCW '06*), pp. 463-469, 2006.

[5] J. Mitsugi, T. Inaba, B. Pátkai, L. Theodorou, J. Sung, T.S. López, D. Kim,D. McFarlane, H. Hada, Y. Kawakita, K. Osaka and O. Nakamura, "Architecture Development for Sensor Integration in the EPCglobal Network," Auto-ID Labs, White Paper WP-SWNET-018, 2007.

[6] H. Liu, M.B.a.A. Nayak and I. Stojmenovic, "Taxonomy and Challenges of the Integration of RFID and Wireless Sensor Networks," *IEEE Network*,vol. 22, no. 6, pp. 26-35, 2008.

[7] A. Al-Fagih, F. Al-Turjman, W. Alsalih and H. Hassanein, "A Priced Public Sensing Framework for Heterogeneous IoT Architectures," *IEEE Trans. Emerg. Topics Comput.* - Special Issue on Cyber-Physical Systems,vol.1,no.1,pp.133-147,2013.

[8] E. Ilie-Zudor, Z. Kemeny, F. Blommestein, L. Monostori, and A. Meulen,"A survey of applications and requirements of unique identification systems and RFID technique," *Comput Ind*, vol. 62, pp. 227-252, 2011.

[9] J. S. Veen, B. Waaij, and R. J. Meijer, "Sensor data storage performance:SQL or NoSQL, Physical or Virtual," in *Proc. 5th IEEE Cloud*, pp. 431-438, 2012.