

Towards data integrity in Cassandra database applications using conceptual models

Pablo Suárez-Otero
Computer Science Department
University of Oviedo
suarezgpablo@uniovi.es

ABSTRACT

Data modeling in Cassandra databases follows a query-driven approach where each table is created to satisfy a query, leading to repeated data as the Cassandra model is not normalized by design. Consequently, developers bear the responsibility to maintain the data integrity at the application level, as opposed to when the model is normalized. This is done by embedding in the client application the appropriate statements to perform data changes, which is error prone. Cassandra data modeling methodologies have emerged to cope with this problem by proposing the use of a conceptual model to generate the logical model, solving the data modeling problem but not the data integrity one. In this thesis we address the problem of the integrity of these data by proposing a method that, given a data change at either the conceptual or the logical level, determines the executable statements that should be issued to preserve the data integrity. Additionally, as this integrity may also be lost as a consequence of creating new data structures in the logical model, we complement our method to preserve data integrity in these scenarios. Furthermore, we address the creation of data structures at the conceptual level to represent a normalized version of newly created data structures in the logical model.

CCS CONCEPTS

•Software and its engineering → Software creation and management

KEYWORDS

NoSQL, Cassandra, Data Modelling, Data Integrity, Consistency

ACM Reference format:

Pablo Suárez-Otero. 2019. Towards data integrity in Cassandra database applications using conceptual models: In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings*. <https://doi.org/10.1145/3377812.3381405>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICSE '20 Companion, May 23–29, 2020, Seoul, Republic of Korea
© 2020 Copyright is held by the owner/author. Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7122-3/20/05...\$15.00
<https://doi.org/10.1145/3377812.3381405>

1 Introduction

Most applications work with databases to manage their data, NoSQL databases are specifically used when high performance to insert and query data is needed, and relational databases cannot achieve it [21]. This performance is achieved in part by allowing a distributed system as well as by replicating data, which is not possible in relational databases [12]. Depending on their data model, NoSQL databases are classified in [3]: those based on key-values such as Redis, those based on columns such as Cassandra, those based on documents such as MongoDB and those based on graphs such as Neo4J. These databases do not have the ACID properties, but they guarantee the BASE properties [16]: Basic Availability, Soft state and Eventual consistency. For example, this last property ensures that eventually all the nodes will contain the same data. These characteristics bring challenges to the developers such as how to guarantee the validity of transactions. Other challenges are related to the nature of the database, such as in those based on columns where the duplication of data in several tables to improve performance brings new challenges to the integrity of these data [6].

Development of software applications that use relational databases usually starts with the definition of the conceptual model with its entities and relationships and continues with the creation of the logical and physical models with the definition of the tables and columns. However, for Cassandra databases this process differs as the tables in the logical model are created following a query-driven approach to retrieve data faster, in which each table satisfies a query defined in the client application. This implies data duplication in the database as the same data can be queried from different tables. Data modeling methodologies also recommend the use of a conceptual model in this process to better organize the data [4, 8, 22]. Logical data integrity of the database means that each piece of data is consistent when it is stored in different tables. This integrity must be preserved in the application by embedding the appropriate database statements [22], as opposed to relational databases where it is preserved by defining constraints in the database. If the developer does not appropriately build these statements, their execution would break the logical data integrity.

We may illustrate this problem in Fig. 1 through an example of a database that stores members pertaining to a department. This figure depicts both the conceptual model (Department and Member) and the logical model, which contains the tables that satisfy the queries: 1) “find members from a given department” (Members_by_department) and 2) “find a member giving their ID”

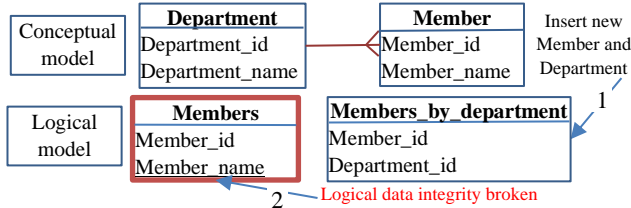


Fig. 1 Example of logical data integrity broken in an insertion

(Members). Note that the data of a certain member is stored in both tables. Suppose that during the development of a function to insert a new member and its department into the database, the statement to insert the member in “Members” is not implemented. The execution of this function would break the logical data integrity, as the member is only inserted in “Members_by_department”.

The previous example shows how a wrong implementation of a data change can break the logical data integrity. These data changes can be built upon the conceptual model or the logical model. If the data change is performed at the conceptual level, it must be properly reflected at the logical level, while if it is performed at the logical level in a table, it must be replicated to other tables.

When the logical model is modelled after a conceptual model, this conceptual model represents a normalized version of the logical model and every column of the logical model is mapped to a single attribute of the conceptual model. In this thesis we refer to this as inter-model consistency. However, application requirements can change, creating the need to create new data structures in the logical model (tables or columns), breaking the inter-model consistency when these data structures are not modelled after the conceptual model. In addition, the logical data integrity may be broken if these data structures must store data that is already in other tables (data duplication), as they are initially empty.

In this thesis we aim to research how to preserve both the logical data integrity and the inter-model consistency when there are new application requirements such as data changes at the logical or conceptual level or a new data structure in the logical model.

2 Related Work

Most research about logical data integrity is focused on relational databases to check the integrity constraints defined in the schema [9, 13]. These works propose to analyze the correct implementation of the referential integrities and the use of mutation testing to check them. However, as Cassandra does not have referential integrities, this research cannot be used, requiring a new approach.

The Cassandra official team has developed the feature “Materialized Views” [7], which allows to automatically preserve the logical data integrity by defining materialized views (table-like structures) synchronized to a single base table. Every data change performed in the base table is automatically replicated in the materialized view. However, this feature has limitations as a materialized view must only have columns from the base table as well as the same key. Optionally, a single column that was not key in the base table can be part of the key in the materialized view.

The support of join operation in Cassandra has also been studied [11, 15]. In [15] the join is implemented by modifying the

Cassandra source code. In [11] the join is simulated by executing the queries needed to perform it and then joining the data.

New methodologies to design Cassandra tables considering the conceptual model in addition to the queries have also been proposed [2, 4, 14]. In particular, [4] obtains the logical model from a conceptual model and queries considering data modelling principles and mapping rules. In this logical model, each column is mapped to a conceptual attribute. However, it does not approach how to preserve the logical data integrity of this logical model.

There have been also methods to infer a conceptual model from a logical model [5, 14]. In [5] this is approached for document databases like MongoDB, obtaining through a MDE-based reverse engineering approach a normalized model of the different entities stored in the database as documents. In [14] this inference is proposed for column-oriented databases making use of functional and inclusion dependencies to normalize the model. However, the generation of new entities, relationships and attributes in an existing conceptual model to maintain the consistency between a conceptual model and a logical model has not been researched.

3 Research Questions

As mentioned in the introduction, to preserve the logical data integrity when performing a data change, either at the conceptual level or at the logical level, the developer must implement the appropriate database statements, which is error prone. Furthermore, the creation of new data structures may break the inter-model consistency if not modeled after the conceptual model. This can also break the logical data integrity if the new data structures must store duplicated data. To structure our research, we have defined the following research questions to approach these scenarios:

- RQ1: How can the logical data integrity be preserved when performing a data change at the conceptual level?
- RQ2: How can the logical data integrity be preserved when performing a data change at the logical level?
- RQ3: How can inter-model consistency be maintained when a new data structure is created in the logical model?
- RQ4: How can the logical data integrity be preserved after the creation of a new data structure in the logical model?

4 Research Questions approach

RQ1: How can the logical data integrity be preserved when performing a data change at the conceptual level?

A data change at the conceptual level is an insertion, update or deletion of a tuple that contains values assigned to attributes from one or more entities. We propose a method that, given a data change at the conceptual level, generates the appropriate CQL statements [1] needed to preserve the logical data integrity. This method has been detailed in a work that has already been published [19].

Given a data change at the conceptual level (tuple) this method completes the following process: 1) Identifies the columns mapped to the attributes of the tuple. 2) Collects the tables where the data change needs to be reflected. 3) Determines the CQL statements to execute for each collected table and obtains the data needed for it.

For example, in an insertion the data to insert in each column is obtained either from the given tuple or from other database tables (via the appropriate SELECT statements). 4) Reflects the data changes in the tables executing the required CQL statements.

RQ2: How can the logical data integrity be preserved when performing a data change at the logical level?

A data change at the logical level consists on an insertion, deletion or update of a row in a table, which requires the replication of this change to other tables to preserve the logical data integrity. Given a data change in the logical model, our proposal to address this problem contains the following steps: 1) Generate the equivalent data changes at the conceptual level (insertion, deletion or update of tuples). 2) For each data change at the conceptual level, execute the method defined in RQ1. We have illustrated the combination of these two methods in Fig. 2, showing that when there is a data change at the logical level, it first goes up to the conceptual model (bottom-up) and then goes back down to the logical model (top-down).

RQ3: How can the inter-model consistency be maintained when a data structure is created in the logical model?

As mentioned in the introduction, a logical model modelled after the conceptual model has inter-model consistency. However, this consistency may be broken if new data structures were created in the logical model (columns or tables) without considering the conceptual model. This is because these new data structures would not be represented in the normalized conceptual model and the columns would not be mapped to any attribute either.

We address this problem with a method that analyzes the properties of the new data structures in the logical model and with the information obtained in this analysis which does the following: 1) Generates the appropriate attributes, relationships and entities in the conceptual model needed to preserve the inter-model consistency. The generated attributes are added to either an existing entity or to a new entity that is also generated along with its relationships. 2) Maps the new columns with their appropriate attributes. The columns can be mapped to either attributes generated in step 1 or that were already in the conceptual model.

RQ4: How can the logical data integrity be preserved after the creation of a data structure in the logical model?

After the inter-model consistency is assured when a new data structure is created in the logical model (RQ3), we may face the problem that the logical data integrity is broken, as the new data structures are empty and they may need to store data contained in other tables (data duplication).

We approach this problem by: 1) Identifying the columns that store the data that must be inserted in the new data structures, using the attribute-column mapping. 2) Collect the tables that contain the identified columns. We must also prioritize from what tables the data is obtained if more than one table contains the same column. However, it is possible that no table contains all the columns and therefore the data is obtained from different tables. 3) Obtaining the data to insert into each new data structure. When the data is obtained from different tables, we must combine these data

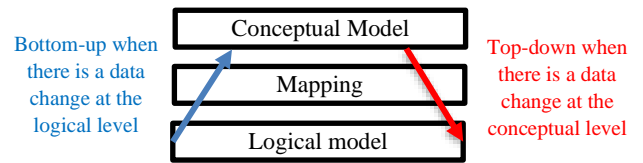


Fig. 2 Methods combination to preserve logical data integrity appropriately. 4) Insert the data into the new data structures. These data must be structured in rows to be inserted in each data structure.

5 Methodological Approach

In this section we define the research products and classify the validation techniques for each RQ following the classification defined by Shaw [17]. We also detail the preliminary results and the case studies that will be used to evaluate our methods.

5.1 Research product

For each RQ we develop a method with its corresponding tool:

- RQ1: Given a data change at the conceptual level, generates the CQL statements needed to reflect it at the logical level while preserving the logical data integrity and executes them.
- RQ2: Given a data change at the logical level, generates the equivalent data changes at the conceptual level and executes the method developed for RQ1 to preserve the data integrity.
- RQ3: Given a new data structure in the logical model, defines the appropriate attributes, entities and relationships to maintain the inter-model consistency and provides a new conceptual model with them included.
- RQ4: Given a new data structure in the logical model, obtains the data needed to preserve the logical data integrity and inserts these data into the new data structure.

5.2 Evaluation

We will use four case studies compound of a logical model and a conceptual model to evaluate the methods described in 5.1. Three of them were used to validate the work of [4] and are available in the KDM tool [10]: Digital Library Portal, Media Cataloguing and Music Database. The fourth case study, developed to explain how to model data in Cassandra, represents a hotel data system [2]. After performing the evaluation with these case studies, we will also evaluate our methods with case studies from the industry.

5.3 Validation techniques

The validation techniques for all RQs are implementation and evaluation. The implementation technique will be done by creating tools that implement the methods developed for each RQ.

To validate the methods developed for RQ1 and RQ2 we will perform several data changes, either at the conceptual level (RQ1) or the logical level (RQ2), in the case studies described in section 5.2. Then, we will use an oracle developed by us that checks if a Cassandra database has logical data integrity to validate the methods. This oracle implements the conceptual model in a relational database and performs in it the same data changes that are performed in the Cassandra database. Then, per each Cassandra table, it executes two queries: one against Cassandra where all the

data from the table is obtained and another one against the relational database that queries the equivalent data that the Cassandra table stores. If both queries retrieve the same data for each analyzed Cassandra table, then the methods have preserved the data integrity.

To validate RQ3 and RQ4 we will create new data structures in the logical model of the case studies described in section 5.2, then we will apply the methods and finally we will validate them using an appropriate oracle for each RQ. For RQ3 we will use as an oracle the KDM tool [10] which, given a conceptual model and a set of queries, generates a logical model. Given both a new data structure in the logical model and the conceptual model generated by our method, the mechanism of this validation would be: 1) Uses the KDM tool to generate a logical model that includes the given new data structure, using as an entry our generated conceptual model. 2) Compares the data structure in the logical model given as an entry with the one generated by KDM. If the given data structure is accurately generated by KDM, then our method would have preserved the inter-model consistency. For RQ4 we use the oracle defined in the previous paragraph to validate RQ1 and RQ2.

5.4 Preliminary results

Thus far, we have finished the method to preserve the logical data integrity in data changes at the conceptual level (RQ1), publishing both its definition and evaluation in two works [18, 19]. In [18] we briefly introduced it along with the identification of data changes (RQ1 and RQ2) and new data structures (RQ4) as causes for breaking the logical data integrity. In [19] we detailed it more by defining the cases where it needs to obtain data from the database to preserve the logical data integrity. In this last work we also briefly introduced our approach to combine the method developed for RQ2 with the one developed for RQ1. Furthermore, we also published evaluation results of this method in [19].

In this evaluation we were able to preserve the logical data integrity in 352 insertions of tuples. These tuples were systematically obtained by creating a set of tuples per each entity and relationship. In these sets, each tuple is different, varying which attributes that have assigned values. We obtained successful results in this evaluation, needing in some insertions to insert data in 66% of the tables of the database. We also worked on the oracle to validate that the logical data integrity was preserved, publishing preliminary successful results in a national conference [20].

6 Timeline

This thesis started in September 2017. In the first part of this year we identified the data changes and creations of data structures that may break the logical data integrity or the inter-model consistency. In the second part we started with our method for data changes at the conceptual level (RQ1), finishing in the first part of the second year and presenting the results in two articles [18, 19] as well as its validation [20]. In the second part of the second year we started approaching the data changes at the logical level, planning to finish it during the first part of this third year. In the second part of this third year we plan to identify when creating new data structures in the logical model implies the loss of the logical data integrity (RQ4) or the break of the inter-model consistency (RQ3), as well as

developing the method to maintain the data integrity (RQ4). In the first part of the fourth year we plan to develop the method to maintain the inter-model consistency (RQ3) as well as its validation. Finally, we plan to write and present the thesis at the end of 2021.

ACKNOWLEDGMENTS

This work was supported by the PERTEST and TESTAMOS projects (TIN2013-46928-C3-1-R, TIN2016-76956-C3-1-R) of the Ministry of Economy and Competitiveness, Spain, the GRUPIN14-007 of the Principality of Asturias and by the ERDF.

REFERENCES

- [1] Apache Software Foundation. 2019. The Cassandra Query Language Retrieved Nov. 18, 2019 from <http://cassandra.apache.org/doc/latest/cql>
- [2] Jeff Carpenter and Eben Hewitt. 2016. *Cassandra: The Definitive Guide: Distributed Data at Web Scale*. O'Reilly Media, Inc.
- [3] Rick Cattell. Scalable SQL and NoSQL data stores. 2011. *Acm Sigmod Record*, 39, 4 (June 2011) 12-27 DOI: <http://doi.org/10.1145/1978915.1978919>
- [4] Artem Chebotko, Andrey Kashlev and Shiyong Lu. 2015. A Big Data Modeling Methodology for Apache Cassandra. *IEEE International Congress on Big Data (BigData'15)*, 238-245. DOI: <http://doi.org/10.1109/bigdatacongress.2015.41>
- [5] Alberto Chillón, Diego Sevilla, Jesús Molina and Feliciano Morales. 2019. A Model-Drive Approach to Generate Schemas for Object-Document Mappers. In *IEEE Access* 7(2019), 59126-59142. DOI: <http://doi.org/10.1109/access.2019.2915201>
- [6] Datastax. 2015. Basic Rules of Cassandra Data Modeling. Retrieved Nov. 18, 2019 from: <https://www.datastax.com/dev/blog/basic-rules-of-cassandra-data-modeling>
- [7] Datastax. 2015. New in Cassandra 3.0: Materialized Views. Retrieved Nov. 1, 2019 from <https://www.datastax.com/dev/blog/new-in-cassandra-3-0-materialized-views>
- [8] Datastax. Data Modeling Concepts. 2018 Retrieved Nov. 18 2019 from <https://docs.datastax.com/en/cql/3.3/cql/ddl/dataModelingApproach.html>
- [9] Gregory M. Kapfhammer, Phil McMinn, and Chris J. Wright. 2013. Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems. 2013 *IEEE Sixth International Conference on Software Testing, Verification and Validation* (2013), 31-40. <http://doi.org/10.1109/icst.2013.47>
- [10] Andrey Kashlev. 2016. Kashlev Data Modeler. Retrieved Nov. 18, 2019 from <http://kdm.dataview.org/>
- [11] Haridimos Kondylakis, Antonis Fountouris, Apostolos Planas, Georgia Troullinou and Dimitris Plexousakis. 2019. Enable Joins over Cassandra NoSQL. In *International Conference on Big Data Innovations and Application*, 3-17 DOI: http://doi.org/10.1007/978-3-030-27355-2_1
- [12] Yishan Li and Sathiamoorthy Manoharan. 2013 A performance comparison of SQL and NoSQL databases. In *IEEE pacific rim conference on Communications, computers and signal processing*, 15-19. <http://doi.org/10.1109/pacrim.2013.6625441>
- [13] Phil McMinn, Chris J. Wright, Colton J. Mccurdy and Gregory M. Kapfhammer. 2015. Automatic Detection and Removal of Ineffective Mutants for the Mutation Analysis of Relational Database Schemas. *IEEE Transactions on Software Engineering* 45, 5 (2015), 427-463. DOI: <http://doi.org/10.1109/tse.2015.2786286>
- [14] Michael J. Mior and Kenneth Salem. 2018. Renormalization of NoSQL Database Schemas. In *37th International Conference on Conceptual Modeling* 479- 487 DOI: http://doi.org/10.1007/978-3-030-00847-5_34
- [15] Christian Peter. 2015. *Supporting the Join Operation in a NoSQL System*. Mather's thesis. Norwegian University of Science and Technology, Norway
- [16] Vatika Sharma and Meenu Dave. 2012. SQL and NoSQL databases. *International Journal of Advanced Research in Computer Science and Software Engineering* 2, 8.
- [17] Mary Shaw. The coming-of-age of software architecture research. In *Proceedings of the 23rd international conference on Software engineering. (ICSE '01)*. IEEE Computer Society, 656. DOI: <http://doi.org/10.1109/icse.2001.919142>
- [18] Pablo Suárez-Otero, María José Suárez-Cabal and Javier Tuya, 2018. Leveraging Conceptual Data Models for Keeping Cassandra Database Integrity. In *Proceedings of the 14th International Conference on Web Information Systems and Technologies*, 398-403. DOI: <http://doi.org/10.5220/0007236303980403>
- [19] Pablo Suárez-Otero, María José Suárez-Cabal and Javier Tuya. 2019. Leveraging conceptual data models to ensure the integrity of Cassandra databases. *Journal of Web Engineering*, 18, 6, 257-286. DOI: <https://doi.org/10.13052/jwe1540-9589.18461>
- [20] Pablo Suárez-Otero, María José Suárez-Cabal and Javier Tuya, Verificación del mantenimiento de la consistencia lógica en bases de datos Cassandra. In *Jornadas de Ingeniería del Software*, 2019. <http://doi.org/11705/JISBD/2019/037>
- [21] Clarence J. Tauro, Aravindh S, and Shreeharsha A-B.. 2012. Comparative study of the new generation, agile, scalable, high performance NOSQL databases. *International Journal of Computer Applications*, 48, 20 (2012), 1-4. DOI: <http://doi.org/10.5120/7461-0336>
- [22] Rajanarayanan Thottuvaikkattamana. 2015 *Cassandra Design Patterns* (2nd ed), Packt Publishing Ltd, Birmingham, United Kingdom.