Leveraging Conceptual Data Models for Keeping Cassandra Database Integrity

Pablo Suárez-Otero, Maria José Suárez-Cabal and Javier Tuya
Computer Science Department, University of Oviedo, Campus de Viesques, Gijón, Spain

Keywords: NoSQL, Conceptual Model, Logical Model, Cassandra, Logical Integrity.

Abstract: The use of NoSQL databases has recently been increasing, being Cassandra one of the most popular ones. In Cassandra, each table is created to satisfy one query, so, as the same information could be retrieved by several queries, this information may be found in several distinct tables. However, the lack of mechanisms to ensure the integrity of the data means that the integrity could be broken after a modification of data. In this paper, we propose a method for keeping the integrity of the data by using a conceptual model that is directly connected to the logical model that represents the Cassandra tables. Our proposal aims to keep the data integrity automatically by providing a process that will undertake such maintenance when there is a modification of the data in the database. The conceptual model will be used to identify the tables that could have inconsistencies and also assist in resolving them. We also apply this approach to a case study where, given several insertions of tuples in the conceptual model, we determine what is needed to keep the logical integrity.

1 INTRODUCTION

Recently, NoSQL databases have been growing in importance due to the advantages they provide in the processing of big data (Moniruzzaman and Hossain, 2013). These databases are not meant to replace relational databases but instead they are intended as alternatives that could achieve better performance in some situations (Leavitt, 2010) such as writing and reading (Li and Manoharam, 2013). Other studies (Cattell, 2011) have claimed that these improved results are due to the abandonment of ACID constraints. Four types of NoSQL databases have been developed without these constraint (Tauro et al., 2012): those based on key-values like Dynamo, those based on documents like MongoDB, those based on graphs like Neo4J and those based on columns like Cassandra.

Cassandra is a distributed database developed by the Apache Software Foundation (Datastax, 2016). Its characteristics are (Han et al., 2011): 1) a very flexible scheme where the addition or deletion of fields is very convenient; 2) high scalability, so if a single element of the cluster fails, it does not affect the whole cluster; 3) a query-driven approach in which the queries are used to organize the data. This last characteristic means that, in general, each Cassandra table is designed to satisfy a single query (Datastax, 2015). This means that the same information could be retrieved in several queries, so in each table that satisfies these queries the information is repeated. This also implies that the model that represents the tables in Cassandra is a denormalized model, unlike in relational databases where it is a normalized model.

Cassandra does not have mechanisms to ensure the logical integrity of the data, therefore it is needed to be kept by the developer (Thottuvaikkatumana, 2015). This could lead to inconsistencies within the data. For example, consider the situation of a Cassandra database that stores data relating to authors and their books. This database has two tables, one created to accomplish the query “books that a given author has written” (books_by_author) and another one created to accomplish the query “find information of a book giving its identifier” (books). Note that the information pertaining to a specific book is repeated in both tables. Suppose that, during the development of a function to insert books in the database, the developer forgets to introduce a database statement to insert the book in “books_by_author”, producing an inconsistency. This example is illustrated in Figure 1:

Suárez-Otero, P., Suárez-Cabal, M. and Tuya, J.
Leveraging Conceptual Data Models for Keeping Cassandra Database Integrity.
DOI: 10.5220/0007236303980403
In *Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018)*, pages 398-403
ISBN: 978-989-758-324-7

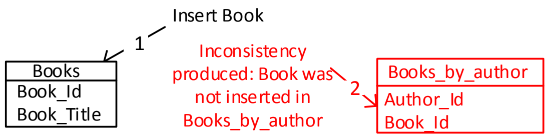


Figure 1: Logical integrity broken.

As the number of tables in a database increases, so too does the difficulty of maintaining the consistency. This article approaches this problem, proposing a solution based on a conceptual model connected to the Cassandra datamodel that automatically keeps the integrity of the data. The contributions of this paper are as follows:

1. A method that automatically identifies the tables that need maintenance of the integrity and proposes how they may be maintained.
2. An evaluation in a case study of the proposed method.

This paper is organized as follows. In section 2, we review the current state of the art. In section 3, we describe our approach to keep the logical integrity of the data. In section 4, we evaluate the results of applying our method to keep the logical integrity in a case study. The article finishes in section 5 with the conclusions and the proposed future work.

2 RELATED WORK

Most works that study the integrity of the data are focused on the physical integrity of the data (Datastax, 2017). This integrity is related to the consistency of a row replicated throughout all of the replicas in the Cassandra cluster. However, in this paper we will treat the problem of the logical integrity, which is related to the integrity of the information repeated among several table.

The official team of Cassandra has studied the problem of keeping the data integrity by developing the feature "Materialized views" (Datastax, 2015). The "Materialized views" are table-like structures where the denormalization is handled automatically on the server-side, ensuring the integrity. Usually, in Cassandra data modelling, a table is created to satisfy one specified query. However, using this feature, the created tables (named base tables) are meant to store data that will be queried in several ways through Materialized Views, which are query-only tables. Every modification of the data in a base table is reflected in the materialized views, it not being possible to write data directly in a materialized view. Each materialized view is synchronized with only one base table, not being possible to have information from more than one table, unlike what happens in the materialized views of the relational databases. This means that if there is a query that involves information stored in more than one base table, it is not possible to use Materialized Views to satisfy it, and the creation of a normal table is required.

Related to the aforementioned problem is the absence of Join operations in Cassandra. A study (Peter, 2015) has researched the possibility of adding the Join operation in Cassandra. This work achieves its objective of implementing the join by modifying the source code of Cassandra 2.0 but it still has room for improvement regarding the performance of the implementation.

There have also been studies (Chebotko et al., 2015) that have given a great deal of importance to the conceptual model, such as where a new methodology for Cassandra data modelling is proposed that uses a conceptual model to create the Cassandra tables in addition to the queries. This is achieved by the definition of a set of data modelling principles, mapping rules, and mappings. Regarding our problem, the work in (Chebotko et al., 2015) introduces an interesting concept: the use of a conceptual model that is directly related to the tables of a Cassandra database, an idea that we will use for our approach.

The conceptual model is the core of the previous work (Chebotko et al, 2015) but it is unusual to have such a model in NoSQL databases. Regarding this, there have been studies that propose the generation of a conceptual model based on the database tables. One of these works (Ruiz et al., 2015) is focused on generating schemas for document databases but claims that the research could be used for other types of NoSQL databases. These schemas are obtained through a process that, starting from the original database, generates a set of entities, each one representing the information stored in the database. The final product is a normalized schema that represents the different entities and relationships.

3 KEEPING THE LOGICAL INTEGRITY

In Cassandra there is no mechanism to ensure the integrity of the data, and therefore it must be controlled in the client application that works with the Cassandra database. We have identified two types of modifications that can break the logical integrity:

- *Modifications of the logical model*: When there is a modification regarding the tables, such as the creation of a new table or the addition of columns to an existing table. In this case, the data integrity is broken because the information that the new columns must store may already be found in the database.

- *Modification of data:* We define a modification of data as the change of the values stored in a row in the logical model or the change of the values of a tuple in the conceptual model. After a modification of data in a table, an inconsistency is produced if the modified data has functional dependencies with other data stored in other tables.

In this work we will focus on the modifications of data.

3.1 Complete Approach

Firstly, we intend to identify a way to detect the tables where the logical integrity can be broken with a given modification of data. To address this problem, we have decided to use a conceptual model that has a connection with the logical model (model of the Cassandra tables). This connection (Chebotko et al, 2015) provides us a mapping where each column of the logical model is mapped to one attribute of the conceptual model and one attribute is mapped from none to several columns. We will use this mapping for our work to determine in which tables an attribute is stored. This is done through a static analysis of both models.

Our approach is divided in two: the top-down approach and the bottom-up approach. In the top-down approach, given a modification of data in the conceptual model (insertion, update or deletion of a tuple), we identify in the logical model the insertions, updates and deletions of rows we must do to keep the data integrity and how they must be done. In the bottom-up approach, given a modification of data in the logical model (insertion, update or deletion of a row), we identify in the conceptual model, through the use of the mapping attribute-column, the attributes mapped to the columns of the row. Finally, we determine what modification of data are equivalent in the conceptual model (insertion, update or deletion of a tuple). Note that the result of the bottom-up approach, a modification of data in the conceptual model, is the entry of the top-down approach. Therefore we can combine both approaches to provide an automatization for keeping the data integrity after a modification of data in the logical model. This combination between both approaches is illustrated in Figure 2:

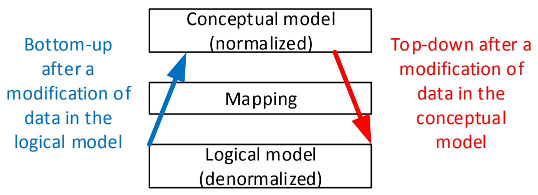
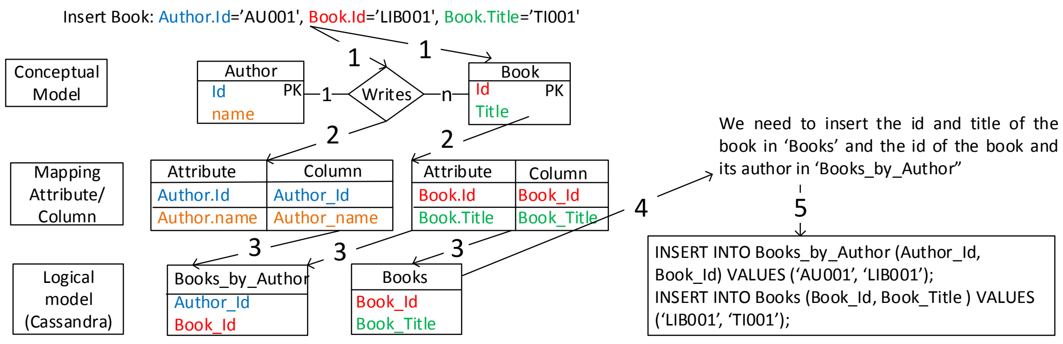

Bottom-up after a modification of data in the logical model
Conceptual model (normalized)
Mapping
Logical model (denormalized)
Top-down after a modification of data in the conceptual model

Figure 2: Approaches top-down and bottom-up combined.

In this work, we will detail the process of the top-down approach in the next section through an example.

3.2 Top-down Approach

The case in Figure 1 is an example of this approach. In addition to the tables "books_by_author" and "books" we also have a conceptual model with the entities "Author" and "Book" where an author has written several books. Suppose that we insert a tuple with the information of a book and the author who wrote it (step 1). Then, we detect, by means of attribute-column mapping (step 2), that in both table there are columns that correspond with the attributes of the inserted tuple (step 3). In order to keep the data integrity, we determine (step 4) that we need to insert in each table a row containing the appropriate information from the tuple. Finally (step 5) in order to automate the approach, we translate these insertions to CQL statements (Apache Software Foundation, 2017). This is illustrated in Figure 3.


Insert Book: Author.Id='AU001', Book.Id='LIB001', Book.Title='TI001'
Conceptual Model
Author
Id PK
name
1
Writes
n
Book
Id PK
Title
1
1
2
2
Mapping Attribute/ Column
Attribute Column
Author.Id Author_Id
Author.name Author_name
Attribute Column
Book.Id Book_Id
Book.Title Book_Title
3
3
3
4
We need to insert the id and title of the book in 'Books' and the id of the book and its author in 'Books_by_Author"
5
Logical model (Cassandra)
Books_by_Author
Author_Id
Book_Id
Books
Book_Id
Book_Title

```
INSERT INTO Books_by_Author (Author_Id,
Book_Id) VALUES ('AU001', 'LIB001');
INSERT INTO Books (Book_Id, Book_Title ) VALUES
('LIB001', 'TI001');
```

Figure 3: Top-down approach.

We have analysed this approach and in section 4 we show the results of the experimentation of applying this approach for several insertions of tuples.

4 EVALUATION

This case study (Chebotko et al, 2015) is about a data library portal. Its conceptual model contains 4 entities and 5 relationships displayed in Figure 4. On the other hand, the logical model contains 9 tables and it is displayed in Figure 5. We have evaluated 126 insertions of tuples in the different entities and relationships of this conceptual model with the objective of determining what CQL operations (INSERT, UPDATE or SELECT) are needed to keep the integrity.

The tuples inserted always contain values assigned for the primary key of the entity or, in the case of the relationships, the primary keys of both entities related. Optionally, the tuples can also contain values assigned for the attributes that are not key. For each entity and relationship we have inserted several tuples, where each one contains different attributes with assigned value. In Table 1 a summary of these insertions is displayed: the number of insertions of tuples evaluated (in entities and relationships) and total, average and maximum number of operations INSERT, UPDATE and SELECT operations needed to keep the logical integrity in the database.

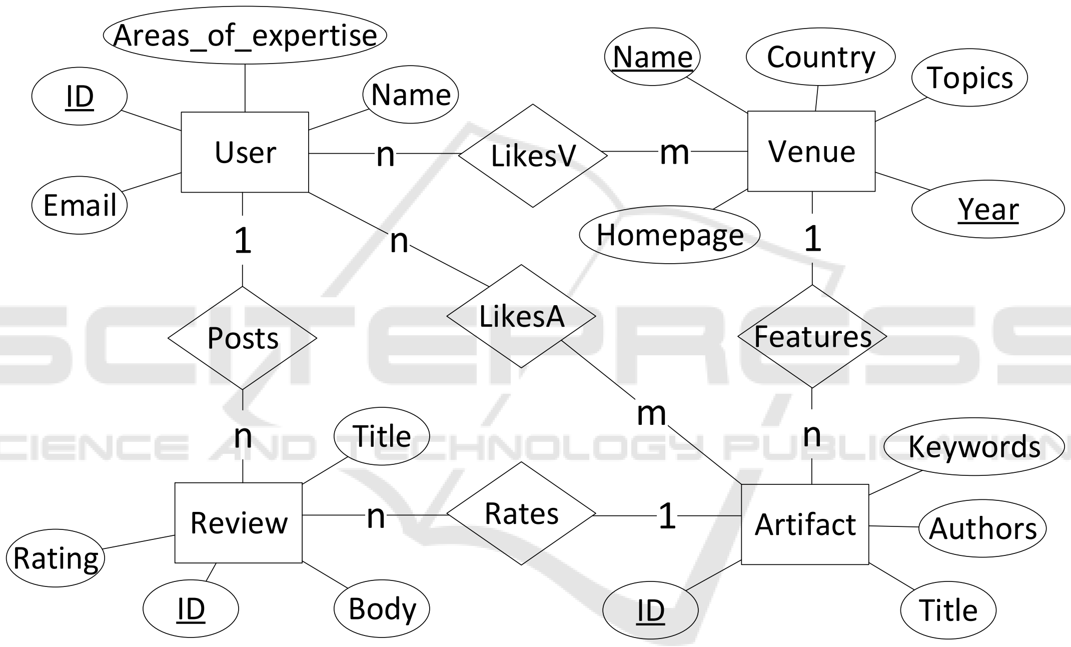
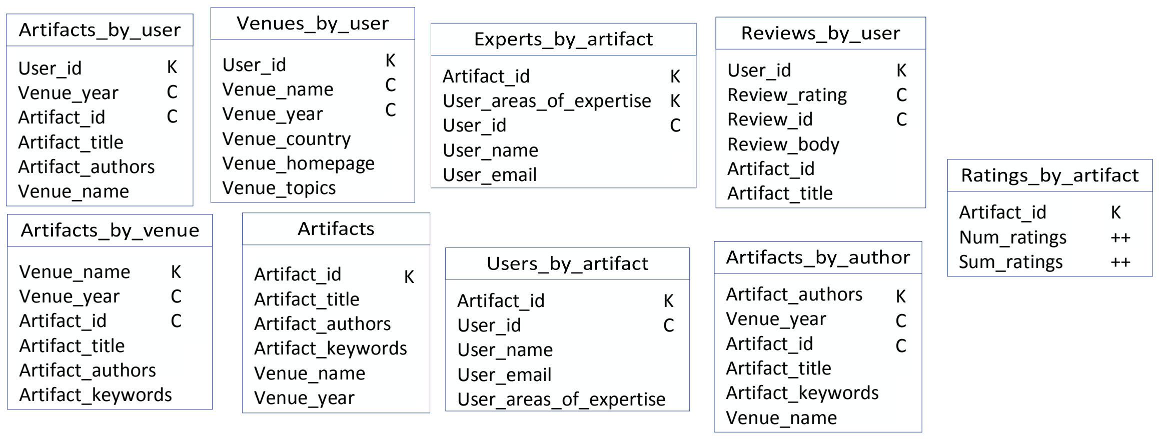


Figure 4: Conceptual model of the case study.



Artifacts_by_user	
User_id	K
Venue_year	C
Artifact_id	C
Artifact_title	
Artifact_authors	
Venue_name	

Venues_by_user	
User_id	K
Venue_name	C
Venue_year	C
Venue_country	
Venue_homepage	
Venue_topics	

Experts_by_artifact	
Artifact_id	K
User_areas_of_expertise	K
User_id	C
User_name	
User_email	

Reviews_by_user	
User_id	K
Review_rating	C
Review_id	C
Review_body	
Artifact_id	
Artifact_title	

Ratings_by_artifact	
Artifact_id	K
Num_ratings	++
Sum_ratings	++

Artifacts_by_venue	
Venue_name	K
Venue_year	C
Artifact_id	C
Artifact_title	
Artifact_authors	
Artifact_keywords	

Artifacts	
Artifact_id	K
Artifact_title	
Artifact_authors	
Artifact_keywords	
Venue_name	
Venue_year	

Users_by_artifact	
Artifact_id	K
User_id	C
User_name	
User_email	
User_areas_of_expertise	

Artifacts_by_author	
Artifact_authors	K
Venue_year	C
Artifact_id	C
Artifact_title	
Artifact_keywords	
Venue_name	

Figure 5: Logical model of the case study.

Table 1: Summary of the results for keeping the data integrity for the inserted tuples.

		Operation INSERT			Operation UPDATE			Operation SELECT		
Entiity/Relationships	Number of inserted tuples	Total	Average	Maximum	Total	Average	Maximum	Total	Average	Maximum
Artifact	4	12	3	3	8	2	2	18	4,5	9
Review	4	4	1	1	8	2	2	10	2,5	4
User	4	0	0	0	0	0	0	0	0	0
Venue	4	0	0	0	0	0	0	0	0	0
Features	20	60	3	3	40	2	2	90	4.5	9
LikeA	25	135	5.4	6	40	1.6	2	215	8.6	16
LikeV	25	25	1	1	0	0	0	39	1.56	3
Posts	20	20	1	1	40	2	2	50	2.5	4
Rates	20	68	3.4	4	40	2	2	110	5.5	13
Total	126	324	2.57	6	176	1.4	2	532	4.2	16

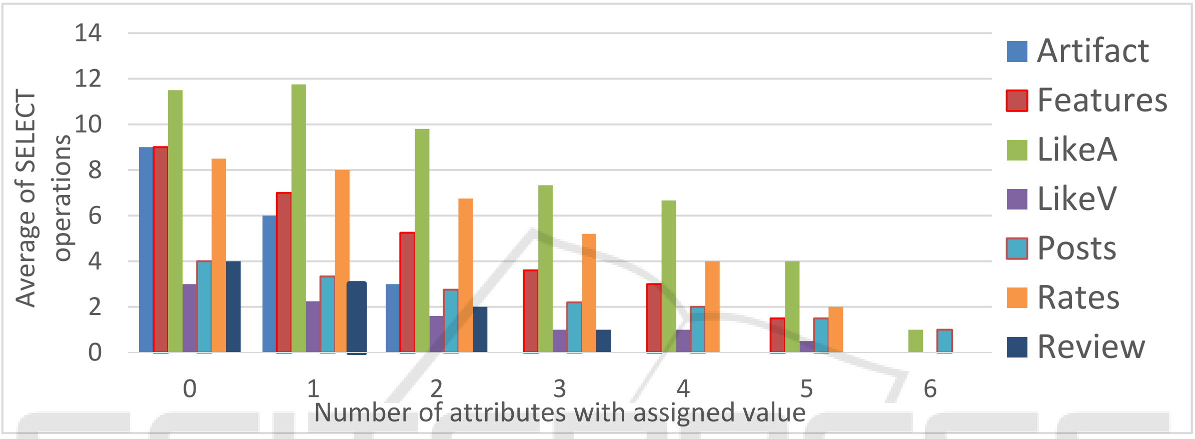


Figure 6: Inverse Relationship between SELECT operations and the number of attribute with assigned value.

The results displayed in Table 1 show that in general a denormalized datamodel requires to determine several operations to keep the logical integrity after a single insertion of a tuple in the conceptual model. For 126 insertions in the conceptual model, we needed 324 INSERT operations, 176 UPDATE operations and 532 SELECT operations to keep the integrity in the logical model. The particular cases of the insertions of tuples of Venue and User, where there are no operations needed, is caused because we cannot insert the information of these tuples alone in any table of the logical model. In order to insert the information of a Venue or a User it needs to be inserted alongside the information of relationships such as LikesV or LikesA, respectively.

We have also detected an inverse relationship between the number of SELECT operations and the number of attributes with assigned value in the tuple. The tuples with the information of entities can contain up to 3 non-key attributes with assigned values while the ones with information of relationships contain up to 6 non-key attributes with assigned values (the combination of the 3 attributes of each entity of the relationship). This inverse relationship is shown in Figure 6 where each bar represents the average of SELECT operations needed for the number of attributes with assigned value in the tuple. We observe how the average of SELECT operations decreases as the number of attributes with assigned value increases.

5 CONCLUSIONS

Nowadays, the use of NoSQL databases is increasing due to the advantages that they give to the processing of big data. When we work with NoSQL databases, we need to use models due to the complexity of the data. In this work we have used two models, the conceptual model and the logical model, for our objective of keeping the consistency in Cassandra. Although Cassandra lacks mechanisms to preserve the integrity of the data, we have proposed in this article a method that automatically keeps the integrity of the data using a conceptual model directly connected to the Cassandra data model (logical model). We have also observed through experimentation that, after an insertion of a tuple, in order to keep the integrity in a logical model with 9 tables we needed up to 6 INSERT operations and 16 SELECT. This shows how complex keeping the integrity of the data can be and that if the number of tables that store the same information increases, the complexity of keeping its integrity also increases. With our automatized method we ensure the data integrity which helps the developer avoiding potential defects.

As future work we want to deepen in the bottom-up approach. We also want to study how to create conceptual models based just in the logical model in order that the systems that were not created with a conceptual model can also use our method.

ACKNOWLEDGMENTS

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

REFERENCES

Apache Software Foundation. (2017) *The Cassandra Query Language (CQL)*. [online]. Available at: http:// cassandra.apache.org/doc/latest/cql/ [Accessed 19-07-2018]

Cattell, R. (2011). Scalable SQL and NoSQL data stores. ACM Sigmod Record, 39, 4 (December 2010)

Chebotko. A, Kashlev, A., Lu, S. (2015). *A Big Data Modeling Methodology for Apache Cassandra*. In IEEE International Congress on Big Data (BigData'15), 238-245, New York, USA, 2015.

Datastax. (2015) *Basic Rules of Cassandra Data Modeling*. [online] Available at https://www.datastax.com/dev /blog/basic-rules-of-cassandra-data-modeling [Accessed 19-07-2018]

Datastax (2015). *New in Cassandra: Materialized Views.* [online] Available at: https://www.datastax.com/dev/ blog/new-in-cassandra-3-0-materialized-views [Accessed 19-07-2018]

Datastax. (2016). *Apache Cassandra*. [online]. Avialable at: http://cassandra.apache.org/ [Accesed 19-07-2018]

Datastax. (2017) *Data consistency* [online]. Available at https://docs.datastax.com/en/cassandra/3.0/cassandra/dml/dmlAboutDataConsistency.html [Accessed 19-07-2018]

Han, J., Haihong, E., Le, G., Du, J. (2011). *Survey on NoSQL database*. In Pervasive computing and applications (ICPCA), 2011 6th international conference on (pp. 363-366). IEEE.

Leavitt, N. (2010). Will NoSQL databases live up to their promise?. Computer, (February 2010), 43, 2.

Li, Y. & Manoharan, S. (2013). *A performance comparison of SQL and NoSQL databases*. In Communications, computers and signal processing, 2013 IEEE pacific rim conference on (PACRIM '13), IEEE, 2013. p. 15-19

Moniruzzaman, A. B. M., Hossain, S. A. (2013). *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison*.

Peter. C. (2015). *Supporting the Join Operation in a NoSQL System. Master's thesis*. Norwegian university of Science and Technology, Norway

Rajanarayanan Thottuvaikkatumana. (2015). *Cassandra Design Patterns, second edition*, ed. Packt Publishing Ltd

Ruiz, D. S., Morales, S. F., Molina, J. G. (2015). *Inferring versioned schemas from NoSQL databases and its applications.* In International Conference on Conceptual Modeling (ER 2015), 467-480. Springer, Cham.

Tauro, C. J., Aravindh, S., Shreeharsha, A. B. (2012). *Comparative study of the new generation, agile, scalable, high performance NOSQL databases*. International Journal of Computer Applications, 48(20), 1-4.