Recovery of Partitioned Databases Based on Time Stamp Data and the Role of CRUD Operations: Two Educational Web Applications

Lj. Kazi*, S. Nadrljanski *, G. Gecin*, A. Kansara**,
Z. Kazi*, B. Radulović*and N. Chotaliya***
* University of Novi Sad, Technical Faculty "Mihajlo Pupin" Zrenjanin, Serbia
** Saurashtra University, Rajkot, MP Shah Arts & Science College, Surendranagar, India
*** Parth Systems ~~LTD~~, Navsari, Gujarat, India
ljubica.kazi@gmail.com, sloba92ki@gmail.com, biljana.radulovic66@gmail.com,
zoltan.kazi@gmail.com, gorangecin@gmail.com, amar.kansara@gmail.com,
narendra_chotaliya@yahoo.com

Abstract – Storage of time stamp value with every record in a table of a database is recomended standard. This way, changes over the set of records can be monitored, by consuming the value of time stamp data. Aim of this paper is to present the use of time stamp data with all CRUD (Create, Read, Update, Delete) operations over the records in a relational database. Particular aim of this paper is to present the use of time stamp data in the context of partitioned databases recovery. The paper describes two web applications that enable performing CRUD operations over partitioned databases and include databases recovery function performed by exploiting time stamp data. These web applications were developed within the educational process in the course Distributed Information Systems at University of Novi Sad, Technical Faculty „Mihajlo Pupin" Zrenjanin, Serbia. Special focus of this paper is to present the impact of CRUD operations to time stamp data and possibilities for using time stamp data for partitioned databases recovery.

I. INTRODUCTION

Contemporary software development is often oriented towards creation and maintenance of distributed information systems, based on creating web and mobile applications. Distributed information systems belong to a broader category of distributed systems [1], with emphasize on distributed data and distributed data processing [2].

In aim to enhance students' knowledge and skills, so they get eligible and profession ready before enter professional environment, higher education courses are designed and regularly updated according to technology trends. University of Novi Sad, Technical Faculty "Mihajlo Pupin" Zrenjanin, Serbia (in the rest of paper: @TFZR) have included the course Distributed Information Systems in the educational process within the Master studies in Information Technology [3]. This course has been established at this Faculty for more than a decade, starting with the accredited Master studies in IT. Since then, the course evolved, particularly in the teaching content and methods. Several years ago, particular educational software has been created as an example for students – a web application that has all important components implemented and functional, to illustrate the most important educational concepts. In the rest of the paper it will be named DIS@TFZR.

Aim of this paper is to emphasize recent improvement of the DIS@TFZR web application related to dis. The DIS@TFZR software solution and its' distributed databases recovery module based on time stamp data has already been presented in [4]. This paper presents new results in the improvement of the DIS@TFZR solution, that are related to all CRUD operations support in the solution, with particularly addressed concerns related to the role of time stamp in these operations.

The rest of the paper is organized as follows: next section is related to theoretical background related to distributed systems, distributed information systems, distributed databases and time stamp. Third section presents related work in database recovery and time stamp applications. Fourth section describes the basic version of DIS@TFZR software solution, including the latest improvements with data consistency checking upon all database partitions and data recovery. Fifth section explains the second example of software solution, while the section six emphasizes the timestamp value role in CRUD operations and the relationships between the timestamp values in central and partitioned databases. Final section provides conclusions and future work directions.

II. THEORETICAL BACKGROUND

Distributed system could be defined as a collection of independent computers that appear to users as a coherent complete system [1]. The most important features of distributed systems include autonomy for each node in the system, multi-component and multi-layered system, (non-) transparency to users, scalability, reliability, openness, flexibility and others. Certain advantages of creating and using distributed systems include increased availability of resources, improved collaboration, reliability and availability, as well as increase of system performance. Risks related to the use of distributed systems include safety of data, data privacy, reliability of nodes connections, compatibility and synchronization of processes and data. Distributed systems consist of distributed computing systems (grid and cluster), distributed information systems and distributed embedded systems.

Distributed information system is defined as an information system that supports distributed data processing, over the distributed database [2]. Distributed data processing is usually supported by modular approach, with software components, class libraries and services. Distributed data includes data formats for data exchange between software components (such as XML, JSON) and distributed databases.

"A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network. The database is physically distributed across the data sites by fragmenting and replicating the data. Given a relational database schema, fragmentation subdivides each relation into horizontal (by selection operation) or vertical (by projection operation) partitions."[5] According to [5], benefits of fragmentation are in reducing data transmission costs, with reducing size of data within user queries and potentially closer proximity to the location of use.

Time stamp is defined as a data member that is closely related to time of creation of a data record or a message [6]. According to [6], the time stamp data is usually created according to local time at the place of creation. Time stamp values are created in an infinite row and they could be used for organization of records or messages, their grouping, ordering and prioritization. Time stamp is also defined and described as "a unique identifier created by the DBMS to identify a transaction. Timestamp values are assigned in the order in which the transactions are submitted to the system, so timestamp can be thought of as the transaction start time." [7]

III. RELATED WORK

A. *Database recovery*

The problem of database recovery has been addressed in both professional and educational environments [8]. Some of techniques used in database recovery included transactions processing [9], but the data recovery also addressed keeping the whole database in a main memory [10]. Information systems technology advancements introduced the same problem within the client/server [11] and online [12] systems, as well as with partitioned distributed databases [13] [14], by using database catalog (journal) [15].

Other significant aspects and methods are related to using online mirror database [16], performance metrics [17], recovery process acceleration [18], formalization [19] and cost estimation [20]. Finally, with the introduction of mobile technologies, database recovery techniques for Android have also been addressed [21]. A particular concern has been examined in the case of combined data store, consisting of a (relational) database and files (such as multimedia files), with special focus on issues related to backup/recovery and data consistency [22].

B. *Time Stamp Applications*

Time stamp as a term has been examined in the context of relationship between the real-world fact occurrence time (validity of a fact that describes an event or characteristics of an object) and the time of recording of that fact in a database. The first is assigned a "valid time-stamp" and the second is named "transaction time-stamp" [23]. Time stamping is not related only with databases, but they are used generally with diversity of digital document types, such as text, audio, photos, video files and others, in aim to keep and maintain additional data related to the time of creation and modification of these files [24]. Time stamping is found useful in cases of synchronization of multiple data sources for video data, particularly in data acquisition and telecommunication application areas. [25] Another beneficial use of time stamping is related to measurement systems, where accurate time of obtaining measurement data is crucial. [26]

Time stamping has many application areas, particularly in distributed systems. Such systems are collaborative virtual environments [27]. Specially demanding is an orientation towards optimizing timestamp management, in the case of data streams, where simultaneous events occur and their data records tuples have exactly the same value of timestamp. [28] Especially important is using time

stamp data in collective resource acquisition in a distributed system, with the simultaneous access to all requested resources [6]. Very interesting problem is related to dynamism with real-time databases, where data are obtained and serialized, with the use of timestamp value [29].

IV. BASIC EDUCATIONAL SOFTWARE DIS@TFZR

In the last few years, a specific web application (named DIS@TFZR) has been developed for the educational purpose, as an example of distributed information systems software. Previously mentioned software DIS@TFZR has been presented with all models, user interface and other details in [4]. It consists of structural and functional elements to illustrate fundamental concepts of a distributed information system:

1. *Distributed data*, with horizontally and vertically partitioned databases and the use of data exchange formats,
2. *Distributed processing*, with n-tier layered structure of software and the creation and usage of web services.

Particularly, the developed software is ASP.NET web application that consists of layers:

1. *Data Access Layer* – class library with classes that are used for data processing into relational database, as well as data retrieval from the database.
2. *Business Logic Layer* – class library with business-domain named classes, which have methods to make business-related computations, automatisms and enforce business rules.
3. *Service Layer* – consists of: A) web service (SOAP type) which enables additional data and functionality to the business logic layer, B) class library with mapper classes, providing a bridge between different layers and overcoming differences between their data models.
4. *Presentation Layer* – ASP.NET user interface, Presentation Logic as a Class library that obtains input data, validates data according to basic validations and business-related validations and transforms input data in the form suitable for database processing.

Previously mentioned software enables storing data in multiple databases, which are logically connected. These databases represent horizontal and vertical partitions of the central database. This way, the developed web application stores data in the central database, as well as to several horizontal and vertical partitions. The application uses XML file with the data related to all databases (both central and all partitions), in aim to represent a distributed database catalog. Once the data entered, the record is transformed into a set of "Insert into" queries and, within a loop, each database from a database catalog is populated with appropriate data, by processing adequate SQL query.

Regarding functional aspect of DIS@TFZR software solution, there is support to:

- *Data entry* with usual data input form, where entered data is sent to multiple databases for their storage. Horizontal partitions get only the rows that suit their criteria of classification, while all vertical partitions get each row, but only according to particular part of the table structure.
- *Data presentation* in tabular form, where horizontal partitions represent subsets of rows with all equal structure and the data are merged at the memory level (by merging datasets), so the user do not recognize the fact that data come from multiple databases
- *Data consistency check and data recovery* – supported at separate user interface form, where a user could start checking if each database partition has adequate records, comparing to the central database. After checking, the missing records and appropriate databases with the missing records are determined. Here, time stamp data are used to determine which data records are missing, by extracting the last timestamp value for each database (central and all partitions) and comparing these values.

The DIS@TFZR software is created upon the simple domain of keeping records about the preschool children.

V. SECOND WEB APPLICATION WITH EXTENSION TO SUPPORT ALL CRUD OPERATIONS

Within the practical pre-exam requirements at course Distributed Information Systems @TFZR, students used the previously developed example DIS@TFZR as a template for their own pre-exam work. Since the existing solution of DIS@TFZR software, before school year 2019/20 did not have the module for partitioned databases consistency checking and recovery, based on timestamp data, during regular and online classes in school year 2019/20 students and Ljubica Kazi together improved the previous DIS@TFZR version with adding this functionality. This way, students of 2019/20 generation could use an improved version for their own pre-exam required projects.

One of such solution has been developed by Slobodan Nadrljanski upon the improved version of

DIS@TFZR as a template. The second example, created by Slobodan Nadrljanski, has an improved functionality (comparing with DIS@TFZR template), by adding functions for updating (changing) and deleting records in the central database and all partitions. This way, the developed students' solution provided support for all CRUD operations. The developed solution was firstly used as a pre-exam required project. Later, with additional improvements, the software was included in the final master work, defended @TFZR [30]. Business domain that has been used for this example was related to registration of a new company.

Listing 1. explains the principle, firstly developed in DIS@TFZR template, with the example applied in [30]. First part presents CREATE table SQL script and second presents extracting time stamp value and transforming into bigint type in a horizontally partitioned database. Figure 1 and 2. Present results from [30]. Figure 1. presents a data entry form for adding new data about the company to be registered, while Figure 2. shows tabular presentation of merged data from different horizontal partitions, where each row has links to enable: a) loading particular data to another form in aim to be changed (updated); b) deleting a selected record.

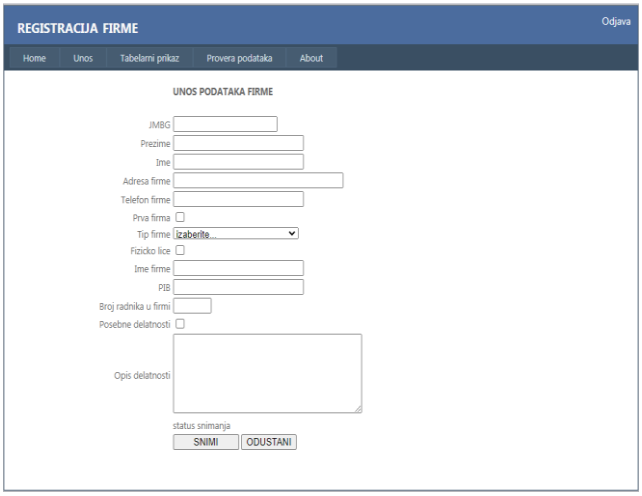

Figure 1. Data entry form for a new company registration [30]

CENTRAL DATABASE

```
CREATE TABLE [dbo].[FIRMA](
[JMBG] [nvarchar](40) NOT NULL,
[Prezime] [nvarchar](40) NOT NULL,
[Ime] [nvarchar](40) NOT NULL,
[AdresaFirme] [nvarchar](60) NOT NULL,
[TelefonFirme] [nvarchar](60) NOT NULL,
[PrvaFirma] [bit] NOT NULL,
[TipFirme] [nvarchar](60) NOT NULL,
[FizickoLice] [bit] NOT NULL,
[ImeFirme] [nvarchar](60) NOT NULL,
[PIB] [nvarchar](60) NOT NULL,
[BrojZaposlenih] [int] NOT NULL,
[PosebneDelatnosti] [bit] NOT NULL,
[OpisDelatnosti] [nvarchar](100) NOT NULL,
[BrojacPoslednjeIzmene] [timestamp] NOT NULL)
```

Copy time stamp value

HORIZONTAL PARTITION DATABASE

```
CREATE TABLE [dbo].[FIRMA](
        [JMBG] [nvarchar](40) NOT NULL,
        [Prezime] [nvarchar](40) NOT NULL,
        [Ime] [nvarchar](40) NOT NULL,
        [AdresaFirme] [nvarchar](60) NOT NULL,
        [TelefonFirme] [nvarchar](60) NOT NULL,
        [PrvaFirma] [bit] NOT NULL,
        [TipFirme] [nvarchar](60) NOT NULL,
        [FizickoLice] [bit] NOT NULL,
        [ImeFirme] [nvarchar](60) NOT NULL,
        [PIB] [nvarchar](60) NOT NULL,
        [BrojZaposlenih] [int] NOT NULL,
        [PosebneDelatnosti] [bit] NOT NULL,
        [OpisDelatnosti] [nvarchar](100) NOT NULL,
        [CentralniBrojacPoslednjeIzmene] [bigint] NOT NULL)
```

TRANSFORMATION OF TIMESTAMP VALUE TO BIGINT

```
DataSet dsIntTimeStamp = new DataSet();
Int64 intVrednostTimeStamp = 0;
 string SQLUpit="";
SQLQuery = "select top 1 CAST(BrojacPoslednjeIzmene as BIGINT) as IntTimeStamp from FIRMA order by BrojacPoslednjeIzmene desc";
dsIntTimeStamp = objTabela.DajPodatke(SQLQuery); // method that retrieves data from database by executing SQLQuery
intValueTimeStamp = Int64.Parse(dsIntTimeStamp.Tables[0].Rows[0].ItemArray[0].ToString());
```

NOTE: bigint value inValueTimeStamp will be inserted into all partitioned databases having the same name and value under field: „CentralniBrojacPoslednjeIzmene"

Listing 1. The structure of a data table in the central database and horizontal partitioned databases and copying the value of timestamp [30]

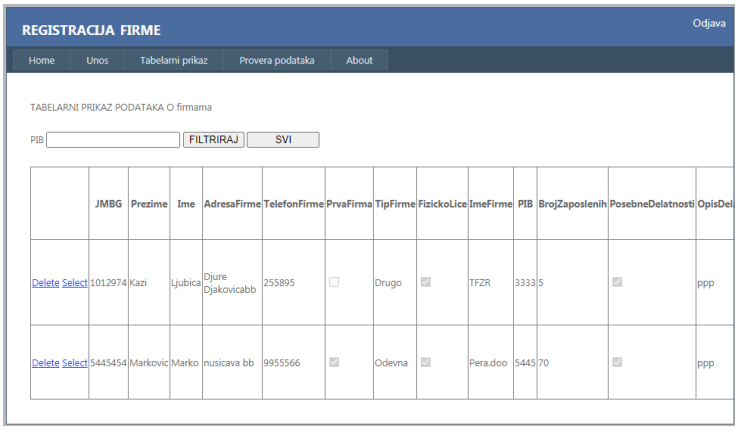

Figure 2. Tabular presentation of merged data from horizontal partitions with added links for delete and select (leading to update) [30]

VI. THE IMPACT OF CRUD OPERATIONS TO TIME STAMP DATA AND THE ROLE OF TIME STAMP DATA IN PARTITIONED DATABASES RECOVERY

Aim of this section is to explain the use and change of time stamp data in all CRUD operations performed upon the central database and distributed database partitions. It is briefly explained in Table 1.

Table 1. CRUD operations and time stamp data in partitioned databases

CREATE operation
1. Central database gets the input data (all except time stamp data) and in the very moment of inserting of all non-timestamp data to DBMS, it automatically creates timestamp value and stores it in timestamp type of field. Timestamp field could not be reached by insert into query, so the insert into query includes all non-time stamp fields. 2. Each horizontal and vertical database partition gets adequate data and exactly the same value of time stamp, stored in integer data type field.
READ operation
Time stamp data could be used in SQL queries of select type, but their value do not change while data retrieval.
UPDATE operation
1. When Update query is performed upon the record in the central database, a new value of timestamp is automatically generated by DBMS and replaces old value. 2. All database partitions should receive new regular data, as well as new timestamp data. This way, all database partitioned databases remain up-to-date with data and comparable with the central database.
DELETE operation
1. Delete operation removes a record from the central database. The time stamp value for any other data record added after the deletion is automatically generated and does not continue to the last value in the list, as well as do not fill the gaps made when some record is deleted in the middle position of all records. 2. It is necessary to remove all records from all horizontal and vertical partitioned databases, to make them be up-to-date with the recent deletion in the central database. Note: Existing module for database recovery is not developed in direction from partitioned databases to the central database, but in the opposite direction. If this feature was developed, then the deleted row from the central database could be recovered by inserting the missing data, but in that case, new timestamp value would be generated and all partitioned databases should then update their time stamp values for the particular record.

To illustrate previously mentioned role of CRUD operations in timestamp creation and update, as well as the timestamp value alignments within the distributed database recovery, Figure 3 presents a simplified data table with regular and time stamp data, with the illustration of the impact of data row update to the value of time stamp. Figure 4. presents the impact of delete and insert SQL statements on time stamp data.

Obviously, in previously mentioned examples, when updating name from “Ljubica” to “Ljubica1”, the value of time stamp data changed. After deletion of “Marko” record, new record, inserted immediately after that, gets the time stamp value as incremented (+1) comparing to the last record being created or updated (in this case, the last operation was update, so that record keeps the last timestamp value, to be used with increment to the new record to come).

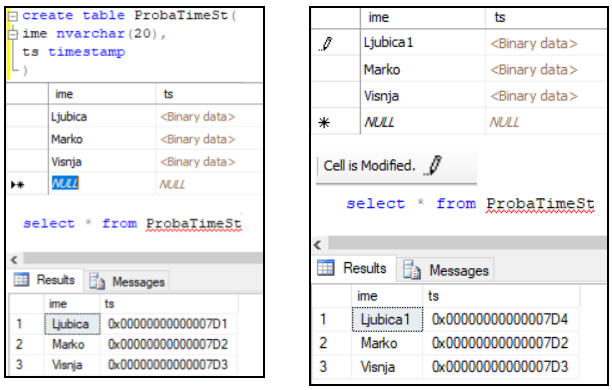

Figure 3. Simplified example of a table with regular and time stamp data and the impact of UPDATE operation to time stamp value

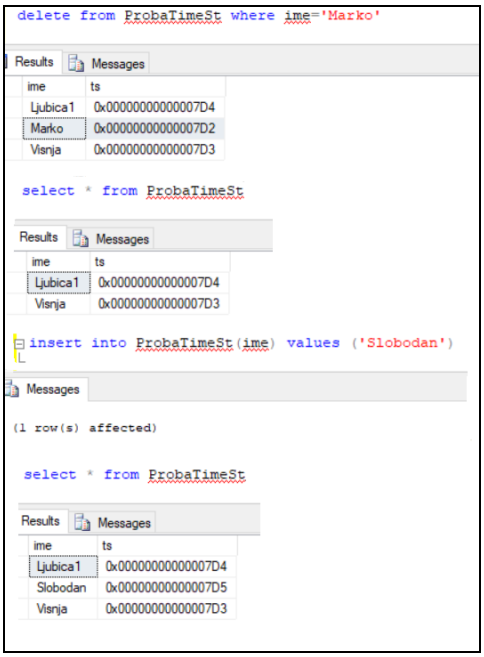

Figure 4. Simplified table with regular and timestamp field – impact of DELETE and INSERT (performed after DELETE) operations to time stamp value

VII. CONCLUSION

Distributed databases usually are created with database partitioning. Horizontal partitioning enables creation of tables with exactly the same structure, but data is categorized before sending to a particular partition. Vertical partitioning enables splitting the structure of a complex database table into multiple tables having different structure, but the same primary key value. In keeping all databases partitions up-to-date with records in central database, usually time stamp data is used.

Aim of this paper was to present the impact of CRUD operations to time stamp data and to explain how the time stamp data could be used for synchronization of data between the central database and partitioned databases. This way, the concept of alignment of central database with all partitions could be used in distributed databases recovery.

This paper presents educational software that was created for the purpose of teaching distributed information systems at master level course in information technologies at University of Novi Sad, Technical Faculty "Mihajlo Pupin" Zrenjanin. Two examples of the developed solution have been presented – initial software DIS@TFZR, created in collaboration of Ljubica Kazi with students, during classes (with only data entry and tabular presentation functionalities, improved with distributed databases checking and recovery) and the second example of students' master thesis work (with other business domain and additional options for delete and update upon data in central database and all database partitions).

The presented solutions demonstrate the creation and use of time stamp data in distributed databases recovery, with special emphasize on all CRUD operations impact on the time stamp data.

REFERENCES

[1] A. Tanenbaum A, M. Van Steen: "Distributed systems, Principles and Paradigms", VrijeUniversiteit, Amsterdam, Pearson Prentice Hall, 2007.

[2] P. Mogin, I. Lukovic, M. Govedarica: "Principles of database design" (In Serbian: "Principi projektovanja baza podataka"), Fakultet tehničkih nauka, Novi Sad, 2000.

[3] Web page for Distributed Information Systems course at Technical Faculty "Mihajlo Pupin" Zrenjanin, http://www.tfzr.rs/Predmet/distribuirani-informacioni-sistemi

[4] Lj. Kazi, A. Kansara, B. Radulović, I. Berković, G. Gecin, S. Nadrljanski, V. Šašić: "Web-Based Educational Software for Teaching Distributed Information Systems with Partitioned Database Recovery", International conference Applied Internet and Information Technologies, AIIT2020, October 16, 2020, Zrenjanin, Serbia (accepted)

[5] M. T. Ozsu, P. Valduriez: "Distributed and Parallel Database Systems", ACM Computing Surveys, Vol. 28, No. 1, March 1996.

[6] Q. Sun, Zhang H, Zhang J: "A Time-stamp Based Solution for Collective Resource Acquisition in a Distributed System", Proceedings of the 33th Hawaii International Conference on System Sciences, 2000.

[7] M. Kaur, H. Kaur: "Concurrency Control in Distributed Database System", International Journal of Advanced Research in Computer science and Software Engineering, Vol 3, Issue 7, July 2013.

[8] Kumar V, Son S.H: "Database Recovery", Springer Science and Business Media, 1998.

[9] Hearder T, Reuter A: "Principles of Transaction-Oriented Database Recovery", Computing Surveys, Vol 15, No 4, 1983.

[10] Eich M. H: "A classification and comparison of main memory database recovery techniques", IEEE Third International conference on Data Engineering, 1987.

[11] Mohan C, Narang I: "ARIES/CSA: a method for database recovery in client-server architectures", SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data, May 1994, pp. 55–66.

[12] Moore D.W et al: "Database recovery to any point in time in an online environment utilizing disaster recovery technology". US Patent - US 6.732,123 B1, 2004.

[13] Hsiao H-I, Chang A, "Method for providing database recovery accross multiple nodes", US Patent - US 6.247.023 B1, 2001.

[14] Dhamankar et al: "Consistent database recovery accross constituent segments", US Patent – US 8.671.085 B2, 2014.

[15] Nishikawa et al: "Database recovery method applying update journal and database log", US Patent – US 7.471.139.B2, 2008.

[16] Hart, D.R: "System and method for database recovery using a mirrored snapshot of an online database", US Patent – US 6.983.295 B1, 2006.

[17] Pommerenk et al: "Performance metric-based selection of one or more database server instances to perform database recovery", US Patent – US 7.809.690 B2, 2010.

[18] Stagg M.R: "Apparatus and method for acceperating database recovery", US Patent – US 7.865.471 B1, 2011.

[19] Gurevich Y, Soparkar N, Wallace C: "Formalizing database recovery", Journal of Universal Computer Science, vol.3, no 4, 1997.

[20] Sauer C, Graefe G, Harder T: "An empirical analysis of database recovery costs", RDSS, ACM, 14. June 2014, Snowbird, UT, USA

[21] Hu X, Wu H: "Database management strategy and recovery methods of Android", IEEE 5th International Conference on Software Engineering and Service Science, 2014.

[22] S. Bhattacharya, C. Mohan, K.W. Brannon, I. Narang, H-I Hsiao, M. Subramanian: "Coordinating Backup/Recovery and Data Consistency Between Database and File Systems", ACM SIGMOD'2002, June 4-6, 2002, Madison, Wisconsin, USA

[23] C. J. Jensen, R.T. Snodgrass: "Temporal Specialization", Proceedings of IEEE Eighth International Conference on Data Engineering, 1992., pp. 594-603.

[24] S. Haber, W.S. Stornetta: "How to Time-Stamp a Digital Document", In: A. J. Menzes and S.A. Vanstone (Eds.): Advances in Cryptology – CRYPTO '90, LNCS 537, pp. 437-455, Springer-Verlag Berlin Heidelberg, 1991.

[25] H-S Yang, B. Kupferschmidt: "Time Stamp Synchronization in Video Systems", International Telemetering Conference Proceedings, Volume 46, 2010.

[26] S. Moreno-Tejera, L. Ramirez-Santigosa, M. A. Silva-Perez: "A proposed methodology for quick assessment of timestamp and quality control results of solar radiation data", Renewable Energy, 78 (2015), 531-537.

[27] S-J Kim, F. Kuester, K.H. Kane Kim: "A global timestamp-based approach to enhanced data consistency and fairness in collaborative virtual environments", Multimedia Systems 10: 220-229, 2005.

[28] H. Thakkar, C. Zaniolo, H. Wang: "Optimizing Timestamp Management in Data Stream Management Systems", IEEE 23rd International Conference on Data Engineering, 2007.

[29] J. Lindstrom, K. Raatikainen: "Dynamic Adjustment of Serialization Order Using TimeStamp Intervals in Real-Tme Databases", In Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications,IEEE Computer Society Press, 1999.

[30] S. Nadrljanski: "Techniques of distributed database recovery in web environment" (In Serbian: "Tehnike oporavka distriburane baze podataka u veb okruženju"), Master thesis, menthor Ljubica Kazi, University of Novi Sad, Technical Faculty „Mihajlo Pupin" Zrenjanin, September 2020.