Automatic and Graceful Repairing of Data Inconsistencies Resulting from Retroactive Updates in Temporal Xml Databases

Hind Hamrouni, Zouhaier Brahmia and Rafik Bouaziz
Department of Computer Science, Faculty of Economics and Management, University of Sfax, Sfax, Tunisia

Keywords: Temporal XML Databases, Retroactive Update, Data Inconsistency, Inconsistency Periods, Repairing an Inconsistency, Side Effect, Side Effects Recovery.

Abstract: In temporal XML databases, a retroactive update (i.e., modifying or deleting a past element) due to a detected error means that the database has included erroneous information during some period and, therefore, its consistency should be restored by correcting all errors and inconsistencies that have occurred in the past. Indeed, all processing that have been carried out during the inconsistency period and have used erroneous information have normally produced erroneous information. In this paper, we propose an approach which preserves data consistency in temporal XML databases. More precisely, after any retroactive update, the proposed approach allows (i) detecting and analyzing periods of database inconsistency, which result from that update, and (ii) repairing of all inconsistencies and recovery of all side effects.

1 INTRODUCTION

Nowadays, supporting the temporal aspect is a requirement for most computer applications, including processing of scientific and census data, banking and financial transactions, record-keeping and scheduling applications. In fact, these applications need to store and manipulate data while taking into account the time dimension; this has led to the appearance of temporal databases (Etzion *et al.*, 1998, Grandi, 2014) which retain data evolution over transaction-time dimension and/or valid-time dimension (Jensen *et al.*, 1998):

- The valid-time of a datum is the time when a datum is true in the real world; each time-varying datum is timestamped with a validity start time (VST) and a validity end time (VET).
- The transaction-time of a datum is the time when a datum is current in the database; each time-varying datum is timestamped with a transaction start time (TST) and a transaction end time (TET).

In temporal databases, there are three types of updates concerned with the time when updates are made: retroactive, proactive (Etzion *et al.*, 1994), and real-time (or on-time) updates.

- A retroactive update is done after the change occurred in reality (i.e., the TST of the datum is superior to its VST).
- A proactive update is done before the change occurs in reality (i.e., the TST of the datum is inferior to its VST).
- A real-time update is done when the change occurs in reality (i.e., the TST of the datum is equal to its VST).

Retroactive and proactive updates occur naturally in many applications. For example, a salary increase may be retroactive to some past date, and a postdated check is a proactive update.

On the other hand, currently, XML databases (Bourret, 2005) are widely used, especially on the web. The introduction of temporal (Dyreson *et al.*, 2009) aspects in such databases gave rise to temporal XML databases (Brahmia *et al.*, 2014). In these databases, any temporal XML document can store transaction-time, valid-time and bi-temporal XML elements. Moreover, these databases are very useful for several domains (e.g., managing evolution of legal texts in e-government systems, online management of patients' medical records...). Notice that temporal XML databases are richer than temporal relational databases at structure and textual content levels. Moreover, temporal XML data are presented with temporally grouped data models. Indeed, each time-varying XML element evolves individually over time.

However, although temporal XML databases allow end users/applications to perform retroactive updates on stored data, such updates are not always performed safely since they could have a harmful effect on the consistency of the database. Let's take the example of correcting a past banking interest rate in a bank, which was applied during the period going from 2013-01-01 to 2013-12-31. All existing data (e.g., interest bank accounts, balances of bank accounts, scheduled payment amounts) that have been obtained using the erroneous past banking interest rate are consequently erroneous and should be corrected. The database was inconsistent during the period where this interest rate was effective.

In this paper, we focus on the impacts of retroactive updates on the consistency of the database and we propose an approach that allows the temporal XML database management system (DBMS) (i) to detect any database inconsistency that happens due to a retroactive update, and (ii) to perform automatically the necessary processings, in a transparent way, in order to repair the detected inconsistencies.

The rest of this paper is organized as follows: the next section motivates the need for a new approach for preserving the consistency of temporal XML databases; Section 3 describes data inconsistencies resulting from retroactive updates in such databases; Section 4 presents our approach for an automatic and graceful repairing of data inconsistencies that occur due to retroactive updates; Section 5 discusses related work; Section 6 concludes the paper.

2 MOTIVATION

In this section, we first present an example that illustrates how maintaining consistency in temporal XML databases after a retroactive update is a complicated task that could not be achieved using existing supports provided by DBMSs. Then we show the need for systems providing supports for preserving consistency of temporal XML databases.

2.1 Motivating Example

Suppose that on 2014-03-25, the auditor detects an error that has occurred on 2014-01-05: the bank employee saved a deposit transaction which adds an amount of 550 TND (the erroneous value) instead of 500 TND (the correct value), to the account of a customer; thus, an amount of 50 TND was stored in the database but really was not provided by the customer. Obviously, this error was propagated to all other financial transactions that have been done on this account between 2014-01-05 and 2014-03- 25; there was always an amount of 50 TND which should be subtracted from the balance.

The semantics of existing data update operations (Brahmia *et al.*, 2009), which should be used to correct both the deposited amount and the balance account on 2014-01-05, does not support the correction of the impact of this error (i.e., to correct each balance of this account, related to each transaction performed after 2014-01-05). Such an operation corrects only the details of the financial transaction (i.e., the deposited amount and the balance of the account at the end of the transaction) done on 2014-01-05. To restore the database consistency, the database administrator should proceed in an ad hoc manner: first, he/she should determine the list of all transactions that were performed on this account going from the transaction during which the error has occurred until the last one. Then, he/she should update all erroneous data by writing an appropriate XML update (Tatarinov *et al.*, 2001; W3C, 2011; Hamrouni, 2012).

2.2 Need for New DBMS Supports

The consistency of a temporal XML database could not be ensured easily, since (i) all temporal dimensions are supported (i.e., data can evolve over transaction time and/or valid time), and (ii) a data management operation that is originally devoted to insert, delete, or update an XML element could involve several other XML elements (e.g., when the temporal interval of a new element that modifies an existing one overlaps, completely or partially, temporal intervals of other existing XML elements). Furthermore, since the database administrator lacks methods and tools needed to automate the activity of detecting and repairing data inconsistencies in temporal XML databases, such an activity remains an error-prone and time-consuming undertaking.

Thus, the challenges described above show that end users/applications, using temporal XML databases, need DBMSs with built-in support for detecting and repairing automatically inconsistencies which result from retroactive updates.

3 DATA INCONSISTENCIES RESULTING FROM RETROACTIVE UPDATES

In (Bouaziz *et al.*, 1998), the authors defined an inconsistency period resulting from a retroactive update of data as the temporal interval which delimits the scope of side effects that are expected to be generated by this update. Such an inconsistency period could be of one of the following three types: "**Wrong Absence of Data**", "**Wrong Presence of Data**", or "**Errors in Data**".

- Wrong Absence of Data: the inconsistency is due to the absence of a datum that had to be present in the database during this period;
- Wrong Presence of Data: the inconsistency comes from the presence of a datum that had to be absent in the database during this period;
- Errors in Data: the inconsistency results from the existence of some data with erroneous values during this period.

An inconsistency period, resulting from a retroactive update can be divided into several sub-periods; each one of these sub-periods should be interpreted according to the nature (i.e., insertion, deletion, or correction) of the retroactive update. In the following, we study periods of inconsistency, and their sub-periods.

3.1 Data Inconsistencies Resulting from a Retroactive Insertion of Data

The insertion of an XML element with retroactive effect generates an inconsistency period of "Wrong Absence of Data" type: the inserted element, which was absent before its transaction start time, should be normally present in the temporal database since its validity start time. Fig. 1 illustrates such a period of inconsistency. In the following, we use CT to denote the "current time".

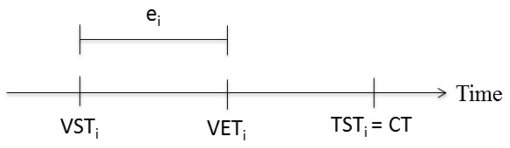

Figure 1: The inconsistency period resulting from a retroactive insertion of data.

As shown by Fig. 1, the period of inconsistency, which is delimited by the VST (period beginning) and the TST (period ending) of e_i, can be divided into two sub-periods:

- $[VST_i - VET_i]$: the interval during which the consequent side effects concern all processings that had to use the element e_i while it had to be a current element;
- $]VET_i - TST_i]$: the interval during which the generated side effects concern all processings that had to use the element e_i while it had to be a past element.

3.2 Data Inconsistencies Resulting from a Retroactive Deletion of Data

The removal of an XML element with retroactive effect generates an inconsistency period of "Wrong Presence of Data" type: the deleted element, which was present before the instant of its deletion (i.e., before the TST of the deletion element (Brahmia *et al.*, 2009) which is used to perform this deletion), should not normally exist in the temporal database since its validity start time. Fig. 2 illustrates such a period of inconsistency.

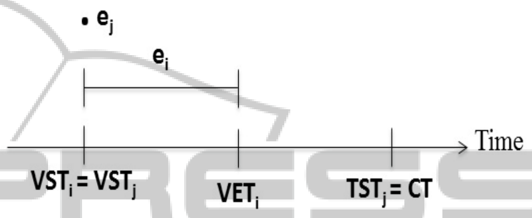

Figure 2: The inconsistency period resulting from a retroactive deletion of data.

As shown by Fig. 2, the period of inconsistency, which is delimited by the VST of e_i (period beginning) and the TST of e_j (period ending), can be divided into two sub-periods:

- $[VST_j - VET_i]$: the interval during which the resulting side effects concern all processings that had used the element e_i while it was a current element;
- $]VET_i - TST_j]$: the interval during which the consequent side effects concern all processings that had used the element e_i while it was a past element.

3.3 Data Inconsistencies Resulting from a Retroactive Correction of Data

Retroactive correction operations could be done only on valid-time and bitemporal data. In this paper, we deal only with retroactive correction of bitemporal data, since we think that it is more complex and, thus, it requires much attention.

The correction of a bitemporal element is performed by inserting a new element containing the correct values, called the element of correction (Brahmia *et al.*, 2009). A correction operation can affect (i) the contents of the corrected element, (ii) values of non-temporal attributes (i.e., attributes different of VST, VET, TST, and TET attributes) of the corrected element, and/or (iii) the valid-time interval of the corrected element (i.e., values of VST and VET attributes); obviously, the transaction-time interval (i.e., values of TST and TET attributes) of any element cannot be modified owing to the definition of transaction time. In the first and/or the second case (i.e., points (i) and (ii)), the correction operation generates an inconsistency period of type “Errors in Data”. However, in the third case (i.e., point (iii)), it generates an inconsistency period which can be divided into several sub-periods each one of them has a different type.

In the following, we deal with inconsistencies resulting from a retroactive correction operation that updates the contents and/or the values of non-temporal attributes of a bitemporal element; it modifies neither the VST attribute, nor the VET attribute. This correction generates an inconsistency period of type “Errors in Data”: it means that the corrected element had an erroneous value. Fig. 3 illustrates such a period of inconsistency.

When the valid-time interval of a bitemporal element is modified, we distinguish ten cases according to Allen's interval algebra (Allen, 1983) and all possible relations between the valid-time interval of the correction element (e_j) and that of the corrected element (e_i); more details on all these cases can be found in (Hamrouni *et al.*, 2014).

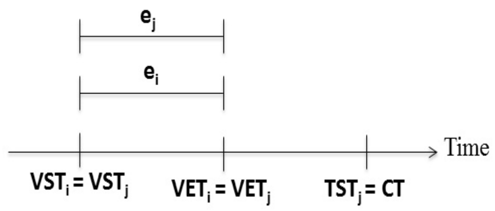

Figure 3: The inconsistency period resulting from a retroactive correction operation which does not modify the valid-time interval of a bitemporal element.

As shown by Fig. 3, the period of inconsistency, which is delimited by the VST of e_i (period beginning) and the TST of e_j (period ending), can be divided into two sub-periods:

- [VST_i - VET_i]: the interval during which the generated side effects concern all processings that had used the element e_i while it was a current element;
-]VET_i - TST_j]: the interval during which the consequent side effects concern all processings that had used the element e_i while it was a past element.

4 DETECTING AND REPAIRING DATA INCONSISTENCIES RESULTING FROM RETROACTIVE UPDATES IN TEMPORAL XML DATABASES

In this section, we propose an approach that allows restoring automatically the database consistency after a retroactive update of temporal XML data. First, we describe the process of detecting and repairing automatically database inconsistencies. Then, we present the architecture of a native temporal XML DBMS which provides supports for an automatic detection and repair of data inconsistencies after a retroactive update.

4.1 Process of Detecting and Repairing Automatically Data Inconsistencies

When an end user or an application submits to the temporal XML DBMS a retroactive update of temporal XML data, the DBMS performs the following sequence of tasks:

Task 1: it updates the database as required by the end user or the application (obviously after checking the update syntactically).

Task 2: it determines automatically the period of inconsistency resulting from the retroactive update of data, and its sub-periods.

Task 3: it determines automatically the list of transactions that were executed during each sub-period of inconsistency and had used erroneous past data (in case that the corresponding sub-period of inconsistency is of type “Wrong Presence of Data” or “Errors in Data”) or had to use new data (in case that the corresponding sub-period of inconsistency is of type “Wrong Presence of Data” or “Errors in Data”); for each concerned transaction, it should provide its commit time, all its elementary operations (for the sake of simplicity, we suppose that a transaction is composed of a single operation, i.e., a single insert, delete, or update operation), and all data that were written and read by this transaction.

Task 4: it re-executes in a provisory workspace the list of corresponding transactions during each sub-period of inconsistency either (a) without using the corresponding past data, if this sub-period is of type “Wrong Presence of Data”, or (b) while using (b.1) the correct values of past data, if this sub-period is of type “Errors in Data”, or (b.2) the specified past data, if this sub-period is of type “Wrong Absence of Data”.

Task 5: it compares the old results of determined transactions (i.e., results already stored in the database, as written data by these transactions) with the new results of them (i.e., the new data that are written by these transactions in the provisory workspace).

Task 6: it replaces every old result with the corresponding new result when there is a difference between them.

In the following, we provide main requirements of some tasks presented above:

- Task 1 requires that the DBMS supports management of temporal XML data under schema versioning; we have studied this aspect in our previous work (Hamrouni, 2012).
- Task 3 requires beforehand keeping track of all transactions which are executed: all operations which compose each transaction, all written and read data, and its commit time;
- Task 4 requires that the provisory workspace should be a copy of the database (schema and instances) during the inconsistency period, before the execution of the retroactive update operation.
- Task 6 requires that replacing old data with new data should be performed logically and not physically (i.e., in a destructive manner); each existing erroneous data is logically corrected by a new correct data. After restoring the database consistency, only correct data must be used by the DBMS to answer user/application queries; erroneous data could be vacuumed later (Skyt *et al.*, 2003) by the database administrator.

4.2 Architecture of a DBMS Supporting Detection and Repair of Data Inconsistencies

Owing to the general architecture of a DBMS (Hellerstein *et al.*, 2007), the transaction manager is the component which is devoted to managing transactions resulting from user/applications queries and updates submitted to the database. Therefore, if we would like to have an automatic restoring of the database consistency after any retroactive update of temporal XML data, we think that (i) the transaction manager of a temporal XML DBMS should be extended by four new components: "Retroactive Update Checker", "Inconsistency Period Manager", "Side Effect Recovery Manager" and "Optimizer", and (ii) the temporal XML DBMS itself should include a "Transaction Catalog Manager", a "Transaction Catalog", and a "Provisory Workspace". The new general architecture of a temporal XML DBMS is depicted in Fig. 4.

The "**Retroactive Update Checker**" checks that the Temporal XML Update submitted by the end user/application is an update operation with retroactive effect (i.e., this operations adds, deletes, or modifies past data). Notice that a past data has a valid-time interval which ends before the current time (i.e., VET < CT).

The "**Inconsistency Period Manager**", which is invoked by the "Retroactive Update Checker" in case it detects a retroactive update, determines automatically the period of inconsistency which results from a retroactive update of data, and its sub-periods with their types. Suppose that an erroneous past element "ee" is corrected by a correct past element "ce". The resulting period of inconsistency is defined by either the interval [VST_{ee} - CT] (if $VST_{ee} < VST_{ce}$), or [VST_{ce} - CT] (if $VST_{ce} < VST_{ee}$). The sub-periods of inconsistency related to this period are identified according to the study presented in the subsection 3.3.

After determining all sub-periods of inconsistency, the "**Inconsistency Period Manager**" (i) invokes the "**Transaction Catalog Manager**" in order to retrieve from the "**Transaction Catalog**" the list of transactions that were executed during each one of these sub-periods, and (ii) sends all retrieved transactions to the "Side Effect Recovery Manager".

The "**Side Effect Recovery Manager**" controls the re-execution of transactions which have used erroneous data (i.e., transactions executed during a period of inconsistency of type "Wrong Presence of Data" or "Errors in Data") and transactions that had to use newly added data (i.e., transactions executed during a period of inconsistency of type "Wrong Absence of Data"). The concerned transactions are re-executed in a "Provisory Workspace" which is a copy of the database during the corresponding period of inconsistency. At the end of the re-execution of each transaction, the "Side Effect Recovery Manager" compares the results of determined transaction already stored (as written data) in the database with the results of the execution in the provisory workspace. If there are differences between them, so an inconsistency is detected and it should be repaired. For that, the "Side Effect Recovery Manager" replaces the old result with the new one.

The "Side Effect Recovery Manager" interacts with the "**Optimizer**" module which implements a set of optimization rules. Indeed, the "Optimizer" receives a sequence of non-optimized transactions that should be re-executed, and tries to reduce them (if possible). In the following, we present three examples of these optimization rules:

- ***Rule 1***: if two successive transactions, T1 and T2,

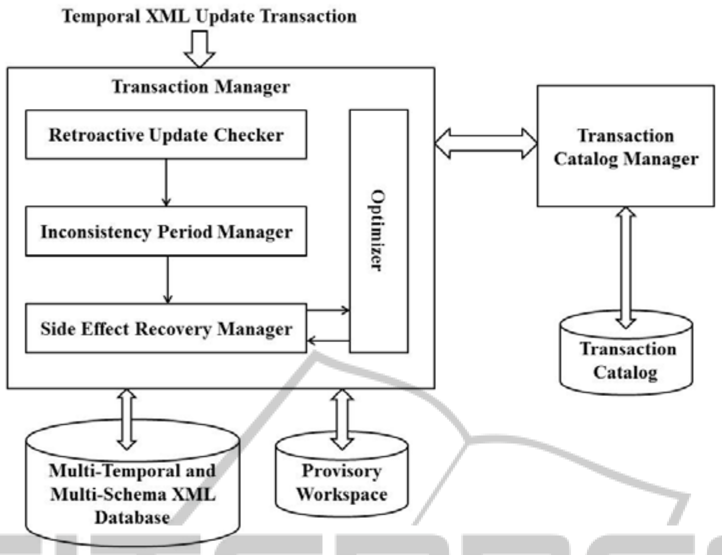
Temporal XML Update Transaction
Transaction Manager
Retroactive Update Checker
Inconsistency Period Manager
Side Effect Recovery Manager
Optimizer
Transaction Catalog Manager
Transaction Catalog
Multi-Temporal and Multi-Schema XML Database
Provisory Workspace

Figure 4: General Architecture of a temporal XML DBMS supporting automatic detecting and repairing of data inconsistencies resulting from retroactive updates.

act on the same XML element e12, such that T1 adds e12 and T2 deletes e12, then the system has to ignore these two transactions and do not re-execute them.

- ***Rule 2***: if two successive transactions, T3 and T4, act on the same XML element e34, such that T3 adds e34 and T4 updates e34, then the system has to combine/merge the two transactions into the first one (which is T3) and re-executing it (i.e., re-executing T3) but with updated values provided in the second one (which is T4): adding e34 with updated values provided in T4.
- ***Rule 3***: if a transaction does not include any operation of type data insertion, deletion, or modification, then the system has to ignore this transaction and do not re-execute it.

The "**Transaction Catalog Manager**" is added in order to have a history of transactions, which is complete (all details of transactions) and useful (i.e., easy-to-use by Side Effect Recovery Manager). For each transaction, it saves its commit time, the specified insert, delete, or update operation (since we suppose that each transaction include only one data manipulation operation), data read from the database, data written to the database, data values provided by the user in its data manipulation operation.

5 RELATED WORK

Despite the importance of preserving consistency of temporal databases after retroactive updates, this issue has been considered only to a limited extent in the current literature.

In (Samet, 1997), the author proposed the use of temporal active rules and retroactive rules (Pissinou *et al.*, 1994) in order to redress side effects resulting from a retroactive update in a temporal relational active database. An active rule or an Event-Condition-Action (E-C-A) rule is said to be temporal if (1) the event is temporal, or (2) the condition is temporal. A retroactive rule is a rule whose action includes a retroactive update.

In (Bouaziz *et al.*, 1998), the authors proposed a solution for redressing side effects generated by a retroactive update, named "correction propagation". This solution is defined to repair only inconsistencies which affect cumulative attributes (i.e., attributes which can undergo only operations of additions or subtractions of values, like the balance of a bank account or the turnover of a company).

Preserving the consistency of temporal databases was studied in other works which did not consider retroactive updates. Indeed, some of these works have dealt with database consistency with regard to (i) the respect of integrity constraints (Campo *et al.*, 2006; Svirec *et al.*, 2012), (ii) the concurrency control of transactions, by proposing new pessimistic (De Castro, 1998) and optimistic algorithms (Makni *et al.*, 2010), or (iii) the forensic analysis of database tampering (Pavlou *et al.*, 2013).

Retroactive and proactive updates were studied in temporal active databases (Etzion *et al.*, 1994) and in conventional (non-temporal) databases (Deng *et al.*, 1995). However, none of these works has dealt with inconsistencies that could result from such updates.

In (Pardede *et al.*, 2008), the authors propose a generic methodology for the management of XML data update in XML-enabled databases. However, they did not deal with retroactive updates, and define a data inconsistency as a data invalidity resulting from an XML data update.

Brahmia *et al.* (2009) and Hamrouni (2012) have studied data management in multi-temporal XML databases supporting schema versioning, but none of them have taken into account data inconsistencies resulting from retroactive update operations.

Afrati *et al.* (2009) have dealt with managing inconsistency in databases, within the framework of database repairs (Arenas *et al.*, 1999); a repair of an inconsistent database is a database over the same schema that satisfies the integrity constraints at hand and differs from the given inconsistent database in some minimal way.

Consistency of data, which takes into account the violation of semantic rules defined over a set of data items, has been also studied within the issue of data cleansing (Mezzanzanica *et al.*, 2013) and considered as a data quality dimension (Batini *et al.*, 2006).

Recently, Zellag *et al.* (2014) propose an approach for detecting consistency anomalies and automatically reducing their occurrence, in multi-tier architectures.

6 CONCLUSION

In this paper, we proposed an approach for an automatic and graceful repairing of data inconsistencies in temporal XML databases, resulting from retroactive updates. It allows the DBMS (i) to detect any database inconsistency that happens after a retroactive update operation, and (ii) to perform necessary processings, in a transparent way, in order to repair inconsistencies automatically.

We think that our approach (i) maintains effectively the consistency of the database, and (ii) provides a low-impact solution since it requires neither modifications of existing temporal database, nor extensions to existing temporal XML models (e.g., τXSchema (Snodgrasss *et al.*, 2008)) and query languages (e.g., TXPath (Rizzolo *et al.*, 2008)).

A system prototype which shows the feasibility of our approach is under development (at the University of Sfax), as a temporal stratum on top of the existing XML DBMS xDB (EMC, 2014). In fact, the first author is extending the prototype TempoXUF-Tool (Hamrouni, 2012), developed within her master's project for temporal XML data management under schema versioning, to support detecting and repairing data inconsistencies resulting from retroactive updates. Currently, this prototype allows only determining periods of inconsistencies when it receives a temporal XQuery Update Facility query with retroactive effect.

As a part of our future work, we envisage to extend our work by (i) dealing with retroactive updates which concern several temporal XML elements (in our present work we have supposed that a retroactive update consider always one temporal XML element), and (ii) studying transactions that include several temporal XML updates with retroactive effect (in fact, in the current work we supposed that a transaction include always a single temporal XML retroactive update).

Furthermore, we also plan to study how to repair inconsistencies resulting from on-time and proactive updates of temporal XML databases, since the update of a current or a future temporal XML element could also give rise to some inconsistencies.

REFERENCES

Afrati, F. N. & Kolaitis, P. G. 2009, 'Repair Checking in Inconsistent Databases: Algorithms and Complexity', *Proceedings of the 12th International Conference on Database Theory (ICDT 2009)*, St. Petersburg, Russia, 23-25 March, pp. 31-41.

Allen, J. F. 1983, 'Maintaining Knowledge About Temporal Intervals', *Communications of the ACM*, Vol. 26, No. 11, pp. 832-843.

Arenas, M., Bertossi, L. & Chomicki, J. 1999, 'Consistent Query Answers in Inconsistent Databases', *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1999)*, Philadelphia, Pennsylvania, USA, 31 May – 2 June, pp. 68-79.

Batini, C. & Scannapieco M. 2006, (eds.), *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, Heidelberg.

Bouaziz, R. & Moalla, M. 1998, 'Historisation of Data and Recovery of Side Effects' (in french), *Proceedings of 14th Journées de Bases de Données Avancées (BDA'1998)*, Hammamet, Tunisia, 23-26 October, pp. 487-507.

Bourret, R. 2005, *XML and Databases*, available at: <http://www.rpbourret.com/xml/XMLAndDatabases.htm>, last updated in September 2005.

Brahmia, Z. & Bouaziz, R. 2009, 'Data Manipulation in Multi-Temporal XML Databases Supporting Schema Versioning', *Proceedings of the 4th International EDBT Workshop on Database Technologies for Handling XML Information on the Web (DaTaX'09)*, Saint-Petersburg, Russia, 22 March.

Brahmia, Z., Grandi, F., Oliboni, B. & Bouaziz, R. (in press, to appear during 2014), 'Schema Change Operations for Full Support of Schema Versioning in the tauXSchema Framework', *International Journal of Information Technology and Web Engineering*.

Campo, M. & Vaisman, A. 2006, 'Consistency of Temporal XML Documents', *Proceedings of the 4th International XML Database Symposium (XSym 2006)*, Seoul, Korea, 10-11 September, pp. 31–45.

De Castro, C. 1998, 'On concurrency management in temporal relational databases', *Proceedings of 6th Italian Symposium on Database Systems (SEBD 1998)*, Ancona, Italy, June, pp. 189-202.

Deng, M., Sistla, A. P., & Wolfson, O. 1995, 'Temporal Conditions with Retroactive and Proactive Updates', *Proceedings of the 1st International Workshop on Active and Real-Time Database Systems (ARTDB-95)*, Skövde, Sweden, 9-11 June, pp. 122-141.

Dyreson, C. E., & Grandi, F. 2009, 'Temporal XML', in L. Liu and M. T. Özsu (Eds.), *Encyclopedia of Database Systems*. Heidelberg: Springer-Verlag, pp. 3032–3035.

EMC. 2014, *Documentum xDB*. Available at: <http://www.emc.com/products/detail/software2/documentum-xdb.htm>

Etzion, O., Gal A. & Segev A. 1994, 'Retroactive and Proactive Database Processing', *Proceedings of the 4th International Workshop on Research Issues in Data Engineering: Active Database Systems (RIDE-ADS 1994)*, Houston, Texas, USA, 14-15 February, pp. 126-131.

Etzion O., Jajodia S., Sripada S. 1998, (eds.), *Temporal Databases: Research and Practice*, LNCS 1399, Springer-Verlag.

Grandi F. (in press, to appear in July 2014), 'Temporal Databases', In M. Koshrow-Pour, (Ed.), *Encyclopedia of Information Science and Technology (3rd Ed.)*, IGI Global, Hershey.

Hamrouni, H. 2012, *Extending XQuery Update Facility to Temporal and Versioning Aspects*, Master thesis, Faculty of Economics and Management of Sfax, Tunisia.

Hamrouni, H., Brahmia, Z. & Bouaziz, R. 2014, *An Efficient Approach for Detecting and Repairing Data Inconsistencies Resulting from Retroactive Updates in Multi-Temporal and Multi-version XML Databases*, TimeCenter Technical Report TR-97, 22 pages, 17 June. <http://timecenter.cs.aau.dk/TimeCenterPublications/TR-97.pdf>

Hellerstein, J. M., Stonebraker, M. & Hamilton, J. 2007, 'Architecture of a Database System', *Foundations and Trends® in Databases*, Vol. 1, No. 2, pp. 141-259.

Jensen, C. S., Dyreson, C. E., (Eds.), et al. 1998, 'The Consensus Glossary of Temporal Database Concepts – February 1998 Version', In O. Etzion S. Jajodia, & S. Sripada, (Eds.), *Temporal Databases: Research and Practice*, LNCS 1399, pp. 367–405. Berlin: Springer-Verlag.

Makni, A. & Bouaziz, R. 2010, 'Performance evaluation of an optimistic concurrency control algorithm for temporal databases', *Proceedings of the 2nd International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2010)*, Menuires, France, 11-16 April, pp. 75–81.

Mezzanzanica, M., Boselli, R., Cesarini, M., & Mercorio, F. 2013, 'Automatic Synthesis of Data Cleansing Activities', *Proceedings of the 2nd International Conference on Data Management Technologies and Applications (DATA 2013)*, Reykjavík, Iceland, 29-31 July, pp. 138-149.

Pardede, E., Rahayu, J. W., & Taniar D., 'XML data update management in XML-enabled database', *Journal of Computer and System Sciences*, Vol. 74, No. 2, pp. 170-195.

Pavlou, K. E., & Snodgrass, R. T. 2013, 'Generalizing database forensics', *ACM Transactions on Database Systems*, Vol. 38, No. 2, paper 12.

Pissinou, N., Snodgrass, R. T., Elmasri, R., Mumick, I. S., Özsu, M. T., Pernici, B., Segev, A., Theodoulidis, B. & Dayal, U. 1994, 'Towards an Infrastructure for Temporal Databases: Report of an Invitational ARPA/NSF Workshop', *SIGMOD Record*, Vol. 23, No. 1, pp. 35-51.

Rizzolo, F. & Vaisman, A. A. 2008, 'Temporal XML: modeling, indexing, and query processing', *The VLDB Journal*, Vol. 17, No. 5, pp. 1179-1212.

Samet, A. 1997, *Automatic Recovery of Side Effects in a Multi-Version Environment*, Master thesis, Faculty of Science of Tunis, Tunisia.

Skyt, J., Jensen, C. S. & Mark, L. 2003, 'A foundation for vacuuming temporal databases', *Data and Knowledge Engineering*, Vol. 44, No. 1, pp. 1 – 29.

Snodgrass, R. T., Dyreson, C. E., Currim, F., Currim, S. & Joshi S. 2008, 'Validating Quicksand: Schema Versioning in τXSchema', *Data Knowledge and Engineering*, Vol. 65, No. 2, pp. 223-242.

Svirec, M. & Mlýnková, I. 2012, 'Efficient Detection of XML Integrity Constraints Violation', *Proceedings of the 4th International Conference on Networked Digital Technologies (NDT 2012) - Part I*, Dubai, UAE, 24-26 April, pp. 259-273.

Tatarinov, I., Ives, Z. G., Halevy, A. Y. & Weld, D. S. 2001, 'Updating XML', *Proceedings of ACM SIGMOD Conference 2001*, Santa Barbara, California, USA, , pp. 413-424.

W3C. 2011, *XQuery Update Facility 1.0*, W3C Candidate Recommendation, 17 March. <http://www.w3.org/TR/2011/REC-xquery-update-10-20110317/>

Zellag, K. & Kemme, B. 2014, 'Consistency anomalies in multi-tier architectures: automatic detection and prevention', *The VLDB Journal*, Vol. 23, No. 1, pp. 147-172.