BITEMPORAL VERSIONING IN TEMPORAL AND BITEMPORAL DATA WAREHOUSES: A REVIEW

[1]Kriti Saroha, [2]Anjana Gosain
[1]Research Scholar, [2]Professor
[1]USICT,
[1]GGSIPU, New Delhi, India

Abstract : ***Data Warehouses are built by collecting and storing historical data from various heterogeneous sources, which may be distributed or separated in nature. Data warehouses are designed as a medium that aids in multidimensional analysis and also helps in decision making process. It is very well known that the data as well as the schema of the data warehouse undergo changes with time to integrate new requirements. However, the multidimensional model of data warehouses is able to manage changes in the fact data but does not or only partly support changes in the dimension data. Thus, the temporal data warehouse model is recommended as an expansion of multi-dimensional data warehouse to manage the changes occurring in dimension data properly. The model uses either valid-time or bi-temporal time (transaction-time and valid-time grouped together) as a time-stamp for the dimension data and data-revisions for maintaining a complete data history and is therefore referred as either temporal data warehouse or bitemporal data warehouse respectively. The temporal data warehouses employ schema versioning to manage not only the evolution of schema but also the transformation of data represented under distinct versions of schema. Most of the works in the field of temporal data warehouses focus on the schema versions by applying valid-time. Valid-time versioning is essential for systems that have need to manage retro-active or pro-active schema changes but this type of versioning lacks in keeping track of these changes whereas bi-temporal versioning helps to manage as well as track retroactive and proactive schema changes. In this paper, we have performed an extensive review of the several research in the extent of versioning of schema as well as data in temporal and bitemporal data warehouses.***

IndexTerms - ***Data warehouse, temporal, bitemporal, versioning.***

I. INTRODUCTION

In recent times, the users have developed huge interest to examine and analyze data. In order to analyze data, calls for an array of technologies, especially from the area of Data Warehousing, Data Mining, and On-line Analytical Processing (OLAP) to be used. A data warehouse is a collection of historical data from several diverse and distributed data sources in support of multidimensional analysis and helps in decision making process [14, 33, 35, 39, 55, 56]. These data sources are inherently autonomous and time and again experience changes in their content and structure owing to different reasons, such as updations occurring in the sources of data, changing needs of the user, and revisions in legal and regulatory norms [20, 53]. These changes effect the structure and content of data warehouse [15, 16, 22, 42, 43]. The schema as well as data of the data warehouse can therefore evolve with time to sustain the application demands else it would be out-of-date and then users' information demands may not be fulfilled [45].

Several approaches have been suggested in the literature to manage changes in data warehouses and are defined as: (1) schema and data evolution [6–8, 31, 32, 34, 41, 54], (2) schema versioning [3–5, 9, 10, 24, 40, 42, 43, 57, 58], and (3) temporal extensions [13, 18, 21, 38, 47, 49]. Schema and data evolution approach is limited as it retains only one recent data warehouse schema and removes the previous schema version, thus incurring data loss. Multiversion data warehouses are therefore recommended as an alternate to manage the issues of evolution. As per the versioning approach, a new version for the data warehouse is built every time a change is requested, changes are applied in this new version, data is moved into the new version and all the versions are retained. Schema versioning, thus preserves the entire history of the data warehouse evolution as described through a group of schema updates/ changes. This is a proven fact that it is not just the underlying schema that undergo changes, but its associated data can also undergo revisions with time and hence, there is a need to provide support for handling several versions of data along with schema. But multidimensional models of data warehouse mostly regard facts (measures) as the dynamic member of data warehouses, and consider dimensions as static elements [44]. Therefore, these multidimensional data warehouses allow only to track changes for measures and changes in dimension data are not dealt with. But to track the updates in the dimension data along with the time when they have taken place is vital for the purpose of analysis.

Temporal extensions offer a solution to this problem [13, 18, 21, 38, 49], by using time-stamps on the components of the multidimensional data i.e., dimension data and fact data, and also the changes applied are time stamped to generate temporal versions. This gives rise to temporal extensions of multidimensional model of data warehouse to provide the needed solutions to manage time-varying data in dimensions [40]. The temporal model of data warehouses adopt the solutions proposed for temporal databases by using time-stamps on the data and data-revisions using valid, transaction, or bi-temporal time (transaction-time and valid-time combined together) [2, 11, 12, 23, 28, 29, 30, 46, 52]. Valid-time denotes the time when a data is present in reality and transaction-time denotes the time when a data is entered in the database and are mainly included so that the changes applied can be traced back in time. Also, we can use valid-time and transaction-time together [49] and is called as bi-temporal time [51]. Temporal extensions to data warehouse model are thus referred as either temporal data warehouses or bi-temporal data warehouses based upon whether they include only valid-time or both valid and transaction-time (bitemporal) as the timestamp for dimension data, i.e. data stored in dimension tables [52].

The temporal data warehouse model along with versioning of schema allows the use of data over multiple versions of schema formed by the history of updates applied to schema over time. The support of schema versioning includes efforts both at schema as well as data level. In an environment that supports schema versioning, it is necessary to maintain the complete history of schema evolution [9]. Most of the works in the field of temporal data warehouses focus on the schema versions with valid-time.

Valid-time schema versioning is vital for systems that need to manage retro-active (deferred updates) or pro-active (what-if analysis) schema changes but this form of versioning lacks in keeping track of the changes (i.e. it only executes the change but fails to trace back when the change was proposed) whereas the benefit of bi-temporal versioning of schema is that it not only helps in managing retroactive and proactive schema changes but also helps in tracing them if required, (i.e. can be used to find when a change was suggested and when it was really implemented) [12].

Retroactive changes are the revisions that are implemented in current time (Now) but made effective from some time in the past. Whereas, proactive changes are implemented in current time (Now) but made effective from some time in the future [12]. Handling and tracking of retroactive and proactive schema changes is important and may be required for some applications, e.g., identifying the spurious activity, fraud detection [40].

The rest of the paper is arranged as follows: a brief discussion on the types of schema versioning is given in section 2, section 3 presents the discussion on the prior and relevant work in the area and conclusions together with indication of future work is given in section 4.

II. Schema Versioning

A substantial amount of work is done in the field of temporal data warehouses regarding the temporal versioning of dimensional data and schema using valid-time and some of the works have also focused on bitemporal versioning of schema and data. Temporal data warehouses in general apply valid-time for time-stamping of dimension data.

Schema versioning has been classified as transaction-time, valid-time and bitemporal schema versioning. Schema versioning that is done corresponding to only one of the temporal dimensions i.e., along transaction-time or valid-time is referred as transaction-time or valid-time versioning respectively. Versioning of schema that is done using both the times i.e., using transaction-time and valid-time together is referred as bi-temporal versioning [19].

Transaction-time Schema Versioning: It uses transaction-time to timestamp all the versions of schema. Transaction-time schema versioning supports only on-time schema changes, (i.e. changes that are valid since the time they are applied) implemented only on the current version of schema. Retroactive and proactive changes are not supported by this form of versioning [12].

Valid-time Schema Versioning: It uses valid-time to timestamp every version of schema. The new version of schema is active only for its valid-time interval. If valid-time versioning of schema is applied then several schema versions remains available and any of the schema versions can be selected for applying the changes provided it either totally or partly satisfies the validity interval stated for the change. This form of versioning can handle retro and proactive revisions but does not help in the tracking of changes [12].

Bi-temporal Schema Versioning: It uses both transaction-time and valid-time for time-stamping every version of schema. In this form of schema versioning, a change can be applied only in the current and the overlapping versions of schema. For applications demanding full traceability, only bi-temporal versioning of schema can be used to determine whether a new version of schema was created in response to a retro- or a pro-active schema update [12].

III. Related Work

In the literature, the issues with regard to handling of evolutions in the multidimensional data warehouse were highlighted and subsequently, temporal model for data warehouse was suggested. In this section, a study of the existing versioning approaches and techniques for temporal and bitemporal data warehouses is presented. The authors in [1] proposed a temporal model of star schema with time-stamping of dimension members and fact instances with valid-time. The authors in [13] have suggested that the dimension structures may be stored along with their time-stamps using valid-times. The authors in [17-19] have suggested the related idea of using valid-time as time-stamp and proposed the COMET model [21]. They have also presented mapping operations to enable transformations among structure versions. The model essentially deals with only the evolvement of dimension instances and do not deal with the schema or cube evolution. The work presented in [41] recommended a temporal model for multidimensional OLAP together with a temporal query language (TOLAP). The model provides support for changes in the instances as well as dimension structure, thereby aiding schema evolution but fails to record versions of data. The authors in [9, 10] proposed a temporal extension of multidimensional model that supports evolution of multidimensional structures. The approach considers the time stamping of level instances including the hierarchy relations and also time-stamps the fact data. But, they have examined changes to schema and instances of dimensions only. The work in [37, 40] proposed a model that differentiates among the valid-time and transaction-time of data arriving from different sources and their load time at the data warehouse but does not deal with the evolution of cube. Various temporal types are acknowledged (i.e., valid-time, transaction-time, lifespan, and loading time), and provides temporal support for attributes, levels, hierarchies, and facts. Now, schema versioning also raises an important issue about storage of different versions. Authors in [25] have proposed storage solutions to manage schema versions using valid-time in temporal data warehouses. The work includes two different storage options (central and distributed data-stores) that enables to manage distinct versions of schema and their data effectively. The notion of synchronous and non-synchronous mapping of data and schema is also presented with reference to temporal data warehouses.

The authors in [50] have discussed bi-temporal versioning for relational star schemas. As compared to the earlier models, the work in [45] presented a theoretical model for bi-temporal versioning of data warehouse to prevent loss of data, examines the evolution of dimensions as well as cubes, and specifies a set of evolution functions. The authors in [1] proposed a bi-temporal storage structure to model the historical data present in the data warehouse. Each attribute in the storage structure is related with two timestamps, so that the history of its values can be tracked corresponding to both valid as well as transaction-time. The authors in [26] have proposed a conceptual model for bitemporal versioning of schema in temporal data warehouses. They have proposed evolution functions for bitemporal versioning of schema over the temporal model of data warehouse proposed in [19, 21]. The structure of metaschema is discussed for supporting and managing bitemporal versions of schema.

The authors in [12] proposed bitemporal model of data warehouse that uses both valid-time and transaction-time for timestamping dimension data but did not discuss about evolution operations for schema and dimension data or schema versioning for bitemporal data warehouses. The authors in [48] have extended the bi-temporal model of data warehouse proposed by [36] and tried to explore and extend bitemporal schema versioning in the context of bitemporal data warehouses. An evolution model is proposed for bitemporal data warehouses using bitemporal schema versioning. The work includes a set of schema and instance modification functions for managing changes at schema level and in dimension data respectively. An illustration for meta-schema to store the changes in intensional data is also presented. The work suggested a way to handle both data and schema versioning in a bi-temporal DW. The support for bi-temporal versioning of schema raises concern regarding the management and storage of multiple versions of schema together with their data. In [27], the authors have discussed the storage solutions for multiple bi-temporal versions of schema along with their data in temporal data warehouses. Two storage options (central and distributed stores of data) suggested earlier for valid-time versioning are extended to manage the data while bi-temporal schema versioning is employed.

The papers [26, 27] have explored the contribution in the area of data warehousing by proposing bitemporal versioning of schema together with storage solutions for managing several versions of schema in temporal data warehouses. Storage options are presented for managing valid-time as well as bitemporal schema versions. Also authors in [26, 48] have tried to explore and propose bitemporal schema versioning solution for temporal and bitemporal data warehouses with the aim for managing as well as tracking retro and pro-active modifications.

IV. Conclusions and Future Work

Reviewing the available data warehouse literature for schema and temporal extensions revealed that for managing and analyzing multiversion data, several issues are to be handled, namely metadata management, management of multiple versions of schema and data, data sharing, multiversion query languages, and a transactional data warehouse management. Extending this further, in the literature several proposals for temporal data warehouses so far apply valid-time alone for handling versions of schema and only few of the works so far have focused on bitemporal versioning of schema. The related work has given due consideration to the aspects related to bitemporal schema and data evolution model, schema and data versions management and metadata management.

The versioning approaches proposed in [1, 25, 27] in the temporal data warehouse literature have given due consideration to the storage aspects related to different schema versions and their data. Bitemporal versioning of dimension data and schema in temporal data warehouses is proposed in the literature [26]. Bitemporal data warehouse model has also been proposed [36] and bitemporal versioning is discussed in relation to bitemporal data warehouses [48].

The approach proposed in the literature for bitemporal versioning in temporal and bitemporal data warehouses allows to track the history of schema, guarantees accuracy of data and offers various choices for optimizing the management of storage capacity for data.

Versioning raises concerns regarding the storage of multiple schema versions, thus, the data storage options were examined to consider the options to model a temporal as well as bitemporal data warehouse that would support bi-temporal schema versioning.

Availability of storage space has a bearing on the selection of type of storage solution; for example in case of limited availability of space, central data storage solution may be adopted. Both the solutions have their pros and cons. The central data storage option does not make a copy of the data, whereas, the distributed storage solution creates duplicate copies of data among the affected store(s). Also as a consequence of the distributed storage solution, the multiple data stores may carry out independent revisions to the data by making changes using distinct versions of the schema.

Further, to go with synchronous or non-synchronous mapping between the schema and data is based on the amount of independence desired for the data components. As in case of synchronous distributed storage, it may be needed to restore the complete history of schema versions for applications demanding data from both old as well as revised schema versions. Contrarily, in the event of non-synchronous mapping, the entire history of data is retained for all the versions of schema. Thus, the old-time applications continue to run perfectly on old-version of data as earlier, however there would be need for developing new applications to work with new data that consists of new set of attributes. Also, as synchronous distributed storage updates data only in the current version of data store so it is not possible to query data from the old versions of schema. Additionally, for the queries that extend over multiple schemas, the solution would consist of only a single table in case of central storage solution. But, it is needed to build a new version of schema containing all the attributes needed to answer the query if distributed storage is used. Bitemporal schema versioning retains all the valid-time versions of schema generated in respect to continuous schema updates. Additionally, bitemporal versioning offers the advantage that it supports and manages retro- and pro-active schema updates and also maintains a record of the updates.

Further, it is observed that neither bitemporal schema versioning nor storage options for fact data have been explored in full detail in the literature. The evolution operations and storage options for bitemporal schema versioning of temporal and bitemporal data warehouses proposed in the literature can be further explored to handle the changes in fact data with time.

References

[1] Abelló, A. and Martín, C. 2003. A bitemporal storage structure for a corporate data warehouse. Proceedings International Conference on Enterprise Information Systems, 177-183, Angers, France.

[2] Agrawal, R. Gupta, A. and Sarawagi, S. 1995. Modeling Multidimensional Databases. IBM Research Report, IBM Almaden Research Center.

[3] Bębel, B., Eder, J. Konicilia, C. Morzy, T. Wrembel, R. 2004. Creation and Management of Versions in Multiversion Data Warehouse. Proc. of ACM Symposium on Applied Computing (SAC), 717-723.

[4] Bębel, B. Królikowski, Z. Wrembel, R. 2006. Managing Multiple Real and Simulation Business Scenarios by Means of a Multiversion Data Warehouse. Proc. of Int. Conference on Business Information Systems (BIS), Lecture Notes in Informatics, 102-113.

[5] Bębel, B. Wrembel, R. Czejdo, B. 2004. Storage Structures for Sharing Data in Multi-version Data Warehouse. Proc. of Baltic Conference on Databases and Information Systems, 218-231.

[6] Benítez-Guerrero, E. Collet, C. Adiba, M. 2003. The WHES Approach to Data Warehouse Evolution. Digital Journal e-Gnosis, 1665-5745.

[7] Blaschka, M. Sapia, C. Hofling, G. 1999. On Schema Evolution in Multidimensional Databases. Proc. of Int. Conference on Data Warehousing and Knowledge Discovery (DaWaK), 153-164.

[8] Blaschka, M. 1999. FIESTA: A Framework for Schema Evolution in Multidimensional Information Systems. In 6th CAiSE Doctoral Consortium, Heidelberg.

[9] Body, M. Miquel, M. Bédard, Y. Tchounikine, A. 2002. A Multidimensional and Multiversion Structure for OLAP Applications. Proc. of ACM Int. Workshop on Data Warehousing and OLAP (DOLAP), 1-6.

[10] Body, M. Miquel, M. Bédard, Y. Tchounikine, A. 2003. Handling Evolutions in Multidimensional Structures. Proc. of Int. Conference on Data Engineering (ICDE), 581.

[11] Cabibbo, L. and Torlone, R. 1998. A Logical Approach to Multidimensional Databases. Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain.

[12] DeCastro, C. Grandi, F. and Scalas, M. R. 1995. On Schema Versioning in Temporal Databases. In Recent Advances in Temporal Databases, Springer Verlag, 272–294.

[13] Chamoni, P. and Stock, S. 1997. Temporal Structures in Data Warehousing. Proc. of Int. Conference on Data Warehousing and Knowledge Discovery (DaWaK), 353- 358.
[14] Chaudhuri, S. and Dayal, U. 1997. An overview of data warehousing and OLAP technology. SIGMOD Record, 26(1): 65-74.
[15] Chen, J. Chen, S. and Rundensteiner, E. 2002. A Transactional Model for Data Warehouse Maintenance. Proc. of Int. Conference on Conceptual Modeling (ER), 247-262.
[16] Czejdo, B. Messa, K. Morzy, T. Putonti, C. 2000. Design of Data Warehouses with Dynamically Changing Data Sources. Proc. of Southern Conference on Computing.
[17] Eder, J. 2000. Evolution of Dimension Data in Temporal Data Warehouses. Technical Report 11, Univ. of Klagenfurt, Dep. of Informatics-Systems.
[18] Eder, J. and Koncilia, C. 2001. Changes of Dimension Data in Temporal Data Warehouses. Proc. of Int. Conference on Data Warehousing and Knowledge Discovery (DaWaK), 284-293.
[19] Eder, J. Koncilia, C. and Morzy, T. 2001. A Model for a Temporal Data Warehouse. In Proc. of the Int. OESSEO 2001 Conference, Rome, Italy.
[20] Eder, J. Koncilia, C. and Kogler, H. 2002. Temporal Data Warehousing: Business Cases and Solutions. Proc. of the International Conference on Enterprise Information Systems, 81-88.
[21] Eder, J. Koncilia, C. and Morzy, T. 2002. The COMET Metamodel for Temporal Data Warehouses. Proc. of Conference on Advanced Information Systems Engineering (CAiSE), 83-99.
[22] Eder, J. Koncilia, C. and Mitsche, D. 2003. Automatic Detection of Structural Changes in Data Warehouses. Proc. of Int. Conference on Data Warehousing and Knowledge Discovery (DaWaK), 119-128.
[23] Etzion, O. Jajoda, S. and Sripada, S. 1998. Temporal Databases: Research and Practice. LNCS 1399.
[24] Golfarelli, M. Lechtenbörger, J. Rizzi, S. Vossen, G. 2004. Schema Versioning in Data Warehouses. Proc. of ER Workshops, 415-428.
[25] Gosain, A. and Saroha, K. 2017. Storage Structure for Handling Schema Versions in Temporal Data Warehouses. In: Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. Advances in Intelligent Systems and Computing, vol. 518, Springer.
[26] Gosain, A. and Saroha, K. 2017. Bi-temporal Versioning of Schema in Temporal Data Warehouses. In: Data Engineering and Intelligent Computing. Advances in Intelligent Systems and Computing, vol. 542. Springer.
[27] Gosain, A. and Saroha, K. 2017. Maintaining Bi-temporal Schema Versions in Temporal Data Warehouses. In: Progress in Advanced Computing and Intelligent Engineering. Springer.
[28] Grandi, F. Mandreoli, F. and Scalas, M. 2000. A Generalized Modeling Framework for Schema Versioning Support. In Australasian Database Conference, 33–40.
[29] Grandi, F. Mandreoli, F. and Scalas, M. 1998. A Formal Model for Temporal Schema Versioning in object oriented Databases. Technical report, CSITE-014-98, CSITE - CNR.
[30] Gyssens, M. and Lakshmanan, L. V. S. 1997. A Foundation for Multi-Dimensional Databases. Proc. of Int. Conference on Very Large Data Bases (VLDB), 106-115.
[31] Hurtado, C. A. Mendelzon, A. O. and Vaisman, A. A. 1999. Maintaining Data Cubes under Dimension Updates. Proc. of Int. Conference on Data Engineering (ICDE), 346-355.
[32] Hurtado, C. A. Mendelzon, A. O. and Vaisman, A. A. 1999. Updating OLAP Dimensions. Proc. Of ACM Int. Workshop on Data Warehousing and OLAP (DOLAP), 60-66.
[33] Inmon, W. H. 1996. Building the Data Warehouse. Wiley and Sons.
[34] Kaas, Ch. K. Pedersen, T. B. and Rasmussen, B. D. 2004. Schema Evolution for Stars and Snowflakes. Proc. of Int. Conference on Enterprise Information Systems (ICEIS), 425-433.
[35] Kimball, R. and Ross, M. 2002. The Data Warehouse Toolkit. John Wiley & Sons.
[36] Koncilia, C. 2003. A Bi-Temporal Data Warehouse Model. In CAiSE'03 Short Paper Proceedings, vol. 74 of CEUR Workshop Proceedings, 77–80.
[37] Kumar, P. P. Chowdappa, K. B. and Lakshmi, S. S. 2014. A Survey on Bitemporal Data Warehousing System and Data Warehouse Design Techniques. (IJCSIT) International Journal of Computer Science and Information Technologies, 5(3): 3227 – 3231.
[38] Letz, C. Henn, E. T. and Vossen, G. 2002. Consistency in Data Warehouse Dimensions. Proc. of Int. Database Engineering and Applications Symposium (IDEAS), 224-232.
[39] Li, C. and Wang, X. 1996. A Data Model for Supporting On-Line Analytical Processing. CIKM.
[40] Malinowski, E. and Zimanyi, E. 2006. A Conceptual Solution for Representing Time in Data Warehouse Dimensions. In 3rd Asia-Pacific Conf. on Conceptual Modelling, 45–54.
[41] Mendelzon, A. O. and Vaisman, A. A. 2000. Temporal Queries in OLAP. Proc. of Int. Conference on Very Large Data Bases (VLDB), 242-253.
[42] Morzy, T. and Wrembel, R. 2003. Modeling a Multiversion Data Warehouse: A Formal Approach. Proc. of Int. Conference on Enterprise Information Systems (ICEIS), 120-127.
[43] Morzy, T. and Wrembel, R. 2004. On Querying Versions of Multiversion Data Warehouse. Proc. ACM Int. Workshop on Data Warehousing and OLAP (DOLAP), 92-101.
[44] Pedersen, T. B. Jensen, C. S. and Dyreson, C. E. 2001. A foundation for capturing and querying complex multidimensional data. Information Systems, 26 (5): 383-423.
[45] Rechy-Ramirez, E. J. and Benitez-Guerrero, E. 2006. A Model and Language for Bi-temporal Schema Versioning in Data Warehouses. Proceedings International Conference on Computing.
[46] Roddick, J. 1996. A Survey of Schema Versioning Issues for Database Systems. In Information and Software Technology, 37(7): 383-393.
[47] Sarda, N. L. 1999. Temporal Issues in Data Warehouse Systems. Proceedings International Symposium on Database Applications in Non-Traditional Environments, 27-34.
[48] Saroha, K. and Gosain, A. 2016. Bi-temporal schema versioning in bi-temporal data warehouse. CSI Transactions on ICT, vol3, Issue2-4, 135-142.

[49] Schlesinger, L. Bauer, A. Lehner, W. Ediberidze, G. Gutzman, M. 2001. Efficienlty Synchronizing Multidimensional Schema Data. Proc. of ACM Int. Workshop on Data Warehousing and OLAP (DOLAP), 69-76.
[50] Serna-Encinas, M.T. and Adiba, M. 2005. Exploiting Bitemporal Schema Versions for Managing an Historical Medical Data Warehouse: A Case Study. In Proc. of the 6th Mexican Int. Conf. on Computer Science (ENC’05), 88–95. IEEE Computer Society.
[51] Soo, M. 1991. Bibliography on Temporal Databases. ACM SIGMOD Record.
[52] Tansel, A. Gadia, J. Jajodia, S. Segev, A. Snodgrass, R. 1993. Temporal Databases: Theory, Design and Implementation, Benjamin Cummings.
[53] Turki, I. Z. Jedidi, F. G. and Bouaziz, R. 2010. Multiversion Data Warehouse Constraints. Proceedings of the ACM 13th international workshop on Data warehousing and OLAP.
[54] Vaisman, A. and Mendelzon, A. 2001. A Temporal Query Language for OLAP: Implementation and Case Study. Proc. of Workshop on Data Bases and Programming Languages (DBPL), 78-96.
[55] Vassiliadis, P. and Sellis, T. 1999. A Survey of Logical Models for OLAP Databases. SIGMOD Record 28 (1).
[56] Vetterli, T. Vaduva, A. and Staudt, M. 2000. Metadata Standards for Data Warehousing: Open Information Model vs. Common Warehouse Metadata. SIGMOD Record, 29(3): 68-75.
[57] Wrembel, R. and Bębel, B. 2005. Metadata Management in a Multiversion Data Warehouse. Proc. of Ontologies, Databases, and Applications of Semantics (ODBASE), 1347-1364.
[58] Wrembel, R. and Morzy, T. 2006. Managing and Querying Versions of Multiversion Data Warehouse. Proc. of Int. Conference on Extending Database Technology (EDBT), 1121-1124.