Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.4, 2012

Challenging Issues of Spatio-Temporal Data Mining

A.N.M. Bazlur Rashid[1*] Md. Anwar Hossain[2]

1. Department of Computer Science and Engineering, Comilla University, Comilla, Bangladesh
2. Department of Information and Communication Engineering, Pabna Science and Technology University, Pabna, Bangladesh

* E-mail of the corresponding author: bazlur.cse@cou.ac.bd

Abstract

The *spatio-temporal database* (STDB) has received considerable attention during the past few years, due to the emergence of numerous applications (*e.g.*, flight control systems, weather forecast, mobile computing, *etc.*) that demand efficient management of moving objects. These applications record objects' geographical locations (sometimes also shapes) at various timestamps and support queries that explore their historical and future (predictive) behaviors. The STDB significantly extends the traditional *spatial database*, which deals with only stationary data and hence is inapplicable to moving objects, whose dynamic behavior requires re-investigation of numerous topics including data modeling, indexes, and the related query algorithms. In many application areas, huge amounts of data are generated, explicitly or implicitly containing spatial or spatiotemporal information. However, the ability to analyze these data remains inadequate, and the need for adapted data mining tools becomes a major challenge. In this paper, we have presented the challenging issues of spatio-temporal data mining.

Keywords: database, data mining, spatial, temporal, spatio-temporal

1. Introduction

Classical data mining techniques often perform poorly when applied to spatial and spatio-temporal data sets because of the many reasons. First, these dataset are embedded in continuous space, whereas classical datasets (e.g. transactions) are often discrete. Second, patterns are often local where as classical data mining techniques often focus on global patterns. Finally, one of the common assumptions in classical statistical analysis is that data samples are independently generated. When it comes to the analysis of spatial and spatio-temporal data, however, the assumption about the independence of samples is generally false because such data tends to be highly self correlated. For example, people with similar characteristics, occupation and background tend to cluster together in the same neighborhoods. In spatial statistics this tendency is called autocorrelation. Ignoring autocorrelation when analyzing data with spatial and spatio-temporal characteristics may produce hypotheses or models that are inaccurate or inconsistent with the data set.

In application areas such as robotics, computer vision, mobile computing, and traffic analysis, huge amounts of data are generated and stored in databases, explicitly or implicitly containing spatial or spatiotemporal information. For instance, the proliferation of location-aware devices gives rise to vast amounts of frequently updated telecommunication and traffic data, and satellites generate terabytes of image data daily. These huge collections of spatiotemporal data often hide possibly interesting information and valuable knowledge. It is obvious that a manual analysis of these data is impossible, and data mining might provide useful tools and technology in this setting. Spatiotemporal data mining is an emerging research area that is dedicated to the development of novel algorithms and computational techniques for the successful analysis of large spatiotemporal databases and the disclosure of interesting knowledge in

spatiotemporal data. However, the ability to analyze these data remains inadequate and the need for adapted data mining tools becomes a major challenge.

Spatio-Temporal Databases (STDB) explores recent trends in flexible querying and reasoning about time- and space-related information in databases. It shows how flexible querying enhances standard querying expressiveness in many different ways, with the aim of facilitating extraction of relevant data and information. Flexible spatial and temporal reasoning denotes qualitative reasoning about dynamic changes in the spatial domain, characterized by imprecision or uncertainty (or both).

Spatio-temporal databases need to support a wide variety of continuous spatio-temporal queries. For example, a continuous spatio-temporal range query may have various forms depending on the mutability of objects and queries. In addition, a range query may ask about the past, present, or the future. A naive way to process continuous spatio-temporal queries is to abstract the continuous queries into a series of snapshot queries. Snapshot queries are issued to the server (*e.g.*, a location-aware server) every T seconds. The naive approach incurs redundant processing where there may be only a slight change in the query answer between any two consecutive evaluations [1, 2, 3].

The reminder of the paper is organized as follows: Section 2 presents an architecture of data mining. Section 3 and 4 briefly illustrate the spatial and temporal data mining respectively. Section 5 describes the spatio-temporal data mining. A conclusion is given in Section 6.

2. An Architecture of Data Mining

Data mining, *the extraction of hidden predictive information from large databases*, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

To best apply these advanced techniques, they must be fully integrated with a data warehouse as well as flexible interactive business analysis tools. Many data mining tools currently operate outside of the warehouse, requiring extra steps for extracting, importing, and analyzing the data. Furthermore, when new insights require operational implementation, integration with the warehouse simplifies the application of results from data mining. The resulting analytic data warehouse can be applied to improve business processes throughout the organization, in areas such as promotional campaign management, fraud detection, new product rollout, and so on. Figure 1 illustrates an architecture for advanced analysis in a large data warehouse.

The ideal starting point is a data warehouse containing a combination of internal data tracking all customer contact coupled with external market data about competitor activity. Background information on potential customers also provides an excellent basis for prospecting. This warehouse can be implemented in a variety of relational database systems: Sybase, Oracle, Redbrick, and so on and should be optimized for flexible and fast data access.

An OLAP (On-Line Analytical Processing) server enables a more sophisticated end-user business model to be applied when navigating the data warehouse. The multidimensional structures allow the user to analyze the data as they want to view their business – summarizing by product line, region, and other key perspectives of their business. The Data Mining Server must be integrated with the data warehouse and the OLAP server to embed ROI-focused business analysis directly into this infrastructure. An advanced, process-centric metadata template defines the data mining objectives for specific business issues like campaign management, prospecting, and promotion optimization. Integration with the data warehouse enables operational decisions to be directly implemented and tracked. As the warehouse grows with new

decisions and results, the organization can continually mine the best practices and apply them to future decisions.

This design represents a fundamental shift from conventional decision support systems. Rather than simply delivering data to the end user through query and reporting software, the Advanced Analysis Server applies users' business models directly to the warehouse and returns a proactive analysis of the most relevant information. These results enhance the metadata in the OLAP Server by providing a dynamic metadata layer that represents a distilled view of the data. Reporting, visualization, and other analysis tools can then be applied to plan future actions and confirm the impact of those plans.

3. Spatial Database

Spatial data mining is the process of discovering interesting and previously unknown, but potentially useful patterns from large spatial datasets. Extracting interesting and useful patterns from spatial datasets is more difficult than extracting the corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation.

A spatial database contains objects which are characterized by a spatial location and/or extension as well as by several non-spatial attributes. Figure 2 illustrates a spatial database on Bavaria as an example. Depicted is the relation Communities containing polygons which represent communities in a geographic information system. This spatial database on Bavaria - referred to as the BAVARIA database. The database contains the ATKIS 500 data and the Bavarian part of the statistical data obtained by the German census of 1987, *i.e.* 2043 Bavarian communities with one spatial attribute (polygon) and 52 non-spatial attributes (such as average rent or rate of unemployment). Also included (in a separate table of the database) are spatial objects representing natural objects like mountains or rivers and infrastructure such as highways or railroads. The total number of spatial objects in the database then amounts to 6924.

The discovery process for spatial data is more complex than for relational data. This applies to both the efficiency of algorithms as well to the complexity of possible patterns that can be found in a spatial database. The reason is that, in contrast to mining in relational databases, spatial data mining algorithms have to consider the neighbours of objects in order to extract useful knowledge. This is necessary because the attributes of the neighbours of some object of interest may have a significant influence on the object itself.

4. Temporal Database

Temporal data stored in a temporal database is different from the data stored in non-temporal database in that a time period attached to the data expresses when it was valid or stored in the database. Conventional databases consider the data stored in it to be valid at time instant now, they do not keep track of past or future database states. By attaching a time period to the data, it becomes possible to store different database states.

A first step towards a temporal database thus is to timestamp the data. This allows the distinction of different database states. One approach is that a temporal database may timestamp entities with time periods. Another approach is the timestamping of the property values of the entities. In the relational data model, tuples are timestamped, where as in object-oriented data models, objects and/or attribute values may be timestamped. What time period do we store in these timestamps? There are mainly two different notions of time which are relevant for temporal databases. One is called the valid time, the other one is the transaction time. Valid time denotes the time period during which a fact is true with respect to the real world. Transaction time is the time period during which a fact is stored in the database. Note that these two time periods do not have to be the same for a single fact. Imagine that we come up with a temporal database storing data about the 18th century. The valid time of these facts is somewhere between 1700 and 1799, where as the transaction time starts when we insert the facts into the database, for example, January 21,

1998.

Assume we would like to store data about our employees with respect to the real world. Then, the result appears as like in Table 1.

The above valid-time table stores the history of the employees with respect to the real world. The attributes *ValidTimeStart* and *ValidTimeEnd* actually represent a time interval which is closed at its lower and open at its upper bound. Thus, we see that during the time period [1985 - 1990), employee John was working in the research department, having a salary of 11000. Then he changed to the sales department, still earning 11000. In 1993, he got a salary raise to 12000. The upper bound INF denotes that the tuple is valid until further notice. Note that it is now possible to store information about past states. We see that Paul was employed from 1988 until 1995. In the corresponding non-temporal table, this information was (physically) deleted when Paul left the company.

4.1 Different Forms of Temporal Databases

The two different notions of time - valid time and transaction time - allow the distinction of different forms of temporal databases. A historical database stores data with respect to valid time, a rollback database stores data with respect to transaction time. A bitemporal database stores data with respect to both valid time and transaction time.

Commercial DBMS are said to store only a single state of the real world, usually the most recent state. Such databases usually are called snapshot databases. A snapshot database in the context of valid time and transaction time is depicted in Figure 3.

On the other hand, a bitemporal DBMS such as TimeDB stores the history of data with respect to both valid time and transaction time. Note that the history of when data was stored in the database (transaction time) is limited to past and present database states, since it is managed by the system directly which does not know anything about future states.

A table in the bitemporal relational DBMS TimeDB may either be a snapshot table (storing only current data), a valid-time table (storing when the data is valid wrt. the real world), a transaction-time table (storing when the data was recorded in the database) or a bitemporal table (storing both valid time and transaction time). An extended version of SQL allows specifying which kind of table is needed when the table is created. Existing tables may also be altered (schema versioning). Additionally, it supports temporal queries, temporal modification statements and temporal constraints.

The states stored in a bitemporal database are sketched in Figure 4. Of course, a temporal DBMS such as TimeDB does not store each database state separately as depicted in Fig. 4. It stores valid time and/or transaction time for each tuple, as described above.

5. Spatio-Temporal Database

Depending on the temporal aspects of data, a STDB aims at either *historical* or *predictive* retrieval. Specifically, given a set of objects o_1, o_2, . . . , o_N (where N is termed the cardinality), a historical STDB stores, for each object o_i ($1 \leq i \leq N$), its extent $o_i.E(t)$ at all the timestamps t in the history. Following the convention of spatial databases, each extent $o_i.E(t)$ can be a polygon describing the object's actual shape at time t (e.g., the contour of a moving typhoon). Specially, if the shape is not important (*e.g.*, cars, flights, etc.), $o_i.E(t)$ degenerates to a point describing the location of o_i at time t. In practice, the extents of the same object at the successive timestamps can be compressed using various methods (*e.g.*, if the object remains stationary at several continuous timestamps, its extent is stored only once during this period). A predictive STDB, on the other hand, stores, for each (usually point) object o_i, its most recent updated location $o_i.L(t_{upd})$ (where t_{upd} is the time of the object's last update), and the motion function describing its current movement. The most popular motion function is the linear function [4, 5], because it (*i*) can approximate any trajectory, and (*ii*) requires the fewest number of parameters. Specifically, in addition to $o_i.L(t_{upd})$, the system only needs to record the object's velocity $o_i.vel$, such that the object's location at any future time $t >$

t_{upd} can be calculated as $o_i.L(t) = o_i.L(t_{upd}) + o_i.vel.(t - t_{upd})$. Using such modeling, an object needs to issue an update to the database only if the parameters of its motion function (*e.g.*, $o_i.vel$ for linear movement) change.

Since the spatial database can be regarded as a special type of STDB where all the objects have zero velocities, all the spatial query types naturally find their counterparts in STDB, except that they are augmented with additional temporal predicates. Specifically, a *window query* (WQ) specifies a query region q_R and time interval q_T (consisting of continuous timestamps), and finds all the objects whose extents (or locations for point data) intersect q_R during q_T. Particularly, the *selectivity* of a WQ equals the number of retrieved objects divided by the dataset cardinality, and its accurate estimation [6, 7] is imperative to query optimization. A *k nearest neighbor* (*k*NN) query specifies a query point q_P and time interval q_T , and finds the *k* objects whose distances to q_P during q_T are the smallest. These problems become even more complex if query regions/points (in WQ/*k*NN) are also moving. While the above queries involve only one dataset, the *within-distance* join (WDJ), given two datasets S_1, S_2 reports all the object pairs (o_1, o_2) in the cartesian product $S_1 \times S_2$, such that the distance between o_1, o_2 during a query time interval q_T is smaller than certain threshold *d*. The selectivity of a join is the number of retrieved pairs divided by the size of $S_1 \times S_2$. Similarly, the *k closest pair* (*k*CP) query retrieves the k object pairs (o_1, o_2) such that the distance of o_1,o_2 during q_T is the smallest, among all the pairs in $S_1 \times S_2$. Note that the above queries can be defined in both historical and predictive STDB.

In addition to queries inherited from conventional spatial databases, the dynamic nature of STDB also leads to several novel query types. For historical databases, the *navigational* WQ has been introduced which specifies two query regions q_{R1}, q_{R2} and timestamps q_{T1}, q_{T2} and retrieves all the objects that intersect q_{R1} at q_{T1}, and also intersect q_{R2} at q_{T2} (*e.g.*, find all the vehicles that appeared in Harvard at 5pm yesterday and then appeared in MIT 10 minutes later). In predictive STDB, [8] points out that the results of traditional queries (*i.e.*, WQ, *k*NN, WDJ, *k*CP) are usually inadequate because they may change (sometimes almost immediately) due to the movements of objects and/or queries (*e.g.*, a user's nearest gas station may change as s/he drives on the highway).

Motivated by this, [8] proposes the *time-parameterized* (TP) query, which applies to any traditional query, and returns, in additional to the result *R*, also (*i*) an expiry time *T* of *R*, and (*ii*) the change *C* of the result after *T*. An example of TPNN is to report (*i*) the nearest station *s*, (*ii*) when *s* will cease to be the nearest (given the user's moving direction and speed), and (*iii*) the new nearest station after the expiry of *s*. The concept of TP is extended to the *continuous query* in [9], which is another general concept applicable to all traditional queries and aims at continuously tracking the result changes until certain conditions are satisfied. A continuous WQ, for instance, may "return the aircrafts within 10 miles from flight UA183 now and continuously update this information until its arrival". In TP and continuous processing, the moving direction of the query can be clearly specified, which is not true in some applications (*e.g.*, a tourist wandering around casually). The concept useful in such scenarios is the *location-based* (LB) query [10], which applies to WQ and *k*NN and finds the query result as well as its validity region such that, as long as the query is in this region, its result will remain the same. For example, a LB NN may return the nearest restaurant of a tourist, as well as a validity region in which the restaurant will remain the nearest.

Numerous access methods have been proposed for efficient spatio-temporal query processing. A straightforward approach to index historical STDB is to create a spatial index (the most common ones include the R-tree) at each timestamp in history, managing objects' extents at that timestamp. This is the idea behind the so-called *partially persistent structures* which in order to reduce the space consumption allows the R-trees at consecutive timestamps to share common nodes if the objects in these nodes do not incur extent changes. The first partially persistent structure, the *historical R-tree* (HR-tree), however, still involves considerable data redundancy, which led to the development of the *multi-version R-tree* (MVR-tree) and its subsequent versions. Besides the partially persistent methodology, historical STDB can also be indexed using a 3D R-tree by treating time just as an extra dimension (in addition to the two spatial dimensions). Specifically, each record in the 3D R-tree represents a 3D box, whose spatial projection

corresponds to the extent of a stationary object, and whose temporal projection denotes the time interval during which the object is stationary. Similar ideas are used in the *trajectory bundle tree* (TB-tree), a structure optimized for navigational WQ queries. In practical STDB, it adapts the Quadtree (a spatial index) for indexing the movements of 1D object, while the *time-parameterized R-tree* (TPR-tree) and its improved versions support objects of arbitrary dimensionality. Finally, indexing moving objects has also been studied in theory [11] which develops numerous interesting structures with provable worst-case performance bounds. These bounds, however, usually involve large hidden constants, rendering these "theoretical" solutions to be outperformed by the "practical" solutions introduced earlier.

Due to the time component, spatiotemporal databases need to manage large amounts of data accumulated over long period of time. A user asks queries over this data and the straightforward solution to find the answer is to read all objects in the database and return the objects that belong to the answer. However this approach is inefficient due to the size of the database. A better solution is to construct indexes over the data and answer a query by reading only a small part of the database. In general, an index is a way to organize a dataset in disk pages in order to answer efficiently a specific type of queries, by reading only a small number of disk pages [12-14].

6. Conclusion

Spatio-temporal data mining is becoming now very important field of research as it focuses the data for not only static view point but also on time and space. Thus it is useful to locate future statistics based on time and space but querying, indexing and many other relevant issues of spatio-temporal data are not easy till days. Spatiotemporal data support is considered to be an important research direction, since many applications need to manipulate data that change over time. STDBMS, in particular, should (*i*) offer appropriate data types and query languages for time-evolving spatial objects, (*ii*) provide efficient indexing techniques and access methods for spatiotemporal query processing and (*iii*) exploit cost models for query optimization purposes. So the research on spatio-temporal data mining is still going on.

References

Geetha, R., Sumathi, N. & Sathiabama, D. S. (2008), "A Survey of Spatial, Temporal and Spatio-Temporal Data Mining", *Journal of Computer Applications* **1(**4), 31-33.

Roddick, J. F., Egenhofer, M., J., Hoel, E., Papadias, D. & Salberg, B. (2004), "Spatial, Temporal and Spatio-Temporal Databases – Hot Issues and Direction for PhD Research", *ACM SIGMOD Record*, **33**(2), 126-131.

Hadzilacos, T., Manolopoulos, Y., Roddick, J. F. & Theodoridis, Y. (2003), "Advances in Spatial and Temporal Databases", *Lecture Notes in Computer Science* **2750**, 306-324.

Forlizzi, L., Guting, R., Nardelli, R. & Schneider, M. (2000), "A Data Model and Data Structures for Moving Objects Databases", *ACM SIGMOD Record* **29**(2), 319-330.

Gutting, R., Bohlen, M., Erwig, M., Jensen, C., Lorentzos, N., Schneider, M. & Vazirgiannis, M. (2000), "A Foundation for Representing and Querying Moving Objects", *ACM Transactions on Database Systems (TODS)* **25**(1), 1–42.

Choi, Y. & Chung, C. (2002), "Selectivity Estimation for Spatio-Temporal Queries to Moving Objects", *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, 440-451.

Hadjieleftheriou, M., Kollios, G. & Tsotras, V. (2003), "Performance Evaluation of Spatio-Temporal Selectivity Techniques", *15th International Conference on Science and Statistical Database Management*, 202-211.

Tao, Y., Sun, J. & Papadias, D. (2003), "Selectivity Estimation for Predictive Spatio-Temporal Queries", *19th International Conference on Data Engineering (ICDE'03)*, 417-428.

Tao, Y., Mamoulis, N. & Papadias, D. (2003), "Validity Information Retrieval for Spatio-Temporal Queries", *Lecture Notes in Computer Science* **2750**, 159-178.

Papadias, D., Kalnis, P., Zhang, J. & Tao, Y. (2001), "Efficient OLAP Operations in Spatial Data Warehouses", *Proceedings of Symposium on Spatial and Temporal Databases (SSTD)*, 443-459.

Tao, Y. & Papadias, D. (2002), "Time-Parameterized Queries in Spatio-Temporal Databases", *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, 334-345.

Hadjieleftheriou, M., Kollios, G., Tsotras, V. & Gunopulos, D. (2002), "Efficient Indexing of Spatiotemporal Objects", *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology (EDBT)*, 251-268.

Kollios, G., Gunopulos, D., Tsotras, V., Delis, A. & Hadjieleftheriou, M. (2001), "Indexing Animated Objects using Spatiotemporal Access Methods", *IEEE Transactions on Knowledge and Data Engineering* **13**(5), 758-777.

Saltenis, S., Jensen, C., Leutenegger, S. & Lopez, M. (2000), "Indexing the Positions of Continuously Moving Objects", *ACM SIGMOD Record* **29**(2), 331-342.

A.N.M. Bazlur Rashid born in 1984 at Rangpur, Bangladesh. He received his B.Sc. Engg. degree in Computer Science and Engineering from Rajshahi University of Engineering and Technology, Rajshahi, Bangladesh in 2005 and M.Sc. Engg. degree in Information and Communication Technology from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2010. He has several years of professional experience in the field of database having Oracle professional certification in database administration. Currently, he has been serving as a Lecturer in the Department of Computer Science and Engineering of Comilla University, Comilla, Bangladesh. His major field of study includes query processing, query optimization, materialized view, database and information system, data warehousing and data mining. E-mail: bazlur.rashid@yahoo.com

Md. Anwar Hossain was born in 26th November 1983 in the district of Pabna, Bangladesh. He has got his B.Sc (Honours) and M.Sc degree in Information and Communication Engineering from the University of Rajshahi, Rajshahi, Bangladesh in 2007 and 2009 respectively. He has several years of experience in the field of programming and IT sector. Currently, he has been working as a Lecturer in the department of Information and Communication Engineering at Pabna Science and Technology University, Pabna, Bangladesh. His major field of study is radio resource management for wireless networks, network coding and game theory in wireless communication, cognitive radio networks, wireless sensor network, mobile ad hoc network. E-mail: manwar_ice@yahoo.com

Appendix

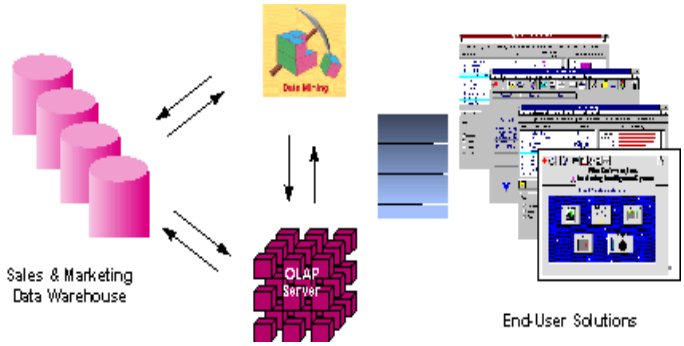

Figure 1. Integrated Data Mining Architecture

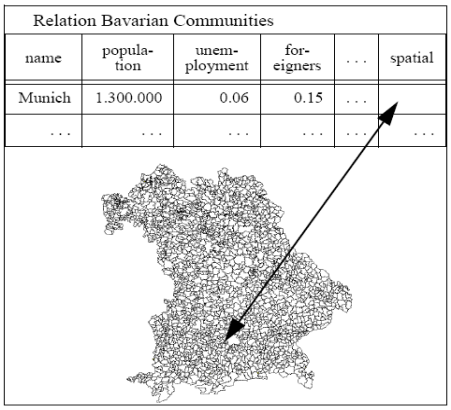

Figure 2. Spatial and Non-Spatial Attributes of Communities

Table 1. Employees record with valid start and end time

EmpID	Name	Department	Salary	ValidTimeStart	ValidTimeEnd
10	John	Research	11000	1985	1990
10	John	Sales	11000	1990	1993
10	John	Sales	12000	1993	INF
11	Paul	Research	10000	1988	1995
12	George	Research	10500	1991	INF
13	Ringo	Sales	15500	1988	INF

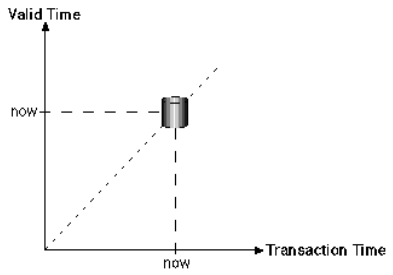

Figure 3. Snapshot Database in the Context of Valid Time and Transaction Time

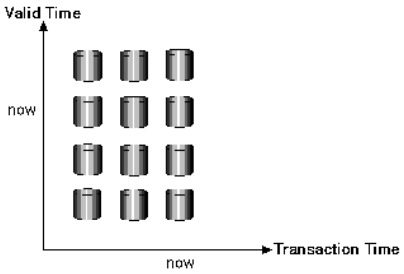

Figure 4. The States Stored in a Bitemporal Database

This academic article was published by The International Institute for Science, Technology and Education (IISTE). The IISTE is a pioneer in the Open Access Publishing service based in the U.S. and Europe. The aim of the institute is Accelerating Global Knowledge Sharing.

More information about the publisher can be found in the IISTE's homepage:
http://www.iiste.org

The IISTE is currently hosting more than 30 peer-reviewed academic journals and collaborating with academic institutions around the world. **Prospective authors of IISTE journals can find the submission instruction on the following page:** http://www.iiste.org/Journals/

The IISTE editorial team promises to the review and publish all the qualified submissions in a fast manner. All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Printed version of the journals is also available upon request of readers and authors.

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar