Direct RESEARCH JOURNALS
Vol.4 (2), pp. 8-14, October 2017
ISSN 2354-4155
DOI: https://doi.org/10.26765/DRJEIT.2017.0161
Article Number: DRJA14030161

Direct Research Journal of Engineering and Information Technology
http://directresearchpublisher.org/journal/drjeit

Research Paper

Survey of Time-varying Data Models

***[1]Terungwa Simon Yange, [2]Oluoha, O., [3]Hettie Abimbola Soriyan and [4]Adenike Olaogun**

[1]Department of Mathematics/Statistics/Computer Science, Federal University of Agriculture, Makurdi, Nigeria.
[2]Department of Computer Science, University of Nigeria, Nsukka.
[3]Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria.
[4]Department of Nursing Science, Obafemi Awolowo University, Ile-Ife, Nigeria.
*Corresponding author E-mail: lordesty2k7@ymail.com .

Received 20 February 2017; Accepted 8 October, 2017

There are two types of data models: time-varying (temporal) data model and non-temporal data model. Data in the real-world exist in three (3) states: past, present and future; the past state is refers to as history, current/present is the valid state, and the future is that which supposed to valid after the current state. Data assumes these states as time progresses. Non-temporal data model only represent present state of data. With the time-varying model, it is possible to represent these three states. In this paper, we carry out the review of the time-vary data model and proposed a temporal data model for nursing process.

Key words: Data, temporal data, model, time-varying, nursing process.

INTRODUCTION

A model is the abstract representation of an object, behaviour, or system that one wants to understand. A model creates the view of a concept that only exists in the mind. This view could be conceptual or mathematical. A data model creates a conceptual structure or defines the constructs and formalisms available to define, modify and access the data. The data model provides users with data structures which hide many of the details of how the data is stored physically. There are two classes of data models: non-temporal and temporal data models. The non-temporal (time-varying) data models do not take into consideration the changes that occur to data with respect to time in the real-world. They consider only the present state of data. But in the real-world, data exist in three (3) states (*Steiner, 1998*): past, present and future; the past state is refers to as history, current/present is the valid state, and the future is that which supposed to valid after the current state. Data assumes these states as time progresses. Most of the existing data models are non-temporal. Examples of non-temporal data models include hierarchical data model, network data model, relational data model, object-oriented data model, object-relational data model, associative data model, context data model, etc. All these data models consider only the valid state (current) of data. The temporal data model only come into being through a process known as database extension. This involves the incorporation of features which were not present in the database to it, in this case time. This enables the database to understand the time aspect/dimension of data so as to effectively manage its different states. The term temporal or time-varying according to (*Bolour et al., 1982*) is applied to the data item which is "stamped" with the time during which it is valid or in effect. The primary purpose for temporal data is to provide information on past, current and future states of data.

This is a major change in concept for databases. With non-temporal databases, only one value is associated

with an attribute. But in temporal databases more than one value can be assigned to an attribute. For example, the marital status of a client can be both married and single at different times or views of the database. In medical and financial applications, it is important to maintain data in the different states so as to make proper decisions about events. The impetus for the temporal database is perpetuated by application's requirement for time-varying data (*Hom, 1988*).

Temporal data management can be very difficult using conventional/classical (non-temporal) data models (*Matiasko et al., 2008*). Software systems for the management of data in a wide variety of domains (e.g., medicine, economics, accounting, etc.) must include information about evolution of entities of interest so as to efficiently manage information about time as well as the time variations of the entities of the domains they modelled (*Mahmood et al., 2012*). The conventional database system is the currently most often used type of databases (*Kvet and Matiasko (2012)*. Temporal databases can be applied in areas where data generated varies with time so as to maintain the states transition. The nursing process is central to all nursing care; it encompasses all steps taken by the nurse in caring for a client. Its primary aim is to know the health status and the problems of clients which may be actual or potential. It is made up of a series of stages that are used to achieve the objective - the health improvement of the client. The use of nursing process can stop at any stage as deemed necessary provided the client need no care or can be repeated as needed. As this process is repeated, new data is collected at every point in time; hence, its temporal nature. This paper takes a survey of the different temporal data models and based on this, we designed a temporal data model for nursing process. This would provide more information to assist the nurses to manage the client care process and other related issues. It will also provide a reference point for nursing research. By this, more historical data will be made available, making it possible for more informed choices regarding present and future decision making.

RELATED WORK

According to *Steiner (1998)*, there are two types of data models: non-temporal and temporal data models. These are the basic types of data models. They have other models within them, for instance, the non-temporal model consist of the hierarchical, the network, the relational, object, the object-relational data models, etc. Our focus in this paper is on the temporal data model.

Temporal data model

Humans have been interested in the concept of time from the very beginning of life. Philosophers, mathematicians, logicians, physicists and others have discussed the concept of time without a definite definition. According to (*Bolour et al., 1982*), time is defined as the period during which something exists or happens. It is also the valuable information concerning the life span of something in the real-world. Time is universal. A classic representation of an object or thing is a three-dimensional description of its height, width and length. Little or no attention is paid to the fourth dimension (i.e., time). An object from a person's view is a continuity of changes over time (*Bolour et al. 1982*). Present day use of database systems does not represent time aspects of data. They only reflect the most current view, a snapshot of the real-world at a specific moment in time (*Rajak and Saxena, 2009*). *Bolour et al. (1982)* listed several drawbacks to this scheme. First, the data in the database is only valid for the most recent updates. Any new updates will not be reflected in the database until it is actually entered. Secondly, previous information stored in the database is forever lost in an update or deletion. Database systems have been developed to model the reality, which should include historical data. The three well known data models, namely, object-relational, object-oriented, relational, and hierarchical and network, do not address the issue of time (*Rajak and Saxena, 2009*). Ad hoc methods, such as backups, checkpoints and transaction logs have been used for the retrieval of historical data. Additionally, attributes involving time have been handled by the application program, where dates, for example, are values of some specific type, usually integers. Three types of time aspects were identified by (*Ahn, 1993*): user-defined time, valid time and transaction time.

User-defined time

Temporal data models support programmer-defined data type for time by offering, for instance, a type date or timestamp for attribute values (*Ahn, 1993*). The values can be any time instant referring to past, present or future time points. Programmer-defined time values are supplied by the user and may be updated. The distinction of interpreted or un-interpreted time attributes arises because temporal data models use the data structures of the underlying non-temporal data model to store temporal data by extending the schemas with special time attributes. In case a query is evaluated using temporal semantics, the algebra operations access these special time attributes and thus interpret them.

Valid time

Another aspect of a temporal data model is to be able to denote when facts are true with respect to the real-world. For example, we would like to know when a care plan for

a particular client is designed and at what time it is supposed to be delivered to the client, the time at which updates are made to the care plans, etc. Sometimes, we also would like to record future events, for example, in a care plan, most of the actions are scheduled to take place at a later date. This notion of time, recording data with respect to when it was, is or will be valid in the real-world, is called valid time. A valid-time interval thus records the time period when a fact is true. It is interpreted by a DBMS supporting valid time, for example, during query evaluation or constraint checking. Valid time must be supplied by the user when adding or modifying data. Valid-time values can be updated (*Ariav, 1986*).

Transaction time

Often times, data cannot be recorded in a database in real time, for example, due to a delay in the processing of information. There is always a time gap between data being valid in the real world and recording the data in a database (*Ahn, 1993*). Sometimes, it is also necessary to keep track during which time periods facts are stored in a database. This notion of time is called transaction time. We have seen that valid time is used, for example, to record the history of a client. This can be viewed as the history of value changes. Assume that a medical record officer (MRO) records a wrong date of client visit to a facility or wrong personal information. In this case, a correction of the wrong values has to be made. This must be recorded, for example, since the hospital also wants to know when values of the client history were corrected (and not only when they were changed). When the MRO updates the values, the system automatically sets the transaction-time interval of the old values to end at the current time, and the transaction-time interval of the corrected values to start at the current time. Thus, recording transaction time can also be viewed as recording the history of corrections. This means that data time stamped with transaction time provides for querying the history of data manipulations and errors, whereas data time stamped with valid time allows the querying of value changes. Classical/non-temporal data model updates do not differentiate between a correction and a change of data (*Date et al., 2003*). It makes perfect sense to query the history of corrections just as the history of the real world is queried. A temporal DBMS thus should be able to interpret transaction-time timestamps during query evaluation or constraint checking in the same way as it interprets valid-time time stamps. So, valid time and transaction time can be viewed as orthogonal time lines. Values for transaction time cannot be later than the current time, since transaction time reflects the time when a database operation is actually executed. The DBMS itself records transaction time. It also does not make sense to update transaction time, since a database operation of a committed transaction can never be undone. The only way to change a committed database operation is to do an inverse transaction, which, however, is executed at a later time point and thus leads to another transaction-time record.

Data time stamping

There are two ways of time stamping (*Steiner, (1998)*: tuple and attribute. Detail about these timestamps is given below.

Tuple time stamping

Tuple time stamping is usually applied in temporal relational data models supporting only first normal form relations. Data models applying tuple time stamping add timestamps to each tuple in a relation. In case of historical data, each tuple is stamped with validity time period, denoting when the tuple as a whole was valid in the real world. In a rollback database, tuples are stamped with transaction time periods. Tuples in a bitemporal database are stamped with both valid- and transaction-time periods (*Bolour et al., 1982*). In temporal object data models, tuple time stamping is known as object time stamping (*Ferrari, 1991*). The object state is stored in a record containing atomic or complex fields (or attributes). This data structure can be viewed as a tuple. So the data structure storing an object state corresponds to a tuple in non-first normal form (Snodgrass, 1987; Ariav, 1986).

Attribute time stamping

Attribute time stamping adds timestamps to each attribute value. Values in a tuple which are not affected by a modification do not have to be repeated. So, the history of values is stored separately for each attribute. Using the schema extension approach, attribute time stamping demands that the underlying data model supports non-first normal form relations or complex objects, since all time stamped attributes are of a complex type, storing the attribute value together with its timestamp. Temporal data models applying attribute time stamping are described in (Snodgrass, 1987; Ariav, 1986; *Ahn, 1993*; *Steiner,1998*).

Temporal data models can be distinguished by whether they apply tuple or attribute time stamping (*Steiner,1998)*. Using the schema extension approach, the choice of what to timestamp is restricted by the underlying data model (*Ahn, 1993*). The same holds if we look at how data is timestamped. Typically, data may be timestamped either with a time instant, a time interval or a temporal element (*Bolour et al., 1982*). Temporal relational data models staying within first normal form are restricted to use timestamps which can be mapped to additional scalar

attribute values of a tuple. Data models supporting non-first normal form relations or objects permit complex attribute values, allowing timestamps of higher complexity to be used (*Rajak and Saxena, 2009*).

Temporal databases

In the conventional model the temporal behaviour of data has been largely neglected, being reflected only through updates while ignoring the previous states (*Codd, 1990*). Relational data model can be extended to incorporate temporal relationship (*Ben-Zvi, 1982*). In recent times lots of research has been carried out to extend the relational data model for incorporating time factor (*Johnson and Weis, 2010*). Two approaches were widely used: tuple/relation time stamping and attribute/field time stamping (*Burney et al., 2010*; *Date et al, 2003*; *Mahmood et al., 2012)*. In relation time stamping the time stamp is a field of the relational model and hence it is associated with each record. *Dong-Her et al. (2007)* and *Wahid et al. (2006*) proposed temporal relational algebra based on multiple dimensional relation time stamping to capture the temporal dimension of data.

Some temporal data models are defined over valid time or transaction time while others are defined over both transaction time and valid time (*Rajak and Saxena, 2009*). The valid time of a fact is the collected time-possibly spanning the past, present and future-when the fact is true in mini world. Database model and record information about a part of reality, termed the mini world (*Codd, 1990; Kvet and Matiasko,2012*). Varying time thus captures the temporal state of the mini-world. The transaction time of the database fact is the time when the fact is recorded in the database (*Matiasko et al., 2008*). Based on the ability to represent temporal data, there are different kinds of temporal databases (*Dong-Her et al., 2007*; *Wahid et al., 2006*).

METHODOLOGY

In order to achieve the aim of this paper, a survey of different existing temporal data model was carried out as discussed in the last section; and the schematic view of the data model is designed using entity-relationship diagram as shown in the (Figure 2). This is designed to be implemented using an object-relational database. Figure 1 shows the workflow for nursing in hospital setting. We divide the health care team into two (2): nurses and other personnel; since our interest is in nursing care. Our area of interest is therefore mapped out by the dotted rectangle. Here, when the client visits the facility, he/she is first attended to by the medical record officer at the registration office and a card is issued to the client. The client is either admitted into the facility after due consultation by the physician or book appointment with the nurse as in the case of follow-up. The nurses assess the client in the ward so as to identify the client's need/problem and a plan of care for the client is develop afterwards. The developed care plan is implemented and evaluated thereafter. If the outcomes specified in the care plan are met, the client is discharged and depending on his/her condition, he/she may come for follow-up afterwards.

RESULTS AND DISCUSSION

The proposed model is called nursing care temporal object relational (NC-TOR) data model. Our proposed model is based on (Figure1). It is represented using object-relational diagrams (a modified form of entity-relationship diagrams) depicting the various faces of the nursing process, and the stakeholders (nurses and clients) and their respective activities. The model treats time as discrete and isomorphic to the natural numbers because any practical domain that we might define for time attributes in our proposed model would have at most an infinite countable set of names for time moments or time intervals. Tuple time-stamping approach has been adopted to define a temporal model. The reason for this is the simplicity. The proposed temporal data model (Figure 2) is made up of two main constructs: the entity and the relationship type amongst the entities are n-ary. This research employed the principles of entity-relationship modeling, logical design and normalization to create an object relational database for a nursing care data database. The procedures for designing this model are as follows:

(i) Obtain a data dictionary defining and explaining the classes and properties of data and the relationships among data entities.
(ii) Study the relationships among the variables.
(iii) Resolve any questions with the domain experts at both clinical and research facilities.
(iv) Design the ER diagram according to the relationships between variables. This diagram serves as the blueprint for the normalization of the data set.
(v) Decompose the data sets into the various tables according to the ER diagram.
(vi) Verify the design with a database management expert and build and populate tables.

While developing the conceptual schema using ER notation, we focused on the idealized or model state of data. An advantage of this approach is that it allows us to develop a conceptual diagram to apply to a class of such datasets, and have it be applicable independent of the specific implementation platform or data limitations and constraints (which may differ with each application). Our ER diagram (Figure 2) defines the entity and relationship classes relevant for our nursing care dataset. The notation we use describes entity classes with rectangles and

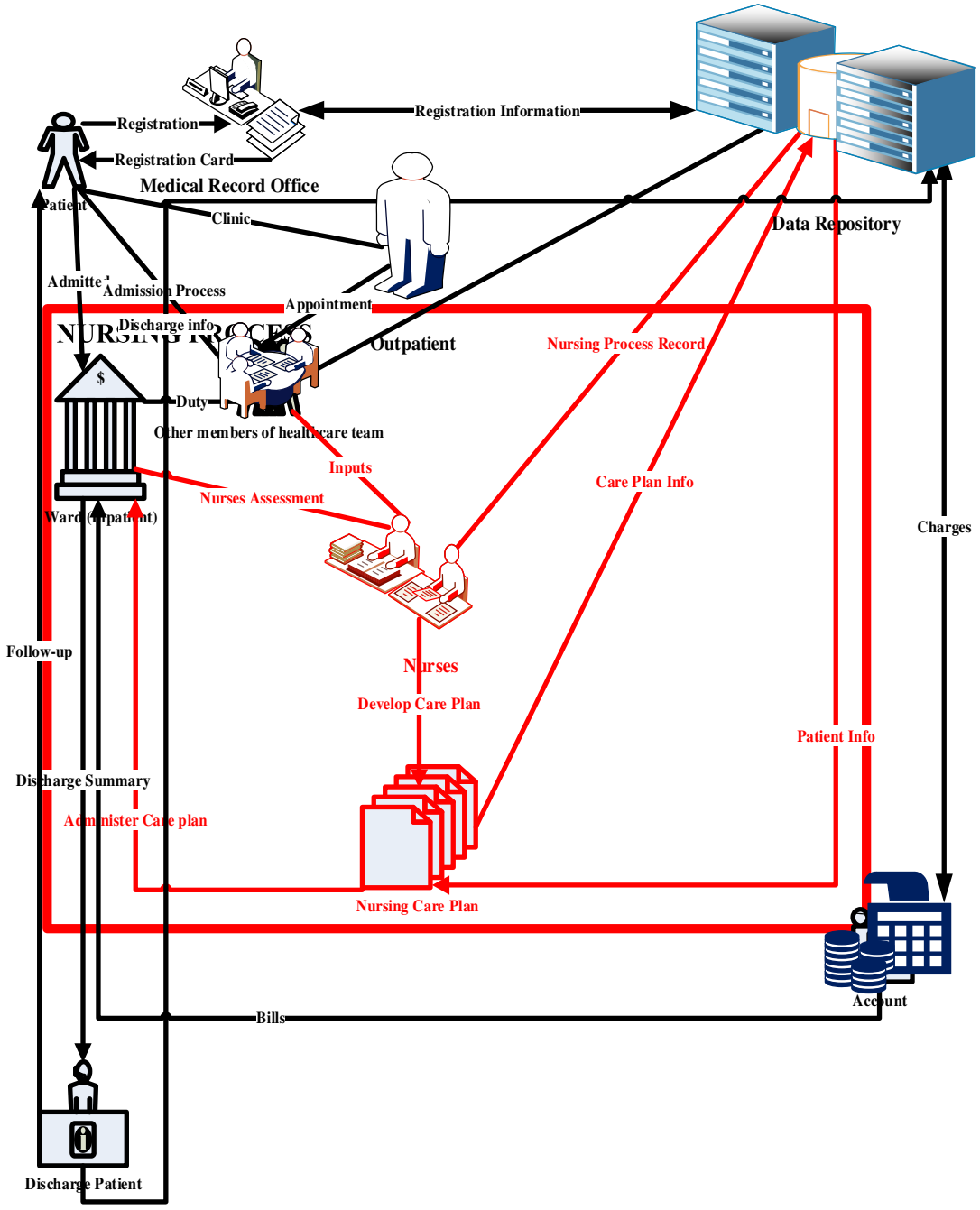

Figure 1. Workflow.

relationships with diamonds. The Diagnoses class depends on Visits and Diagnosis Types, and each diagnosis needs a single visit sequence number and nursing diagnosis type code to identify it. For convenience, we could add an artificial identifier to Diagnoses. Figure 2 does not show attributes to preserve readability. Each client has a set of demographic variables that usually do not change, and we listed them in the entity class Clients. A set of attributes describing a client's specific hospital experience which varies with time for each visit is also collected; these are contained in the entity classes Visits, Diseases, Procedures, Outcomes, Nursing Diagnoses, etc. Visits are central to our schema in that they link client visits to the nurses assigned

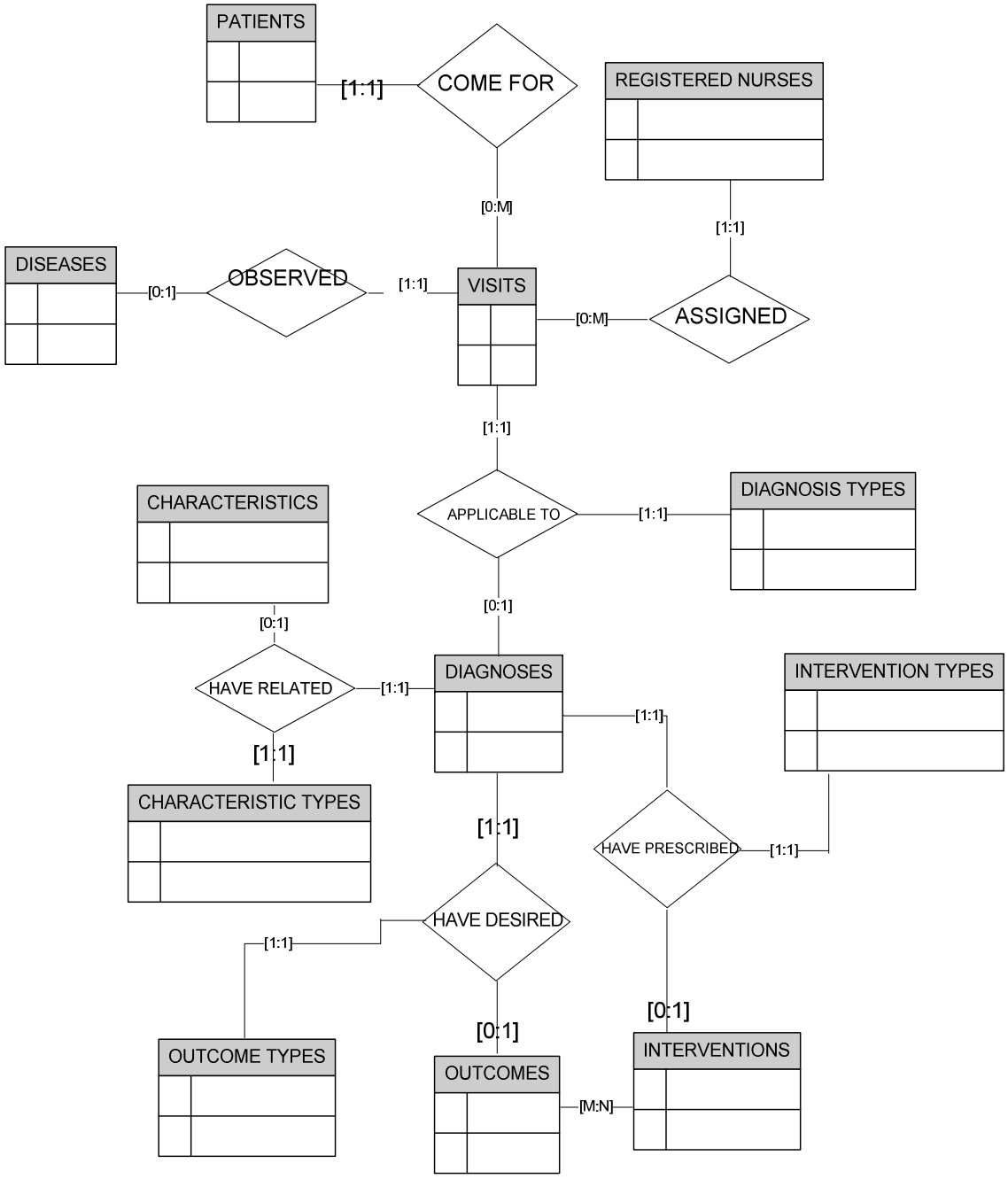

Figure 2. Schematic view of a temporal data model.

during the visit and the diagnoses delivered by them, as well as the observed medical diseases and procedures performed by the hospital.

REFERENCES

Steiner A (1998). A Generalisation Approach to Temporal Data Models and their Implementations. Ph.D Dissertation submitted to the Swiss Federal Institute of Technology. Zurich.

Bolour A, Anderson T, Dekeyser L, Wong HKT (1982). The Role of Time in Information Processing: A Survey, ACM SIGMOD RECORD, Vol. 12, pp. 28-42.

Hom DD(1988).Temporal Data, Temporal Data Models, Temporal Data Languages and Temporal Database Systems. California: Naval Postgraduate School, 1988.

Matiasko K, Vajsova M, Zabovsky M, Chochlik M(2008). Database Systems, 2008.

Mahmood N, Rizwan K, Bari SAK (2012). Fuzzy-Temporal Database Ontology and Relational Database Model, 2012.

Kvet M, Matiasko K(2012). Conventional and Temporal Table. Advanced Research in Scientific Areas, Pp.1-5.

Rajak A, Saxena K(2009). Modelling Temporal Database-A Survey

International Journal of Computer and Electronic Engineering, Pp.77-82.
Ahn, SQLT(1993). A Temporal Query Language. Proceedings of the Infrastructure for Temporal Databases, Arlington.
Ariav GA(1986). Temporally Oriented Data Model. ACM Transactions on Database Systems, 11(4), 499-527.
Date CJ, Darwen H, Lorentzos NA(2003). Temporal Data and the Relational Model. Morgan Kaufmann Publishers, 2003.
Ferrari R(1991). Temporal Extensions to the Iris Model in a Medical Domain. Hewlett Packard, Pp.1-15.
Snodgrass RT(1987). The Temporal Query Language TQuel. ACM Transactions on Database Systems, 12(2):247-298.
Codd EF(1990). A Relational Model for Database Management System. Mass: Addison Wesley, 1990.
Ben-Zvi(1982). The Time Relational Model. Ph.D. thesis, Computer Science Department, UCLA.
Johnson T, Weis R(2010). Managing Time in Relational Databases. Morgan Kaufmann.
Burney A, Ahsan K, Mahmood N, (2010). A Conceptual Temporal Relational Model for Managing Patient Data. Recent Advances in Artificial Intelligence, Knowledge Engineering and Databases, Pp.237-243.
Dong-Her S, Hsiu-Sen C, Binshan L (2007). A Generalized Associative Petri Net for Reasoning, IEEE Transactions on Knowledge and Data Engineering, 19(9):1241-1251.
Wahid TM, Kamruzzaman AZM, Shariff ARM (2006). Spatio-Temporal Object-Relational for Biodiversity System. International Journal of Computer Science and Network Security, Pp.45-53.