ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20160425214024

RELATIONAL MODEL OF TEMPORAL DATA BASED ON 6TH NORMAL FORM

Darko Golec, Viljan Mahnič, Tatjana Kovač

Original scientific paper
This paper brings together two different research areas, i.e. Temporal Data and Relational Modelling. Temporal data is data that represents a state in time while temporal database is a database with built-in support for handling data involving time. Most of temporal systems provide sufficient temporal features, but the relational models are improperly normalized, and modelling approaches are missing or unconvincing. This proposal offers advantages for a temporal database modelling, primarily used in analytics and reporting, where typical queries involve a small subset of attributes and a big amount of records. The paper defines a distinctive logical model, which supports temporal data and consistency, based on vertical decomposition and sixth normal form (6NF). The use of 6NF allows attribute values to change independently of each other, thus preventing redundancy and anomalies. Our proposal is evaluated against other temporal models and super-fast querying is demonstrated, achieved by database join elimination. The paper is intended to help database professionals in practice of temporal modelling.

Keywords: *logical model; relation; relational modelling; 6th Normal Form; temporal data*

Relacijski model vremenskih podataka zasnovan na 6. normalnoj formi

Izvorni znanstveni članak
Ovaj rad povezuje dva različita područja istraživanja, tj. Temporalne podatke i Relacijsko modeliranje. Temporalni podaci su podaci koji predstavljaju stanje u vremenu, a temporalna baza podataka je baza podataka s ugrađenom podrškom za baratanje s podacima koji uključuju vrijeme. Većina temporalnih sustava pruža dovoljno temporalnih karakteristika, ali su relacijski modeli nepravilno normalizirani, a pristupi modeliranju nedostaju ili nisu uvjerljivi. Ovim se prijedlogom daju prednosti modeliranja temporalne baze podataka, prvenstveno korištene u analitici i izvještavanju, gdje tipična pretraživanja uključuju mali podniz atributa i veliku količinu zapisa. U radu se definira posebni logički model koji podržava temporalne podatke i konzistenciju, zasnovan na vertikalnoj dekompoziciji i šestoj normalnoj formi (6NF). Primjena 6NF omogućuje neovisnost u promjeni atributnih vrijednosti i tako sprečava redundanciju i anomalije. Naš je model uspoređen s drugim temporalnim modelima i demonstrirano je super brzo pretraživanje postignuto eliminacijom spajanja baze podataka (database join elimination). Svrha je rada pomoći stručnjacima koji se bave bazama podataka u primjeni temporalnog modeliranja.

Ključne riječi: *logički model; odnos; relacijsko modeliranje; 6. normalna forma; vremenski podaci*

1 Introduction

The key resources of post-industrial society are information and knowledge [24]. Data collection and processing is very important for a company management [30]. The gathering and the preparation of data present great problems [15]. In such situations database systems can be used. A database system is a collection of interrelated data and a set of programs for accessing stored data. A well-defined format to model and store information is required. There are four well-known modelling techniques that represent data storage: a relational database, an object-oriented database, a spatial database and a temporal database [23].

Time is unique, and is measured by a clock. At a certain point in time, two clocks cannot show exactly the same time. For the representation of time in a database, a temporal database is required, which stores a collection of time-related data [23]. Such a system provides facilities for storing, querying and updating historical and future data (generically referred as temporal data) [7]. Temporal data is data that changes over time. Valid time and transaction time can be supported in coexistence and such a database is called a bitemporal database, as opposed to a non-temporal database. The temporal approach introduces additional complexities, such as dealing with model design and keys. It is difficult if not impossible to identify a substantial computer application that does not evolve over time [2]. This means that time-varying data is usually involved. Built-in time management support greatly increases the functionality of a database application. It is desirable for a database system to maintain past, present, and possibly future versions of data. A conventional database, without temporal support, stores only the most recent data. The old values are either replaced or deleted. The evolution of real-world phenomena over time cannot be recorded in the database and accessing past data is not an option. Time values are associated with attributes that indicate their periods of validity. More specifically, the temporal concepts usually include two types of time: valid and transaction time. Valid time is the time period during which a fact is true with respect to reality. Transaction time is the time when a fact is stored in the database. In this research, we cover the modelling of both concepts to present a model which is able to store current, historical and even future data. The concepts of valid time, transaction time and bitemporal data were introduced as a part of TSQL2, a language specification, developed for temporal extensions to SQL [29]. Date [5] stated that "valid times are kept in the database, while transaction times are kept in the log" (p. 302).

Temporal data is a long debate initiated in late 80-ies, but standardization happened in 2011 when ISO released SQL standard to support bitemporal data (ISO/IEC 9075:2011) [31]. An improved support for temporal databases is one of the main features. Language enhancements include time-period definition, temporal primary keys and temporal referential integrities.

The goal of the paper is to formulate a relevant logical data model, which supports both valid time and transaction time (i.e. intended for a temporal database) and also defines the relational modelling approach. Temporal attribute values change independently of each other and at different rates, and typical issues encountered in the consideration of temporal data are data redundancy and anomalies. These issues can only be resolved with high normalization and lossless decomposition. The proposal is presented as a relational logical model, whereof the physical model is addressed in evaluation part. Our objectives stem from practical experience and the finding that most temporal model proposals do not address the relational modelling in the expected manners. We believe that this paper will improve temporal modelling and offer a new and successful paradigm. The temporal model can be based on limited number of well-defined concepts. An exact formulation is possible only with considering high normalization. A different approach – using low normalization – is insufficient when temporal data is handled. Thus, the consistent use of normalization theory is a requirement. High normalization leads to the correct relational structures, which results in more lower-degree relations where consistency is guaranteed. Proposed model is based on relations in sixth normal form (6NF). 6NF is particularly used for temporal data as described later. Fifth normal form (5NF) – as the ultimate normal form in the normalization process – does not comply with temporal relations. Only existing models that include less data redundancy are considered in this paper. Such a list is limited and very few models are available. We are looking to improve the temporal logical model in a way that would better specify the modelling of both valid and transaction time and, most importantly, we are looking for highly consistent model definition.

The anticipated advantage of this proposed model is resistance to changes in the logical model whereby extensions are required instead of modifications. Modifications of the existing state are not required, because changes always result in extensions. Extensions are achieved either by new constructs or new tuples. Other advantages are its robust design approach, reduced complexity, flexibility and simplicity, and the well-considered relational theory, as discussed later.

A contribution of the paper in relation to the previous research is well defined data model intended to support temporal data modelling with combining highly normalized relations in 6NF with metadata information. The paper is authentic in a sense of a model presented as a generic schema in relational model for temporal data. Unique set of concepts supports both valid time and transaction time modelling. The innovation proposed is a formal temporal model that is based on relations in the 6th normal form with selected metadata.

The paper is organized into six sections. In section 1, the introduction has already defined the primary goal and expected advantages of the proposed approach. Section 2 reviews the related work of concepts of time, temporal data and high normalization fundamentals are all considered in turn, with common features explored. Section 3, most importantly, includes various views on the logical model, considerations or dilemmas, and propositions presented. In total eleven concepts are formulated and the proposed model is defined and explained. In section 4, the model is compared with similar models and described with the number of required relations. Furthermore, practical evaluation results are communicated pointing out outstanding results in the section 5. Section 6 covers discussion, where several views are discussed in turn and model is elaborated in details. At the end, we conclude and indicate the direction for continuing this research.

2 Related work

2.1 Concepts of time

Valid time and transaction time are the most typical concepts of time, but also other concepts of time can be considered, each of which is stored in the database. Anchor Modelling defines changing time, recording time and happening time [26]. The name tries to capture what the time presents: when a value is changed (changing time), when information was recorded (recording time) and when an event happened (happening time). Another concept of time is user-defined time, whose domain is time, which is not interpreted by DBMS [3]. User-defined time is treated like any other ordinary data.

In the field of commercial DBMS implementations Oracle 12c DBMS supports horizontal types, which are categorized on the combination of valid time, transaction time and decision time. Decision time describes the date/time when decision has been made. Decision time is independent of an entry into the database and is not directly related to the valid time. Further, Oracle temporal feature Workspace Manager defines create-time and retire-time presenting validity period of the transaction time. IBM DB2 V10 claims to be the first database to have a conforming implementation of SQL:2011 named Time Travel Queries, and Microsoft SQL Server 2016 implements temporal tables with a feature called SYSTEM_VERSIONING.

As described, concepts of time are well established in commercial DBMSs since 2011, but always the same difficulty appears – no modelling definition exists. Temporal modelling approach is missing. Our foregoing proposal gives an extra opportunity – it is not only about differentiating concepts of time, but rather about modelling them in the formulated and theoretically correct way.

2.2 Temporal data modelling

Four categories of temporal database have been identified with respect to valid and transaction times: snapshot, historical, rollback and bitemporal [2]. Snapshot databases store the single state of the real world (usually the most recent state), historical databases store data with respect to valid time, rollback databases store data with respect to transaction time and bitemporal databases store data with respect to both valid and transaction time [28].

The technique proposed by Halawani and Al-Romema [10] suggests the implementation of a temporal database on top of an existing non-temporal database. Their data model is based on tuple time stamping with two relations: one relation is the current snapshot of data and the other is the auxiliary relation that holds the temporal aspects of the whole of the time-varying attributes. A similar proposal from Date et al. [7] defined a model with two sets of relations: one for the current state of affairs and one for the history. The temporal relation with current information is normalized in 5NF and the temporal relation with historical information is normalized in 6NF. The current relation in 5NF has the attribute SINCE (point in time data type), which is used for each and every temporal attribute; the number of SINCE attributes is equal to the number of temporal attributes, while non-temporal attributes do not contain corresponding SINCE attribute. The candidate key is simple and does not include any of SINCE attributes. The historical part is decomposed into several historical relations. The number of historical relations is equal to the number of temporal attributes. Historical relations include the attribute DURING (interval data type). The candidate key is composed and includes the DURING attribute.

Johnson is author of Asserted Versioning Framework (AVF) [13] which offers support for bitemporal data in the form of a middleware product. Unlike vendor-specific solutions, it is an enterprise solution that provides a consistent and queryable implementation across databases managed by different DBMSs. Unlike vendor solutions, the AVF supports an extension of transaction time, named assertion time in this work, so that data can be created and managed in future assertion time. Using AVF, temporal requirements do not have to be expressed in data models and no procedural logic is needed in application programs in order to maintain temporal data. In AVF all temporal data is contained in production tables, and none of it is scattered around various other physical or logical datasets. In another work the same author demonstrates by way of a comprehensive example that much historical data is lost, while maintenance of Slowly Changing Dimensions schemas can be very laborious and expensive [14].

Novikov and Gorshkova stated that problems concerning the representation of the date and time in databases have been well studied in scientific literature, but commercial systems and standards for the query language do not support temporal features [22]. Nowadays temporal features are DBMS supported and well researched, but modelling techniques are not.

2.3 Temporal data modelling in a data warehouse

The notion of time pervades every aspect of the data warehouse [17]. As several mature implementations of data warehousing systems are fully operational, a crucial role in preserving their up-to-dateness is played by the ability to manage the changes that the data warehouse schema undergo over time in response to evolving business requirements [9]. Malinowski and Zimányi realized that there is no well-accepted model for either data warehouses or temporal databases that can be used to capture users' requirements [21]. Abelló and Martin [1] analysed the correspondences between the temporal attributes of the data sources and those of the data warehouse. Depending on whether the data sources manage valid time or transaction time, valid time may or may not be obtained for the data warehouse. Transaction time in the data warehouse can always be obtained, because it is internal to a given storage system [1].

The temporal aspect of the data warehouse is considered in the concept of the Slowly Changing Dimension (SCD) [17]. Dimension tables SCD Type 1, 2 and 3 are in the second normal form, which is very different than using 6NF. Dimension tables SCD Type 4, referred to history tables, divide data between a table keeping current data and an additional table keeping changes. Method resembles how database audit or change data capture techniques function. Another modelling method, the Data Vault [19], provides the long-term historical storage of data coming in from multiple data sources. It is a hybrid approach encompassing the "best of breed" between the third normal form and star schema. Temporal data is well addressed, but the model is less normalized. Another approach, Data Warehousing 2.0 (DW 2.0) is a second-generation attempt to define a standard for data warehouse architecture. A key feature introduced in DW 2.0 is the ability to support changes of data over time. By separating semantically static data from semantically temporal data, systems can gracefully accommodate change [12]. The designers of DW 2.0 pointed out that, in an environment, where temporal and non-temporal data are grouped together, every time there is a change in business requirements the technology infrastructure goes haywire [12]. When trying these proposals in real world implementations, they normally disclose the complexities and uncertain modelling situations.

2.4 Research in the field of high normalization

A denormalized model and therefore an opposite approach, proposed by Anselma [2], presents a temporal relational model in the first normal form, together with new relational algebra to query it. In accordance with the relational theory, only efficient structures, achieved by high normalization, can prevent redundancy of the temporal data. The idea of decomposing relations as far as possible is motivated by the desire for a reduction to the simplest possible terms (meaning that no further nonloss decomposition is possible); in other words, it represents the desire for reduction to irreducible components [11]. 6NF aims to decompose relations to irreducible components – irreducible relations. Thus, irreducibility is an important characteristic, and such relations cannot be decomposed further without losing information. A more normalized model results in an increased number of relations. While 6NF may be unimportant for non-temporal data, it is certainly important when maintaining data containing temporal variables of a point-in-time or interval nature [18].

In the field of relational database theory, 6NF has been used to describe two different terms, presenting two different meanings. The first term is related to Khodorovskii's definition [16] and the second use of the term originates from a book by Christopher J. Date and others. In this work, the relational operators are generalized to support interval data. 6NF is based on an extension of the relational algebra. Relation r is in 6NF if and only if it satisfies no nontrivial join dependencies at all [7] (p. 176) where join dependency $*\{A, B, \ldots, Z\}$ is satisfied if and only if every legal value of r is equal to the join of its projections on $A, B, \ldots, Z$ – that is, if and only if r can be nonloss-decomposed into those projections. Nontrivial join dependency $*\{A, B, \ldots, Z\}$ is satisfied if none of $A, B, \ldots, Z$ is r. In the rest of this paper, term describing nontrivial join dependencies is being used.

Carpenter [4] defined the General Return on Investment of Normalization, which is the relative amount of future anomalies being removed. The relative amount of anomalies being removed is largest with the first normal form, and then it decreases with each subsequent normal form. 6NF is different; temporal data with 6NF-type anomalies are much more common in data warehousing installations than in databases used primarily for daily operations. Not normalizing to 6NF is a different matter from not normalizing to 5NF. Therefore, the return on investment might not be justified unless the firm is designing a data warehouse. To rephrase the statement – return on investment for 6NF is justified when temporal data is considered.

6NF is currently being used in some data warehouses using the Anchor Modelling technique. This technique offers mechanisms for non-destructive extensibility, thereby enabling the robust and flexible management of changes in source systems [26]. Although using 6NF leads to an explosion of tables, modern databases can prune the tables by select queries using a join elimination to eliminate those that are not required and thus speed up the queries that only access several attributes. However, Anchor Modelling is focused especially in data warehousing, non-trivial join dependencies are available in tables and 6NF is partly used, meaning that theoretical principles of the relational theory could be closely considered. Additionally, transaction time is unsupported in the definition of the model.

3 Model definition

3.1 Why to consider 6NF instead of lower normal forms?

Temporal data requires a modern design approach, way beyond the conventional design wisdom called denormalization. Temporal data suffers from certain innate complexities [7]. The nature of temporal data issues is specific. Typical issues are redundancy, circumlocution, contradiction, homogeneity. As the answer, 6NF allows attribute values to change independently of each other, thus preventing redundancy and anomalies. Independence can be achieved with irreducible components. When dealing with temporal data, 6NF is recommended, and no other normal form can be a replacement. We categorically reject lower normal forms in the forgoing proposal. Lower normal forms are incapable of resolving temporal data issues, irreducibility cannot be achieved. One feature that sharply distinguishes our proposal from earlier ones is that it is firmly rooted in the relational theory, which earlier ones mostly were not. It is crucial for any database professional to understand the fundamentals of the database field, and these fundamentals are called relational theory. Solid theoretical principles are applied in this proposal. The proposal is forward-looking, in the sense that describes how temporal data modelling should act in the future. Our proposal is not concerned with commercial products, nor with well-established concepts or SQL language, which are categorically not the same thing. Our proposal plugs the gaps and resolves the problems that temporal data gives rise to, with exact definitions and explanations why temporal modelling is unique and appropriate for 6NF only.

3.2 Remark about physical design

Important remark about physical design is needed before narrowing down the logical design. A relational model has deliberately got nothing to say about physical design. Physical design, unlike logical design, is DBMS dependent. Physical design is derived from the logical design and not the other way around. The goal is to map into physical structures supported by target DBMS. Therefore, the high number of relations does not necessarily mean a high number of tables, thus cannot degrade performance. In principle, logical design has absolutely nothing to do with performance at all [5]. Physical constraints e.g. availability or performance are therefore physical limitations and should be kept aside.

High normalization based on 6NF leads to narrow relations. In worst-case scenario – direct image implementation – with translating relations in tables, numerous narrow tables are consequently required. The benefits are narrow tables causing less I/O activities and data to scan, which are individually better queried. With keeping a small number of tables in a query, narrow tables are more efficient performance-wise, compared to wide tables in less normalized structure. The results are demonstrated in practical evaluation section.

3.3 Design considerations and propositions

No existing concept precisely formulates the structure of bitemporal relations with consideration to data redundancy and anomalies based on high normalization. Different approaches have been proposed [7], [10], [17], but guidance or exact definitions are unavailable, as there are too many unclear scenarios when trying such approach in design. This section describes considerations and propositions, presenting foundations for our model.

A model is an abstract, self-contained and logical definition of the data structures, data operators and so forth that together make up the abstract machine with which users interact [5]. The aim of temporal logical model is to accommodate the nature of time naturally and directly, and to avoid the ad hoc one-off extensions commonly employed in information systems design [27]. Fundamental research questions are: "How to design a logical model to support temporal (historical, current and future) information?" and "How to formulate a model definition?"

Our foregoing proposal – relational model of temporal data based on 6NF – treats current and historical information in the same way. Any kind of temporal data (current or historical information) is located within a single temporal relation. Thus, an approach is adopted that is based on vertical decomposition. This approach semantically distinguishes between non-temporal and temporal data, but does not distinguish between current and historical information. The statement seems perhaps slightly oversimplified, but it embodies an essential principle. Non-temporal data does not need temporal information (data is static anyway, e.g. birthdate, gender). If non-temporal data is ever changed, the change is most likely to be the result of the correction of an error in the original tuple and not due to an actual change in the transaction itself. The DW 2.0 paradigm is considered in the sense that non-temporal and temporal data are entirely separated in the proposed model. Proposition 1: Our proposal semantically distinguishes between non-temporal and temporal data, but not between current and historical information.

6NF benefits by definition occur when the relation has at least one interval attribute in the candidate key. If interval is closed later in the future, then closing is achieved with necessary modification of the existing tuple. A candidate key is not stable over time. Our belief is that benefits can occur also when no interval attributes are available. Manipulation of tuples can be different. We can avoid costly modifications in a scenario when a single point in time is suggested instead of interval. This results in zero-update approach and benefits occur again. Our model has an explicit attribute "point in time from", but there is no explicit attribute "point in time to". Information for "point in time to" may be available in "point in time from" of the later tuple. If no later tuple is available, then the (last) value "point in time from" is still valid. To guarantee enough detailed granularity, TIMESTAMP data type is suggested which has granularity of 1 millisecond. In this proposal, tuples can be managed as additions only, not modifications. The explained benefit outweighs the drawback of considering more than one tuple when searching for the period in which the corresponding value is valid. Proposition 2: Our proposal treats temporal data with "point in time from" type. Proposition 3: Extensions are supported, but not modifications.

In temporal relation, only subset of attributes may change over time and the frequency of change is important. Vertical decomposition plays a role. With regard to that, non-temporal relations should comply with classical normalization theory with 5NF, but temporal relations should comply with 6NF, where vertical decomposition is also required. However, for the purpose of this paper, 6NF is consistently used for non-temporal and temporal data, although normalizing non-temporal data creates an overhead. Assuming that our proposal fits particularly well with temporal data, then the occasional normalization of non-temporal attributes is not too critical. Gadia proposed a homogeneous relational model where normalization leads to homogeneous relations with an identical validity for all attributes in a given tuple of a temporal relation [8]. By using only 6NF, the proposed model is in accordance with Gadia's concept and therefore homogeneous. Proposition 4: Temporal data is addressed in particular, therefore 6NF is proposed for the modelling.

The temporal part of the research aim is especially interesting. By looking in detail, the design of valid and transaction times must be clarified. Transaction time can never refer to the future. In our model, valid time is captured by the VALID_FROM attribute and transaction time by the LOGGED_TIME attribute. There is no concept VALID_TO in order to avoid modifications of the existing tuple. Valid time exists only for temporal attributes and transaction time exists always. Proposition 5: Both valid and transaction times are supported.

Additional information can be captured (i.e. person loading data in the database, data source name etc.). These attributes encompass metadata information. We decide to support metadata information as part of non-temporal and temporal data. To avoid duplicates and redundancy, metadata information is normalized as well. In line with the decision to pursue normal form, 6NF is used across the entire model. Vertical decomposition results in additional set of relations that support metadata attributes. To make a reference between regular and metadata attributes, ID_METADATA foreign key is defined. The relationship is required and a referential integrity rule must exist. Referential integrity is one of two originally formulated generic integrity constraints and means that there must not be any unmatched foreign key values. The same is valid for the entire model. Proposition 6: Additional metadata information is captured in the model in order to increase information available.

3.4 Definition of the relational model of temporal data based on 6NF

The entire relational model of temporal data based on 6th normal form is formulated by the following concepts C_x. In general, key always refers to the candidate key, which presents a minimal irreducible set of attributes where uniqueness is still guaranteed [6]. Normalization is based on dependencies identified between attributes.

- C_1 (Identity): Identity *ID* is an infinite set of unique symbols. *ID* can be of any data type.
- C_2 (Data type): *D* is a data type with domain of data values. *D* can be integer, char, etc.
- C_3 (Time type): *T* is a time type with domain of time values. *T* can be date, time, timestamp etc.
- C_4 (Data key): C_4 is a binary relation with degree of two (A_1, A_2), where corresponding domains are *ID*, ID. Key of C_4 is A_1.
- C_5 (Metadata key): C_5 is a binary relation with degree of two (A_1, A_2), where corresponding domains are *ID*, *T*. Key of C_5 is A_1.
- C_6 (Non-temporal Metadata Attribute): C_6 is a binary relation with degree of two (A_1, A_2), where corresponding domains are *ID*, *D*. Key of C_6 is A_1. C_6 can present Person loading, Source loading etc.
- C_7 (Non-temporal Attribute): C_7 is a relation with degree of three (A_1, A_2, A_3), where corresponding domains are *ID* for data key, *D* for data type, *ID* for metadata key. Key of C_7 is a combination of A_1, A_3.
- C_8 (Temporal Attribute): C_8 is a relation with degree of four (A_1, A_2, A_3, A_4), where corresponding domains are *ID* for data key, *D* for data type, *T* for time type, *ID* for metadata key. Key of C_8 is a combination of A_1, A_3, A_4.
- C_9 (Non-temporal Link): C_9 is a set of at least two data keys and one metadata key. Corresponding domains are *ID* for data keys and metadata key. Non-temporal link $C_9(A_1, A_2, ..., A_n, A_p)$ relating a set of data keys $C_4(A_m)$ and metadata key $C_4(A_p)$ is a relation with $n \geq m$ and minimal degree of three ($n \geq 2+1$). Key of C_9 is a set of (A_1, A_2, ..., A_n, A_p) including A_p. Key has at least three attributes.
- C_{10} (Temporal Link): C_{10} is a set of at least two data keys, one time type and one metadata key. Corresponding domains are *ID* for data keys, T for time type, *ID* for metadata key. Temporal link $C_{10}(A_1, A_2, ..., A_n, T, A_p)$ relating a set of data keys $C_4(A_m)$,

time type $C_3(T)$ and metadata key $C_4(A_p)$ is a relation with $n \geq m$ and minimal degree of four ($n \geq 2+1+1$). Key of C_{10} is a set of (A_1, A_2, ..., A_n, T, A_p) including T, A_p. Key has at least four attributes.

- C_{11} (Non-temporal Metadata Link): C_{11} is a binary relation with degree of two (A_1, A_2), where corresponding domains are ID, D. Key of C_{11} is a combination of A_1, A_2.
- C_{12} (Relational model of temporal data based on 6NF): Model is a set of concepts. $C_{12} = \{C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}\}$, where C_4 is a data key, C_5 is a metadata key, C_6 is a non-temporal metadata attribute (e.g. person loading, source loading), C_7 is a non-temporal attribute, C_8 is a temporal attribute, C_9 is a non-temporal link, C_{10} is a temporal link and C_{11} is a non-temporal metadata link.

Concepts present corresponding relations with a key and a set of attributes. Relational model of temporal data based on 6NF is build-up of relations that are defined respectively as follows.

- C_4 (Data key):
VAR C_4 RELATION $\{A_1\ C_1\#, A_2\ C_5\}$ KEY $\{A_1\}$;
- C_5 (Metadata key):
VAR C_5 RELATION $\{A_1\ C_1\#, A_2\ C_3\}$ KEY $\{A_1\}$; where A_1 may call *ID*_METADATA, A_2 may call LOGGED_TIME.
- C_6 (Non-temporal Metadata Attribute):
VAR C_6 RELATION $\{A_1\ C_1\#, A_2\ C_2\}$ KEY $\{A_1\}$; where A_1 may call ID_SOURCE_LOADING, A_2 may call SOURCE_LOADING or A_1 may call ID_PERSON_LOADING, A_2 may call PERSON_LOADING.
- C_7 (Non-temporal Attribute):
VAR C_7 RELATION $\{A_1\ C_4, A_2\ C_2, A_3\ C_5\}$ KEY $\{A_1, A_3\}$; where A_3 may call ID_METADATA.
- C_8 (Temporal Attribute):
VAR C_8 RELATION $\{A_1\ C_4, A_2\ C_2, A_3\ C_3, A_4\ C_5\}$ KEY $\{A_1, A_3, A_4\}$; where A_3 may call VALID_FROM, A_4 may call ID_METADATA.
- C_9 (Non-Temporal Link):
VAR C_9 RELATION $\{A_1\ C_4, \ldots, A_n\ C_4, A_p\ C_5\}$ KEY $\{A_1, \ldots, A_n, A_p\}$; where $n \geq 2$; $p=n+1$, A_p may call ID_METADATA.
- C_{10} (Temporal Link):
VAR C_{10} RELATION $\{A_1\ C_4, \ldots, A_n\ C_4, A_p\ C_3, A_r\ C_5\}$ KEY $\{A_1, \ldots, A_n, A_p, A_r\}$; where $n \geq 2$; $p=n+1$, $r=p+1$, A_p may call VALID_FROM, A_r may call ID_METADATA.
- C_{11} (Non-Temporal Metadata Link):
VAR C_{11} RELATION $\{A_1\ C_5, A_2\ C_6\}$ KEY $\{A_1, A_2\}$; where A_1 may call ID_METADATA, A_2 may call ID_SOURCE_LOADING or ID_PERSON_LOADING.

4 Model comparison

In this section our temporal model is compared with Anchor Modelling [26] and 5NF/6NF proposal [7]. Both similarities and differences can be considered in turn, and comparison can be made describing them.

4.1 Comparison of characteristics in temporal models

Specific characteristic of our model is a consistent usage of 6NF, which is not the case elsewhere. Anchor Modelling includes 6NF in the logical model definition, but transaction time is missing. Anchor Modelling defines near 1-1 relationship between all levels of modelling, simplifying the need for translation logic in order to move between them [24]. This is not exactly true, because differences can be found comparing logical and physical models and 6NF is not used in accordance with logical model definition. Anchor Modelling is focused in data warehouse modelling in particular. Also 5NF/6NF proposal is very different, because separation between current and historical relations is suggested, using 5NF or 6NF respectively. Also temporal intervals are used in 5NF/6NF proposal, but not in our proposal, where point in time is used. A comparison of certain characteristics of the temporal models is presented in Table 1. Our proposal is marked as unsuitable for non-temporal data, incapable of distinguishing between current and historical information, and weak of its physical design, whilst the model's comparative advantages are its exact formulation, high resistance on changes, transaction time support and high normalization.

Table 1 Model comparison

Characteristic	5NF/6NF	Anchor model	Our proposal
Suitability for non-temporal data	Yes	Partial	No
Support for transaction time	Yes	No	Yes
Time intervals used	Yes	No	No
Normal form used	5NF in current relation, 6NF in historical relations	5NF and 6NF	6NF
Separation of current/historical data	Yes	No	No
Consideration of the physical design	No	Yes	Partial
Definition of operators, constraints	Yes	No	No
Resistance to changes	Middle	High	High
Exact number of required relations	No	Yes	Yes
Specific to data warehousing	No	Yes	No

4.2 Comparison of numbers of relations in temporal models

The number of relations in our logical model is compared with the number of required relations in other temporal model types using high normalization, that is, the Anchor model and the 5NF/6NF proposal. A temporal relation with a simple key is being addressed where attributes are affected by the passage of time. Date stated that "logged times must be made available (along with the data they refer to) in standard relational form" [7] (p. 308), where logged time is a synonym for the transaction time. Regular relation R has an associated logged-time relation R' with the same heading, except that it contains an additional attribute called X_DURING. The 5NF relation for current information and 6NF relation for historical information follow the same concept, which presents a doubled number of relations. The Anchor model defines anchor notion with all attributes normalized in standalone tables. This results in a number of tables equal to the number of attributes in a relation plus one additional anchor table. Our proposal considers 6NF normalization for all attributes in a relation. Moreover, transaction time metadata is included, and is handled separately and normalized as well. Together with the data key, the number of relations in the model is equal to the number of attributes in a relation plus five additional relations, if the regular relation has a simple key defined. When no simple key is defined, the data key becomes a surrogate attribute and the number of relations in the model is equal to number of attributes in a relation plus six additional relations. The number of relations in the Anchor model and our proposal is independent of the number of temporal attributes, because attributes are always normalized.

Tab. 2 shows a comparison for a number of required relations of three different systems. By freezing the number of attributes in relation, numbers of temporal attributes in relation are varying from 1 to all. The results of eight different scenarios are considered. We can conclude that, compared to the 5NF/6NF, our proposal has less relations required when a regular relation has a higher degree and majority of temporal attributes. On the contrary, our model and the Anchor model do not perfectly fit with non-temporal attributes.

Table 2 Number of required relations in the model

Scenario	Number of attributes in relation	Model Type	Number of temporal attributes in relation							
			1	2	3	4	7	10	15	30
1	1	5NF/6NF	2							
		Anchor model	2							
		Our proposal	5							
2	2	5NF/6NF	4	4						
		Anchor model	3	3						
		Our proposal	7	7						
3	3	5NF/6NF	4	6	6					
		Anchor model	4	4	4					
		Our proposal	8	8	8					
4	4	5NF/6NF	4	6	8	8				
		Anchor model	5	5	5	5				
		Our proposal	9	9	9	9				
5	7	5NF/6NF	4	6	8	10	14			
		Anchor model	8	8	8	8	8			
		Our proposal	12	12	12	12	12			
6	10	5NF/6NF	4	6	8	10	16	20		
		Anchor model	11	11	11	11	11	11		
		Our proposal	15	15	15	15	15	15		
7	15	5NF/6NF	4	6	8	10	16	22	30	
		Anchor model	16	16	16	16	16	16	16	
		Our proposal	20	20	20	20	20	20	20	
8	30	5NF/6NF	4	6	8	10	16	22	32	60
		Anchor model	31	31	31	31	31	31	31	31
		Our proposal	35	35	35	35	35	35	35	35

Fig. 1 graphically evaluates all models together in a trend chart. Shown are numbers of required relations in the model for a relation containing 15 attributes (scenario 7).

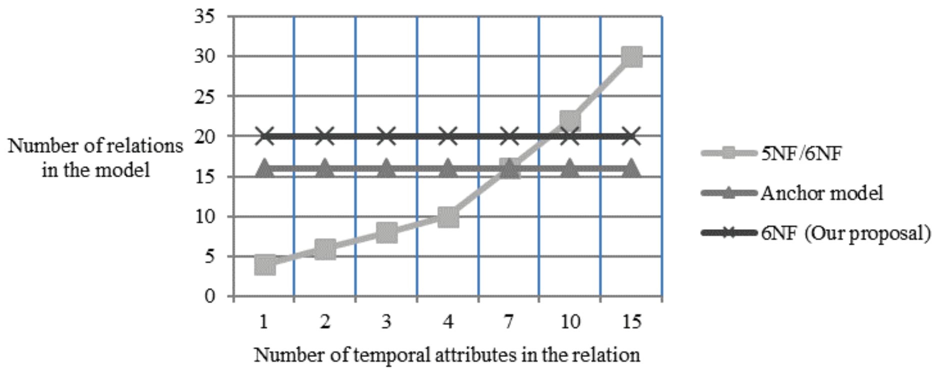

Figure 1 Number of required relations in the model for a relation containing 15 attributes

The number of relations in the model is based on a number of temporal attributes. A relation with attributes X, $X \subseteq N$, $X \neq 0$ has a subset of temporal attributes Y, $Y \subseteq N$, $Y \neq 0$, $Y \leq X$. A model has number of relations Z, $Z \subseteq N$, $Z \neq 0$. Number of relations is:

– In Anchor model:

$$Z = X + 1 \qquad (1)$$

– In our proposal:

$Z = X + 5$ when simple key is available or $Z = X + 6$ when no simple key exists. (2)

– In 5NF/6NF:

$Z = 2 * Y + 2$ when $Y < X > 1$, $Z = 2 * Y$ when $Y = X > 1$ or else $Z = 2$. (3)

5 Practical evaluation

Practical evaluation considered a comparison between nontemporal table, temporal table and our proposal – temporal model in 6NF. Nontemporal table with 15 columns and a simple key was in the third normal form and without any kind of temporal logic. Old value is overwritten by new value, and the history cannot be tracked. Temporal table was also in the third normal form, but with temporal logic included on top of 15 columns. Historical data is tracked by creating multiple records for a given natural key with separate surrogate keys. Changes result both in update of the existing records and additionally in the new records inserted. For testing of temporal table performance, Oracle Workspace Manager (OWM) package was used, which provides a virtual environment to isolate collection of changes and to keep a history. Last model was our proposal, where nontemporal table was normalized according to the definition of the relational model of temporal data based on 6NF. No nontrivial join dependencies existed, which caused the modelling of 20 tables and 50+ columns.

Table 3 Response times of the practical evaluation

Response time (seconds)	Number of (updated/queried) columns	Nontemporal table	Temporal table	Our proposal
INSERT 100k records	3	46	1190	76
	5	57	1238	97
	7	58	1251	113
	10	68	1276	171
	15	108	1312	435
INSERT 1M records	15	1430	54697	2745
UPDATE 100k records		as update	as update+insert	as insert
	1	27	58	38
	2	28	77	61
	3	28	87	80
	5	28	95	169
	7	39	96	176
	10	67	135	309
SELECT 100k records	1	0.05	0.22	0.02
	2	0.17	0.25	0.04
	3	0.22	0.28	0.05
	5	0.51	0.58	0.11
	7	0.63	1.10	0.18
	10	1.25	1.61	0.29
	15	2.46	2.92	0.45
SELECT 1M records	1	4.1	0.7	0.13
	2	4.3	0.8	0.24
	3	4.8	1.1	0.42
	5	5.2	4.5	1.06
	7	8.5	14.6	1.81
	10	13.0	22.8	4.31
	15	20.1	34.2	5.94
DELETE 100k records	Number of records	as delete	as update+insert	as insert
	10k	0.5	4	0.4
	50k	6	23	13
	100k	37	117	45

The tests were performed using Intel Core 2 Quad 2.6 GHz, 8GB of RAM and one HGST disk. The DBMS used was Oracle 12g EE. Setup was accomplished in accordance with Oracle's recommendations for OWM. The input data was based on random values, generated by Oracle DBMS package. Amount of inserted records was either 100k or 1M. This generated data was used as the only data source in all three models, thus data was exactly the same. Also the commands in SQL scripts were transparently used across all models.

The goal of the testing was to evaluate performance by measuring response times for SQL DML operations (INSERT, UPDATE, SELECT, DELETE). A key focus was on SELECT from the performance perspective and on INSERT from the design perspective. To accelerate a performance, join elimination technique was used. Transformation of the optimizer removes redundant tables/joins from the query. A table is redundant when columns are referred only to join predicates, and is guaranteed that those joins will not expand nor filter the resulting rows. Workloads of queries on designed models of different sizes were executed and then benchmarked for response time. Each test was repeated three times and response times were averaged.

Detailed results are displayed in Tab. 3. Response time numbers present how many seconds operation was taking. Each of the areas presents a particular DML operation. Numbers of columns are sequentially increasing: a) number of created columns in INSERT, b) number of updated columns in UPDATE and c) number of queried columns in SELECT. In case of DELETE, percent of deleted records was considered. Total number of columns for UPDATE, SELECT and DELETE was 15 and did never change.

In a temporal table and our proposal records are not deleting. Instead, UPDATE and INSERT are used in a temporal table, and INSERT is used in our proposal. In our proposal records are not updating either. UPDATE and DELETE operations result in INSERT, which presents an important design feature. Important behaviour to observe is that any non SQL DML operation applies always to INSERT, thus no updates are ever needed in our proposal, making the temporal model resistant to changes and non-destructive.

Visual representation of the results is available in Fig. 2 where utmost important results – SELECT response times when querying 1M records are extracted. Query performance is a key factor of successful database application [21]. Proposed model performs better when in fact we deal with a) temporal data, b) SELECT operation, c) high number of records and d) small number of queried columns. In such cases, querying becomes super-fast as illustrated in both charts. Super-fast querying is achieved by narrow tables causing less I/O activity and data to scan.

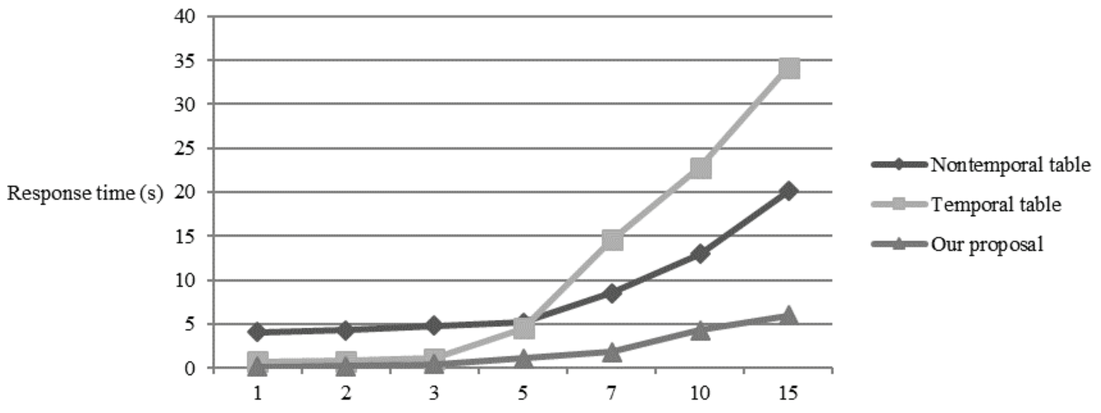

Figure 2 Response times achieved when querying 1M records

6 Discussion

An important characteristic of the proposed model is its simplicity. The model is well defined with a total of 11 distinct concepts (C_1 – C_{11}), which leads to less options being available, clear guidelines and reduced complexity. The number of distinct concepts, defined as a relation, is eight (C_4 – C_{11}). A small number of distinct concepts results in better reusability and automation, where common design patterns can be reused and automated in advance to accelerate implementation. The concepts are atomic; thus, logical design can be accomplished in parallel. Also data loading as part of the physical model can be executed in parallel, so that large implementations can scale out without the need of a major redesign.

The logical model definition consists of identity, type, key, attribute and link concepts. Non-temporal and temporal links are used to support transactions. The design guideline is to design the transaction as a link, if the transaction has no properties, otherwise design the transaction as an attribute. Data keys C_4 are binary relations, used to refer from other attributes. They are present in the link definitions. ID# refers to unique value, but does not necessarily refer to the pointer of the artificial or surrogate key, which is prohibited in the relational model. The surrogate key involves exactly one attribute and carries no additional meaning of any kind. ID# can be of any data type. References are solved by referential integrity and no database is ever allowed to contain any unmatched foreign key values.

According to the definition in [8], the relations in the model are homogeneous, which differs from the proposal [7], where a different SINCE value may be used for each temporal attribute in a temporal relation containing current information. In that sense, our proposal is comparable with the data model discussed in [8], where the temporal domain within a tuple does not change from one attribute to another.

Valid time has important role in temporal attributes and links, while transaction time plays a role in non-temporal and temporal attributes and links. The proposed model corresponds to a rollback database when non-temporal data is handled and to a bitemporal database when temporal data is handled. Valid time and transaction time are separated in the model; also, non-temporal data and temporal data are separated. High independency is achieved and the concept of isolated semantic temporal data in the next generation DW 2.0 [12] is taken into account.

The model definition includes no intervals, since intervals increase the complexity when values are changed, affecting updates. Essentially, no updates are needed in our proposal. Values are not replaced by new values. Any data change implies insert and data structure change implies model extension. Updates may only occur when fixing errors. Another advantage of not using intervals is that there is no need to check redundancy and circumlocution for temporal relations in order to satisfy 6NF.

Non-trivial join dependencies cannot be found in any of the relation definitions: irreducibility is achieved and relations are in 6NF. In relation definitions C_4 – C_8, exactly one non-key attribute is used, while other relation definitions (C_9 – C_{11}) even have no non-key attributes.

Another characteristic is data auditing, which is an important security issue. Auditing can identify malicious behaviour and increase data quality. The model emphasizes the possibility of tracing where all the data came from. The information is accompanied by source loading and person loading, and values can be traced back to the source. Data auditing does not lead business applications to the lost trail of business processes [32]. To cope with auditing, the temporal data model is a solution since it tracks information together with time.

An important observation is the agile-oriented approach. Introducing agile methods into development processes represents an important challenge for many software companies [20]. This approach facilitates iterative and incremental development as it allows independent work to take place on small subsets of the model under consideration, which can later be integrated into a global model [25]. Changing requirements are handled by additions without affecting the existing parts of a model.

The data warehouse can be interesting business case. Proposed model offers advantages when the typical use is to compute aggregate values on a limited number of attributes, as opposed to trying to retrieve all or most of the attributes for a given relation.

Due to the tremendous increase in the amount of data efficiently managed by current database systems, optimization is still one of the most challenging issues in database research. Query rewrite plays an important role in optimizing complex queries where join elimination is important technique [9].

7 Conclusions

In this paper a new paradigm is offered to define the model of temporal data based on high normalization. Formulation of eight distinct relations was presented; the proposed model covered well formulated concepts – defined as 6NF relations. Though this may be relatively unimportant for non-temporal data, 6NF is of significant importance when dealing with temporal data. Any kind of denormalization negatively impacts a temporal data life-cycle and cannot successfully address temporal issues. The utmost important benefit is a fact that no updates or alters are required. Changes always result in extensions, rather than modifications of the existing state. Physical model extensions are achieved either by new tables defined or new records inserted.

The proposed model in comparison with previous research offers temporal approach based on 6NF. The correspondences between non-temporal and temporal attributes were analysed, also existing techniques and models for time evolving data were widely studied. As a result, a relational model of temporal data based on 6th normal form has been presented.

Practical implications of the temporal design can be considered for the wider community to help database professionals in their temporal modelling projects. Evaluation was performed to test how model behaves in practice compared to established approaches. Performance benefits were demonstrated with zero-update design principles and faster query response times using join elimination.

Paper limitation to acknowledge is a differentiation between logical and physical model where availability of the graphical modelling tool would assist. Also computer-driven transformation into a physical storage would be advantageous. The variety of described advantages, like independency, simplicity, flexibility, agility, auditing and consistency, present features of modern approach, intended for designing, in a robust and non-destructive way, a consistent temporal data model. Work in this paper can be taken as input for further research and important implications in the field of database modelling.

8 References

[1] Abelló, A.; Martin, C. A Bitemporal Storage Structure for a Corporate Data Warehouse. Proceedings of the 5th International Conference on Enterprise Information Systems, 2003, pp. 177-183.

[2] Anselma, L.; Piovesan, L.; Terenziani, P. A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy. // Journal of Intelligent Information Systems, 2015.

[3] Atay, C. E.; Tansel, A. U. Bitemporal Databases, Modeling and Implementation. Saarbrücken, Germany: VDM Verlag Dr. Müller, 2009.

[4] Carpenter, D. A. Clarifying normalization. // Journal of Information Systems Education. 19, 4(2008), pp. 379-382.

[5] Date, C. J. SQL and Relational Theory: How to Write Accurate SQL Code. Sebastopol, CA: O'Reilly, 2009.

[6] Date, C. J.; Darwin, H. Databases, Types, and the Relational Model: The Third Manifesto. Boston, MA, USA: Addison-Wesley, 2007.

[7] Date, C. J.; Darwin, H.; Lorentzos, N. A. Temporal Data and the relational model. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2003.

[8] Gadia, S. K. A Homogenous Relational Model and Query Languages for Temporal Databases. // ACM Transactions on Database Systems. 13, 4(1988), pp. 418-448. https://doi.org/10.1145/49346.50065

[9] Ghazal, A.; Crolotte, A.; Bhashyam, R. Outer Join Elimination in the Teradata RDBMS. Lecture Notes in Computer Science, Database and Expert Systems Applications, 2004. https://doi.org/10.1007/978-3-540-30075-5_70

[10] Halawani, S. M.; Al-Romema, N. A. Memory Storage Issues of Temporal Database Applications on Relational Database Management Systems. // Journal of Computer Science. 6, 3(2010), pp. 296-304. https://doi.org/10.3844/jcssp.2010.296.304

[11] Hall, P.; Owlett, J.; Todd, S. Modelling in Data Base Management System. Amsterdam: North-Holland, 1976.
[12] Inmon, W. H.; Strauss, D.; Neushloss, G. DW 2.0: The architecture for the next generation of data warehousing. Burlington, MA, USA: Morgan Kaufmann Publishers, 2008.
[13] Johnson, T.; Weis, R. Managing Time in Relational Databases. Morgan Kaufmann, 2010.
[14] Johnson, T. Bitemporal Data – Theory and Practice. Elsevier, 2014.
[15] Kovač, T.; Resman, M.; Rajkovič, V. The model for evaluating the influence of student participation on school quality. // Napredak. 151, 3-4(2010), pp. 335-349.
[16] Khodorovsksii, V. V. On Normalization of Relations in Relational Databases. // Programming and Computer Software. 28, 1(2002), pp. 41-52. https://doi.org/10.1023/A:1013759617481
[17] Kimball, R.; Ross, M.; Thornthwaite, W.; Mundy, J.; Becker, B. The Data Warehouse Lifecycle Toolkit, 2nd ed. Indianapolis, IN, USA: Wiley Publishing, 2008.
[18] Knowles, C. 6NF Conceptual Models and Data Warehousing 2.0. // Proceedings of the Southern Association for Information Systems Conference, Paper 27. Atlanta, GA, USA, 2012.
[19] Linstedt, D.; Graziano, K.; Hultgren, H. The New Business Supermodel. The Business of Data Vault Data Modeling, 2nd Generation Data Warehouse Architecture. Daniel Linstedt, 2009.
[20] Mahnič, V. A case study on agile estimating and planning using Scrum. // Electronics and Electrical Engineering. 5, 111(2011), pp. 123-128. https://doi.org/10.5755/j01.eee.111.5.372
[21] Malinowski, E.; Zimányi, E. A conceptual model for temporal data warehouses and its transformation to the ER and the object-relational models. // Data & Knowledge Engineering. 64, 2008, pp. 101-133. https://doi.org/10.1016/j.datak.2007.06.020
[22] Novikov, B. A.; Gorshkova, E. A. Temporal Databases: From Theory to Applications. // Programming and Computer Software. 34, 1(2008), pp. 1-6. https://doi.org/10.1134/S0361768808010015
[23] Patel, J. Temporal Database System. London, UK: University of London, Imperial College, Department of Computing, 2003. http://www.doc.ic.ac.uk/~pjm/teaching/student_projects/jaymin_patel.pdf
[24] Pejić-Bach, M.; Demonja, M. Proceedings of the Faculty of Economics and Business in Zagreb. 6, 1(2008), pp. 5-35.
[25] Regardt, O.; Rönnbäck, L.; Bergholtz, M.; Johanneson, P.; Wohed, P. Anchor Modeling, An Agile Modeling Technique using the Sixth Normal Form for Structurally and Temporally Evolving Data. // International Conference on Conceptual Modeling: 28th International Conference on Conceptual Modeling, Gramado, Brazil, 2009, pp. 234-250.
[26] Regardt, O.; Rönnbäck, L.; Bergholtz, M.; Johanneson, P.; Wohed, P. Anchor Modeling – Agile Information modeling in evolving data environments. // Data & Knowledge Engineering. 69, 12(2010), pp. 1229-1253. https://doi.org/10.1016/j.datak.2010.10.002
[27] Roddick, J. F. A Model for Schema Versioning in Temporal Database Systems. // Australian Computer Science Communications. 18, 1(2000), pp. 446-452.
[28] Snodgrass, R.; Ahn, I. A Taxonomy of Time in Databases. Proceedings of the ACM SIGMOD international conference on Management of data, New York, USA. 14, 4(1985), pp. 236-246. https://doi.org/10.1145/318898.318921
[29] Snodgrass, R.; Ahn, I.; Ariav, G.; Batory, D.; Clifford, J.; others. TSQL2 Language Specification. SIGMOD Record. 23, 1(1994), pp. 65-86. https://doi.org/10.1145/181550.181562
[30] Staniszewski, M.; Legutko, S.; Raos, P. Production data collection, exposure and analysis in a small production enterprise. // Technical Gazette. 21, 5(2014), pp. 1177-1181.
[31] ISO/IEC 9075-1:2011, Temporal SQL. http://www.jtc1sc32.org/doc/N2151-2200/32N2153T-text_for_ballot-FDIS_9075-1.pdf
[32] Waraporn, N. Database Auditing Design on Historical Data. // Proceedings of the Second International Symposium on Networking and Network Security, Jinggangshan, China, 2010, pp. 275-281.

Authors' addresses

Darko Golec, MSc
Faculty of Computer and Information Science (PhD Student)
Večna pot 113, 1000 Ljubljana, Slovenia
darko.golec@gmail.com
darko.golec@siol.net

Viljan Mahnič, PhD
Faculty of Computer and Information Science
Večna pot 113, 1000 Ljubljana, Slovenia
viljan.mahnic@fri.uni-lj.si

Tatjana Kovač, PhD
Faculty of Commercial and Business Sciences Celje
Lava 7a, 3000 Celje, Slovenia
tanja.kovac@fkpv.si