Past Indeterminacy in Data Warehouse Design

Christina Khnaisser[1,2], Luc Lavoie[1], Anita Burgun[2], and Jean-François Ethier[1,2,3] (✉)

[1] Département d'informatique, Université de Sherbrooke, Sherbrooke, Canada
`{christina.khnaisser, luc.lavoie}@usherbrooke.ca`
[2] INSERM UMR 1138 team 22 Centre de Recherche des Cordeliers, Université Paris Descartes
`anita.burgun@aphp.fr`
[3] Département de médecine, Université de Sherbrooke, Sherbrooke, Canada
`jf.ethier@usherbrooke.ca`

Abstract. Traditional data warehouse design methods do not fully address some important challenges, particularly temporal ones. Among them past indeterminacy is not handled systematically and uniformly. Furthermore, most methods published until now present transformation approaches by providing examples rather than general and systematic transformation rules. As a result, real-world applications require manual adaptations and implementations. This hinders scaleability, long-term maintenance and increases the risk of inconsistency in case of manual implementation. This article extends the Unified Bitemporal Historicization Framework with a set of specifications and a deterministic process that defines simple steps for transforming a non-historical database schema into a historical schema allowing data evolution and traceability, including past and future indeterminacy. The primary aim of this work is to help data warehouse designers to model historicized schema based on a sound theory ensuring a sound temporal semantic, data integrity and query expressiveness.

Keywords. Data warehouse design, Temporal data warehouse, Temporal indeterminacy, Missing information.

1 Introduction

Historicization is the process that consists of transforming a non-historical (database) schema into a historicized one. There are complex design problems that need to be addressed [2]. A recent survey paper [4] did not identify new innovative models successfully addressing the following gaps. First, when applied to historical data warehouses (HDW), the published methods require multiple manual steps (both in terms of design and implementation) [12]. Second, the existing solutions are based on different data models (relational, entity-relationship, object-oriented, multidimensional) and use different structures and semantics. This diminishes generalizability and large-scale adoption. Finally, many solutions for temporal data warehouses do not handle missing information.

This paper extends the Unified Bitemporal Historicization Framework[1] (UBHF), to cope with past indeterminacy. UBHF and it's time model were formally defined in [11].

In 2006, Rizzi et al. [13] wrote: "Though a lot has been written about how data warehouse should be designed, there is no consensus on a design method yet". According to our last survey of data warehouse design (DWD) methods [12] this is still valid.

Two major temporal models have emerged in the literature and in our domain of interest (clinical data warehousing). The first one is the Bitemporal conceptual data model (BCDM) a bitemporal model based on SQL. Snodgrass presents design "best practices" to build a bitemporal schema starting from a conceptual model (entity relationship) ending with SQL code (and TSQL2 [22] a temporal SQL extension). BCDM was initially defined in [9], a more recent presentation can be found in [14] and an extension to temporal indeterminacy in [3]. The second one is the Date-Darwen-Lorentzos Model (DDLM) which is based on the relational theory. It proposes three submodels, two unitemporal model and one bitemporal model based on the third manifesto relational model [5] and Allen's interval logic [1].In previous work, a comparative study [10] showed interesting similarities (and some differences) between BCDM and DDLM when using UBHF to express them [11] but none can express past indeterminacy.

Regularly, patients cannot provide exact dates regarding important health events that happened sometimes more than 40 years ago. In the event of a fracture, the exact onset moment was known at some point in time but might have been forgotten, leading to a missing date in the electronic medical record. A related but different problem occurs when a new diagnosis is made (e.g. diabetes). While we know that the disease is present at the time of the first diagnosis, for most diseases, the exact onset moment is unknown and it could be days, weeks, months or even years (e.g. hypertension) before. Currently, temporal data related to the examples above is often not included in the database. In the case where the application forces the user to enter a precise value, the clinicians will input an approximate date. This can lead to significant inconsistencies in the treatment of temporal operations during queries or the exclusion of relevant information.

2 UBHF concepts

UBHF is a conceptual framework that defines specifications and deterministic transformations based on fundamental relational and temporal concepts [7, 14]. This paper, presents a synthesis of these concepts, a more elaborated version can be found in [11].

2.1 Time model

UBHF uses a discrete time model based on points and intervals derived from the one defined by Allen in [1] and used in [7]. TIMPEPOINT and PERIOD will be used as a representative time point type and time interval type in the following sections. The re-

[1] Technical report can be found at http://griis.ca/surl/ubfh-dexa

tained timelines are transaction time (@T) and valid time (@V) as defined in the consensus glossary [8]. In the context of a HDW, the distinction between future indeterminacy and past indeterminacy allows easier processing of the retrospective and perspective data.

Future indeterminacy (indeterminate end) concerns facts with a known beginning and an unknown ending. In BCDM, this endpoint is represented by the constant « until changed » for a valid-time period and with « forever » for a transaction-time period. These constants are encoded in the database tables as the maximal value of TIMESTAMP type (9999-12-31 when granularity is one day). In DDLM and UBHF, this is represented with the function *ufn* « until further notice » and is only used in query expression (not for storage purpose).

Past indeterminacy (indeterminate begin) concerns facts with a known ending and an unknown beginning. This indeterminacy is overlooked by most models including DDLM and BCDM. In UBHF, this is represented with the function saw « since a while » and is only used in query expression (not for storage purpose).

2.2 Timelines

UBHF represents a timeline by an attribute (called timeline attribute). A timeline attribute can have different types and values defined as follows:

- A period timeline attribute (be): where the beginning and the end point values are known.
- A point timeline attribute with unknown end value (bx): where the beginning point is known and the end is unknown..
- A point timeline attribute with unknown beginning value (xe): where the beginning point is unknown and the end is known.

The table below defines the notation used. Note that, for transaction timelines the xe-type is not defined, as in a DBMS the beginning point of a transaction is always known.

Table 1. Timeline attribute definition and notation.

Notation	Definition	Timeline
@Vbe	Valid time period	Valid
@Vbx	Valid time begin point	
@Vxe	Valid time end point	
@Tbe	Transaction time period	Transaction
@Tbx	Transaction time point	

2.3 Attributes and relations

In a non-historical schema, we conventionally distinguish between key and non-key attributes. A non-historicized relation R is denoted R(K, A), where:

- $K = \{k_1,\ldots,k_{|K|}\}$ is the set of key attributes ($|K| \geq 1$). Without loss of generality, we consider that each relation contains only one key - although this key may have more than one attribute.
- $A = \{a_1,\ldots,a_{|A|}\}$ is the set of non-key attributes ($|A| \geq 0$).

A historicized relation R is denoted R'(K, B, C, D_V, D_T) with $A = B \cup C$ where:

- $B = \{b_1,\ldots,b_{|B|}\}$ is the set of non-key attributes ($|B| \geq 0$) associated with a valid timeline attribute (called historicized attributes); B is a subset of A.
- $C = \{c_1,\ldots,c_{|C|}\}$ is the set of non-key attributes ($|C| \geq 0$) **not** associated with a valid timeline attribute(called non-historicized attributes); C is a subset of A.
- $D_V = \{@V, b_1@V,\ldots, b_{|B|}@V\}$ is the set of valid timeline attributes, with the following notations @V is associated with K, and $b_i@V$ is associated with $b_i \in B$.

$D_T = \{@T, a_1@T,\ldots, b_{|A|}@T\}$ is the set of transaction timeline attribute where @T is associated with K, and $a_i@T$ is associated with $a_i \in A$.In the context of R', K and "key" do not refer to the same entity. K is the key set of the original relation R where the "key" of R' is the union of K with D_V and D_T.

Keeping history changes may introduce redundancy, contradiction, circumlocution and non-denseness when attributes in the same relation are modified independently. DDLM studied these problems in detail [7] (chap. 5 and 13) and proposed constraints to avoid them. See section 4 for a generalized form of constraint definitions.

3 Historicization process

Data modifications may introduce data inconsistency as described in previous sections. Data inconsistency is usually addressed by schema normalization and constraint specification. The normalization process uses lossless relational decomposition to split a relation into smaller relations ("relparts" for short). Two types of such decomposition are used hereafter: projection-join decomposition (PJ) and the restriction-union decomposition (RU) [11]. The historicization process consists of the following activities which are presented in the remaining of the present section:

- Schema annotation (guided).
- Historical schema construction (automated).
- Temporal constraints specification (automated).

3.1 Schema annotation

The aim of the schema annotation is to "parameterize" the algorithm of the schema transformation according to the domain needs. For each relation R(K, A) of the initial schema previously normalized in 5th normal form (5NF) : Split A into B (attributes with valid time) or C (attributes without).

3.2 Modelling

For each relation R(K, B, C) the following steps are required:

1. Decompose the relation R into 6NF using the PJ decomposition[2] PJ(R_K{K}, R_b$_1${K, b$_1$}, …, R_b$_n${K, b$_n$}, R_c$_1${K, c$_1$}, …, R_c$_m${K, c$_m$}).

To facilitate constraint definition, these (numerous) relparts are conceptually grouped into three types of groupings [11]:

- The *K-grouping* of a relation R is the set of all the K relparts. A K-relpart, denoted R_K, is a relation with K and the associated timeline attributes only.
- A *b$_i$-grouping* of a relation R is the set of all b$_i$-relparts. Collectively, the b$_i$-groupings are called *B-groupings*.
- A *c$_i$-grouping* of a relation R is the set of all c$_i$-relparts. Collectively, the c$_i$-groupings are called *C-groupings*.

For each relation R(K, B, C) the following steps are required to produce the historical representation (i.e. R!VT(K, B, C, D$_V$, D$_T$)).

2. For each relpart in K-grouping and B-groupings: Add the timeline attribute @V.
3. For each relpart in K-grouping and B-groupings: Decompose each resulting relparts, using RU decomposition over the @V timeline attribute to separate timeline between @Vxe, @Vbe and @Vbx types, renaming the relparts accordingly.
4. For all relparts (in K-, B- and C-groupings): Add two transaction time relparts: one with @Tbx and one with @Tbe.

In the following, without loss of generality, all relations are considered bitemporal. If a relation is a transaction time relation, only the last step is required.

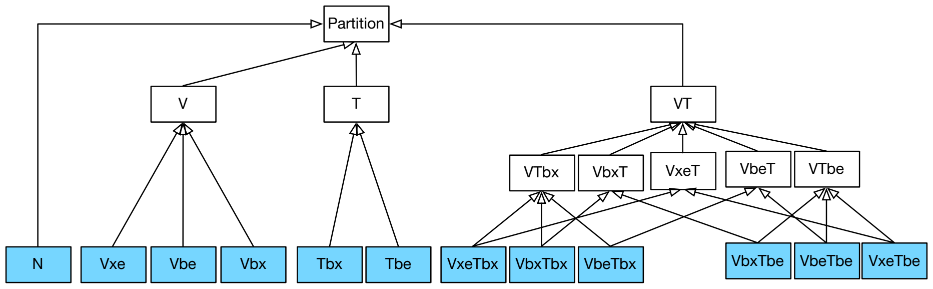

Figure 1. Hierarchy of all potential relational partitions

3.3 Relational partitions

Relparts are conceptually split into “partitions” to facilitate constraint definition and query expressiveness [11]. A partition is defined regarding the relation category and timeline category. Figure 1 shows the hierarchy of potential relational partitions that

[2] A PJ decomposition of a relation R is defined by its contributive projection subsets, each of them containing the common key and the union of them being equals to the attributes of R. Here, the subsets are precisely to the 6NF relparts of R (including the key relpart).

can be found in a historical schema depending on the decomposition. In UBFH, the structure and the constraints are defined over the leaf partitions (in blue).

In the following section a relational partition of R is denoted R@S where S is one of the categories illustrated in the previous figure (white and blue boxes). We may also refer to a specified grouping in a specific partition denoted R_g@S.

4 Constraints

This section presents constraint specifications that are automatically generated for a specified relation. Constraints are defined on V partitions because they can be modified by the user. No constraints are defined for T partitions because they cannot be modified by the user as they are managed by the DBMS using any of the following solutions: (a) as views when the values of the transaction timeline attribute are obtained by a function call to the DBMS journal as in DDLM or (b) as base relations (table) when the values of the transaction timeline attribute are set by the DBMS (as in SQL:2011 with SYSTEM TIME), or by triggers (as in BCDM [14]). In UBHF, all solutions are supported if they satisfy the transaction timeline semantic.

The following constraints must be defined regarding each relation of the initial schema. A constraint is defined with a unique identifier and a boolean expression.

4.1 Candidate keys

For each grouping R_g in partition S{N, Vxe, Vbx, Tbx, VxeTbx, VbxTbx}, the key constraint is the same as the initial relation. The constraint of R_g@S is:

```
RELATION R_g@S (K, B, C, D_V, D_T)  KEY {K};
```

For historicized relparts with be-type timeline, the key constraint must be applied to each time point in the period. For each grouping R_g in partition S{Vbe,VbeTbx}, add:

```
RELATION R_g@S (K, B, C, D_V, D_T)  USING(@Vbe): KEY {K, @Vbe};
```

For each grouping R_g in partition S{Tbe, VbxTbe, VxeTbe}, add:

```
RELATION R_g@S (K, B, C, D_V, D_T)  USING(@Tbe): KEY {K, @Tbe};
```

For each grouping R_g in partition S{VbeTbe}, add:

```
RELATION R_g@S (K, B, C, D_V, D_T)  USING(@Vbe, @Tbe): KEY {K, @Vbe, @Tbe};
```

4.2 Temporal denseness

The temporal denseness is defined over all relparts of an initial relation to ensure the history completeness. Each initial relation R has been decomposed in one K-grouping and some B- and C-groupings by historicization. Those groupings must stay consistent, i.e.: each key defined in the K-relpart must also be defined in the same period in one of its related present or missing relpart of B- and C-groupings. This constraint is inspired by the requirements 3 and 6 in [7] and by the equality dependency defined in [6].

First, a shorthand operator, **gSpace**, defined in [11], is extended to cover Vxe partition. The operator extracts the history (if any) of a specified grouping by calculating the union of all relparts of a grouping R_g with respect to the applicable partition[3] (@N or @V):

```
OPERATOR gSpace(R_g GROUPING) RETURNS RELATION;
  IF R_g is a C-grouping THEN
    R_g@N
  ELSE // R_g is a K-grouping or a B-grouping
    WITH (
      r_xe := (EXTEND R_g@Vxe : {@V:=[saw:@Vxe]}) {ALL BUT @Vxe}),
      r_bx := (EXTEND R_g@Vbx : {@V:=[@Vbx:ufn}) {ALL BUT @Vbx}),
      r_be := R_g@Vbe RENAME {@Vbe AS @V}
    ): USING (@V): r_xe UNION r_bx UNION r_be
  END IF
END OPERATOR
```

The temporal denseness may now be expressed using the two following rules:

For each b_i-grouping, the constraint is defined as follows:

```
CONSTRAINT R_bi_denseness
 USING(@V): gSpace(R_K) = gSpace(R_bi){K,@V}
```

For each c_i-grouping, the constraint is defined as follows:

```
CONSTRAINT R_ci_denseness
  gSpace(R_K){K} = gSpace(R_ci){K}
```

4.3 Foreign keys

A foreign key is evaluated regarding related attributes in different relparts. In a historicized relation, the associated timeline attributes must be considered to guarantee that their related values are asserted at the same time (at each moment of the unpacked relation). On the one hand, the foreign key in the initial schema must be maintained. Let Rs{X}→Rd be a foreign key in the initial schema where Rs is the source relation with X being any subset of attributes of Rs equivalent to the key (K) of Rd, the destination relation. The constraint guarantees temporal referential consistency between groupings by verifying that the projection of Rs on X is included in the projection of Rd on K (with suitable renaming). Using gSpace, another shorthand operator, **gUnpack**, is defined, extracting the unpacked history of an attribute x of a specified relation R:

```
OPERATOR gUnpack(R RelationName, x AttributeName) RETURNS RELATION;
  IF (x in K of R) THEN    UNPACK (@V): (gSpace(R_K){x,@V})
  ELSIF (x in B of R) THEN UNPACK (@V): (gSpace(R_x){x,@V})
  ELSE                     UNPACK (@V): gSpace(R_x){x} JOIN gSpace(R_K)
  END IF
END OPERATOR
```

Each foreign key Rs{X}→Rd induce the following constraint:

```
CONSTRAINT Rs_Rd_@V_fk
  gUnpack(Rs, x1) RENAME {x1 AS k1} ⊆ gUnpack(Rd_K, k1)
  AND … AND
  gUnpack(Rs, xn) RENAME {xn AS kn} ⊆ gUnpack(Rd_K, kn);
```

[3] gSpace is defined to deal with @T and @VT partitions. In this paper only @N and @V are represented.

4.4 Key history uniqueness

The key history uniqueness is defined over the value of the timeline associated with K. It ensures non-redundancy and non-contradiction of the key attributes values over time. In other words, it ensures consistency of the history of a tuple by verifying that the same proposition is represented only once. This constraint is similar to requirement 1 in [7]. For the K-relpart, the following constraint must be applied:

```
CONSTRAINT R_K_history
  IS_EMPTY (R_K@Vbx JOIN R_K@Vbe WHERE @Vbx < POST(@Vbe)) AND
  IS_EMPTY (R_K@Vbe JOIN R_K@Vxe WHERE @Vxe > PRE(@Vbe)) AND
  IS_EMPTY (R_K@Vxe JOIN R_K@Vbx WHERE @Vxe > ~~PRE~~(@Vbx));
```

4.5 Attribute history uniqueness

The attribute history uniqueness is defined over the value of the timeline attribute associated with a b_i attribute. It ensures non-redundancy and non-contradiction of b_i values over time. This constraint is similar to requirement 4 in [7]. For each grouping R_g in {R_b_1, …, R_b_n}, the following constraint must be applied:

```
CONSTRAINT R_g_uniqueness
  IS_EMPTY (R_g@Vbx {K, @Vbx} JOIN R_g@Vbe {K, @Vbe}
    WHERE @Vbx < POST(@Vbe)) AND
  IS_EMPTY (R_g@Vbe {K, @Vbe} JOIN R_g@Vxe {K, @Vxe}
    WHERE @Vxe > PRE(@Vbe)) AND
  IS_EMPTY (R_g@Vxe {K, @Vxe} JOIN R_g@Vbx {K, @Vbx}
    WHERE @Vxe > ~~PRE~~(@Vbx));
```

4.6 Key history non-circumlocution

The key history non-circumlocution is defined over the value of the timeline attribute associated with K. It ensures non-circumlocution of the key attributes values over time. This constraint is similar to requirement 2 in [7]. For the K-relpart, the following constraint must be applied:

```
CONSTRAINT R_K_circumlocution
  IS_EMPTY (R_K@Vbx JOIN R_K@Vbe WHERE @Vbx = POST(@Vbe)) AND
  IS_EMPTY (R_K@Vbe JOIN R_K@Vxe WHERE @Vxe = PRE(@Vbe)) AND
  IS_EMPTY (R_K@Vxe JOIN R_K@Vbx WHERE @Vxe = ~~PRE~~(@Vbx));
```

The two key history constraints (uniqueness and non-circumlocution) may be combined in one.

```
CONSTRAINT R_K_key_history
  IS_EMPTY (R_K@Vbx JOIN R_K@Vbe WHERE @Vbx <= POST(@Vbe)) AND
  IS_EMPTY (R_K@Vbe JOIN R_K@Vxe WHERE @Vxe >= PRE(@Vbe)) AND
  IS_EMPTY (R_K@Vxe JOIN R_K@Vbx WHERE @Vxe >= ~~PRE~~(@Vbx));
```

4.7 Attribute history non-circumlocution

The attribute history non-circumlocution is defined over the value of the timeline attribute associated to a b_i attribute. It ensures non-circumlocution of b_i values over time. This constraint is similar to requirement 5 in [7]. For each present value relpart R_b_i, the following constraint must be applied:

```
CONSTRAINT R_bi_circumlocution
  IS_EMPTY (R_bi@Vbx {K, bi, @Vbx} JOIN R_bi@Vbe {K, bi, @Vbe}
    WHERE @Vbx = POST(@Vbe)) AND
  IS_EMPTY (R_bi@Vbe {K, bi, @Vbe} JOIN R_bi@Vxe {K, bi, @Vxe}
    WHERE @Vxe = PRE(@Vbe)) AND
  IS_EMPTY (R_bi@Vxe {K, bi, @Vxe} JOIN R_bi@Vbx {K, bi, @Vbx}
    WHERE @Vxe = PRE(@Vbx));
```

5 Conclusion

Most proposed DW design methods define transformation rules "by-example" and must largely be adapted and applied manually. More specifically, regarding past indeterminacy, some DW design methods propose "ideas" but none presents a completely integrated deterministic process. UBHF defines (a) relation, attribute and timeline categorization to provide unique semantic; (b) a unified temporal structure and general constraints to be independent of the domain and context (yet providing formal definition and superior automation capabilities); (c) historicization processes with past indeterminacy ensuring traceability over the transformation steps without losing the initial conceptual view.

A model based on UBHF satisfies (1) data integrity with the definition of the described constraints (2), sound temporal schema design using relational concepts and well-defined temporal concepts (3) data schema evolution due to the normalization in 6NF (4) schema traceability and guided automation. This paper has extended UBHF which is now suitable to guided automated historicization of a database schema including past indeterminacy. This will enable improved modelling and query possibilities in many domains, especially in healthcare.

Despite all UBHF concepts, the DW schema still not "fully" historicized. In other words, adding the past indeterminacy induces a fourth timeline category, denoted eb, that represents events that occur "certainly" at some period (from e to b) but the beginning and the end point are not known. This situation may appear when two tuples having the same a_i values and two different timelines xe and bx that merge.

Future work is required to offer a fully applicative solution:

- To add the eb-type timeline.
- To model missing information.
- To implement a tool for designing historicized schema and translating it into standard SQL code or TutorialD code so existing DBMS may be used directly.
- To evaluate UBFH approach in real applications.
- To propose a generalized set of data modification operators (insert, delete, update).

References

1. Allen, J.F.: Maintaining Knowledge About Temporal Intervals. Commun ACM. 26, 11, 832–843 (1983).
2. Anselma, L., Piovesan, L., Terenziani, P.: A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy. J. Intell. Inf. Syst. 1–30 (2015).

3. Anselma, L., Terenziani, P., Snodgrass, R.T.: Valid-Time Indeterminacy in Temporal Relational Databases: Semantics and Representations. IEEE Trans. Knowl. Data Eng. 25, 12, 2880–2894 (2013).
4. Arora, S.: A comparative study on temporal database models: A survey. 2015 International Symposium on Advanced Computing and Communication (ISACC). pp. 161–167 (2015).
5. Darwen, H., Date, C.J.: The Third Manifesto. SIGMOD Rec. 24, 1, 39–49 (1995).
6. Date, C.J., Darwen, H.: Database Explorations: essays on the Third Manifesto and related topics. Trafford Publishing (2010).
7. Date, C.J., Darwen, H., Lorentzos, N.A.: Time and relational theory: temporal databases in the relational model and SQL. Morgan Kaufmann, Waltham, MA (2014).
8. Jensen, C.S., Dyreson, C.E., Bohlen, M., Cliford, J., Elmasri, R., Gadia, S.K., Grandi, F., Hayes, P., Jajodia, S., Kafer, W., Kline, N., Lorentzos, N., Mitsopoulos, Y., Montanari, A., Nonen, D., Peressi, E., Pernici, B., Roddick, J.F., Sarda, N.L., Scalas, M.R., Segev, A., Snodgrass, R.T., Soo, M.D., Tansel, A., Tiberio, P., Wiederhold, G.: The consensus glossary of temporal database concepts-February 1998 version. Proceedings of Seminar Temporal Databases: Research and Practice, 23-27 June 1997. pp. 367–405 Springer-Verlag (1998).
9. Jensen, C.S., Soo, M.D., Snodgrass, R.T.: Unifying Temporal Data Models via a Conceptual Model. Inf. Syst. 19, 513–547 (1993).
10. Khnaisser, C.: Méthode de construction d'entrepôt de données temporalisé pour un système informationnel de santé. Faculté des sciences, Université de Sherbrooke (2016).
11. Khnaisser, C., Lavoie, L., Burgun, A., Ethier, J.-F.: Unified Bitemporal Historicization Framework. Université de Sherbrooke (GRIIS), Sherbrooke, Québec, Canada (2017).
12. Khnaisser, C., Lavoie, L., Diab, H., Ethier, J.-F.: Data Warehouse Design Methods Review: Trends, Challenges and Future Directions for the Healthcare Domain. In: Morzy, T., Valduriez, P., and Bellatreche, L. (eds.) New Trends in Databases and Information Systems. pp. 76–87 Springer International Publishing (2015).
13. Rizzi, S., Abello, A., Lechtenborger, J., Trujillo, J.: Research in data warehouse modeling and design: Dead or alive? 9th ACM International Workshop on Data Warehousing and OLAP - DOLAP'06. pp. 3–10 Association for Computing Machinery, New York, NY, USA (2006).
14. Snodgrass, R.T.: Developing time-oriented database applications in SQL. Morgan Kaufmann Publishers, San Francisco, California (2000).