Schema Versioning and The Generalised Temporal Database System

Viviane Pereira Moreira
Nina Edelweiss

Universidade Federal do Rio Grande do Sul
Curso de Pós-graduação em Ciência da Computação
Av. Bento Gonçalves, 9500 – Bloco IV – Agronomia – Caixa Postal 15.064
CEP: 91501-970 – Porto Alegre – RS – Brasil
Fax: +55(51)336.5576
e-mail: [viviane, nina]@inf.ufrgs.br

Abstract

Raw data and database structures are evolving entities that require adequate support for past, present and even future versions. Temporal databases supporting schema versioning were developed with the aim of satisfying this requirement. Schema versioning allows the viewing of all extensional data, both retrospectively and prospectively, through user definable versions.

This paper considers a generalised temporal database system that keeps both the evolution of data and the evolution of the conceptual schema through the integration of temporal databases concepts and schema versioning mechanisms. The support for schema versioning raises two complex subjects: the storage of the several schema versions and their associate data, and the processing of queries involving more than one schema version. The main goal of this paper is to analyse the second aspect in order to propose a strategy to answer multi-schema queries.

Keywords: temporal databases, schema evolution, schema versioning

1 INTRODUCTION

Database systems are used to store information from the real world. Traditional database systems keep only the current state of data – every time an update is performed the past state is lost. The finding that data evolve with time led to the creation of temporal database systems. Temporal databases keep the evolution of data by using timestamps associated to him. In [Tansel93] four categories of temporal databases are identified:

- ***Snapshot Databases:*** these are the conventional database systems, which do not include any mechanism to deal with temporal information in an implicit way. In these databases every time an update is made, the past state is lost and, only the last value recorded is available for queries.
- ***Transaction Time Databases:*** these databases, also called *rollback*, uses the transaction time of data as timestamp. The transaction time of a fact is the time when the fact is current in the database. This timestamp is provided by the database management system.
- ***Valid Time Databases:*** these databases use the valid time of data as timestamp. The valid time of a fact is the time when the fact is true in the modelled reality. The user must inform this timestamp.
- ***Bitemporal Databases:*** these databases use both transaction and valid time as timestamps.

In [DeCastro95] another kind of temporal database is identified: the *multitemporal*. In a multitemporal database, relations of different temporal formats coexist.

In all temporal databases presented above, schema versioning was investigated only through the extension, allowing the evolution of data but keeping this data associated to a static schema. However, due to several reasons, the database structure may also change, fact that led to the development of schema evolution/versioning mechanisms.

There are three levels of support for schema changes in databases: modification, evolution and versioning. These terms are often confused. In [Roddick94] there are some definitions on which most recent research is based. These definitions were included in a glossary of temporal databases concepts [Jensen94, 98].

- ***Schema Modification:*** happens if the database system allows the modification of the schema definition of a populated database.
- ***Schema Evolution:*** happens if the database system allows the modification on the schema definition without loss of existing information.
- ***Schema Versioning:*** happens is the database system allows the querying of all data, both retrospectively and prospectively, through user definable versions.

The incorporation of schema evolution/versioning concepts in temporal database systems came as a logical extension of the work previously done on extensional data. The first model to support the temporal dimension at intensional level was the *Grammatical Database Model* [Laine79]. Years later, Martin [Martin87] approaches the problems caused by the maintenance of various schema versions; Clifford [Clifford87] proposes the HRDM (*historical relational data model)*; McKenzie [McKenzie90] discusses an extension to relational algebra to support the temporal dimension and schema evolution; Ariav [Ariav91] deals with some aspects related to schema evolution, presenting TODM - *Temporally Oriented Data Model*; Angelakis [Angelkis94] incorporates schema versioning to ERT - *Entity Relationship Time* [Theodoulidis91]; equally, Goralwalla [Goralwalla97] includes schema versioning to TIGUKAT [Peters94]. Among the most important work done on this field there are Roddick's research [Roddick, 92, 94, 95], De Castro and Scala's investigations [DeCastro95, 97, Scala93] and Edelweiss' proposal [Edelweiss95].

This paper considers the *generalised temporal database system* [Edelweiss95], in which both data and schema evolution are kept. In an environment with these features there are two complex subjects: the storage of the several schema versions and their associate data and the processing of queries which deal with more than one schema version (multi-schema queries)

The remainder of this paper is organised as follows. Section 2 introduces the generalised temporal database system. The main concepts of schema versioning in temporal databases are presented in Section 3. Section 4 shows how queries are processed and presents the strategy of answering multi-schema queries. Some conclusions and final considerations are resembled in Section 5.

2 GENERALISED TEMPORAL DATABASE SYSTEM

Non-temporal database systems have a static schema and a corresponding static database extension. Changes in the intensional or extensional data leave no track and it is impossible to query old data or old schema definitions. Conventional temporal database systems have a static schema and a corresponding temporal extension. The Generalised Temporal Database System [Edelweiss95] is a system whose members are temporal database systems, which have a temporal schema and a set of database systems. A temporal schema is structured as one or two (in case of bitemporal databases) sequences of static schemas (versions) whose behaviour is similar to a temporal database. Each data from the extension is associated to a static schema in the temporal schema.

A set of schema structure definitions and a set of constraints complementing them are called the *schema invariants* and characterise the schemas which are elements of the universe of acceptable schemas that may be constructed. This meta-schema is the representation of the requirements and constraints that all schemas constructed according to a conceptual model should satisfy. The picture in figure 1 shows the behaviour of a generalised temporal database system.

When a generalised temporal database system is created, there is an initial schema "A" and a corresponding initial database "a1". During the time that this schema is valid updates are made creating the sequence (a2, a3, a4). When a significant change is made to the schema a new version is created "B", and a corresponding database b1 is constructed. An adaptation is needed for the transaction from a4 to b1.

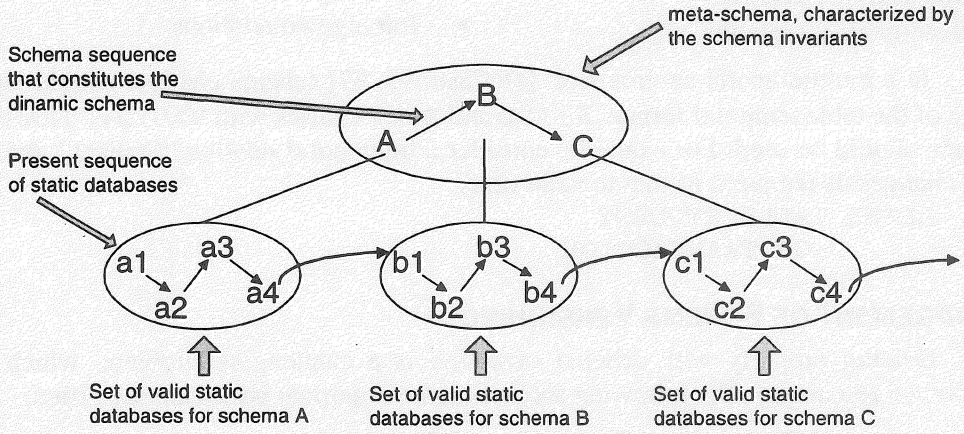

Figure 1 – Generalised Temporal Database System

2.1 Classification of Schema Versioning

The same considerations on temporal dimensions applied for data (see introduction) can be applied to schema level. Thus, schema versioning can be classified as snapshot (this is the conventional case, which will not be examined, since it does not provide support for schema versioning because it considers only one schema version, valid at any time), as transaction time, as valid time or as bitemporal database.

- **Transaction time Schema Versioning:** If transaction time schema versioning is supported, all the successive versions of the schema are available, each one timestamped with its corresponding transaction time. Most research consider only this kind of schema versioning, but it is more limited because the modifications concern only the current schema version and take effect in the moment they are defined lasting until a new version is defined, not allowing retro or proactive changes. In this case the meta-schema is managed as transaction time tables.
- **Valid time Schema Versioning:** In this type, each schema version is timestamped with its corresponding valid time. The new schema version becomes active when its validity is reached. Retro and proactive changes are allowed. The complication from transaction time is that more than one schema version may be affected by a single change, because all schema versions totally or partially overlapped by the validity of the change are affected. The meta-schema is managed as valid time tables, whose rows correspond to schema versions.
- **Bitemporal Schema Versioning:** In this type, each schema version is timestamped with both transaction and valid time. The transaction time tells when the modification was proposed and the valid time tells the period when the schema version is valid. A schema change can only concern the current and the overlapped bitemporal schema versions.

2.2 Modifications on The Schema Definition

The modification operations supported by the temporal generalised database are the same supported by the relational model, namely:

- **Modifications on Attributes**
  - Expanding the domain
  - Restricting the domain
  - Changing the domain
- **Modifications on Relations**
  - Creating a relation
  - Suspending a relation
  - Reactivating a relation
- **Modifications on attribute-relation assignment**
  - Adding an attribute to a relation
  - Suspending a non-key attribute
  - Reactivating an attribute
  - Promoting an attribute
  - Demoting an attribute
  - Splitting a relation
  - Joining two relations

In a multitemporal environment [DeCastro95, 97] schema changes should also include the change of the table temporal format. So `ADD` and `DROP` clauses with `VALID` or `TRANSACTION` specification should be used. For example, consider a bitemporal relation "Student", the following statement changes its temporal format to valid time.

```
ALTER TABLE STUDENT
      DROP TRANSACTION
```

3 Management of Schema Versioning

Dealing properly with schema evolution is a challenging problem, which has been subject of much research. In the following sections some important issues are identified.

3.1 Strategies of Recording Extensional Data

Two different solutions for the recording of extensional data can be used [DeCastro97]:

- ***single-pool solution:*** the data corresponding to all schema versions are maintained into a single repository according to a global schema (completed schema) which includes all attributes introduced by the successive schema changes;
- ***multi-pool solution:*** distinct data repositories (pools) are maintained for distinct schema versions. Each data pool is formatted according to its corresponding schema version. When a new repository is initialised the present information is copied from the old pool according to the changes on the schema. current

3.2 Interaction between Schema and Data Versioning

When extensional and intensional data are versioned along same temporal dimensions another distinction can be made:

- ***synchronous management:*** the temporal pertinence of the schema version must include the temporal pertinence of the corresponding data along the common temporal dimensions, so data are stored, updated and retrieved through the schema version having the same temporal pertinence.
- ***asynchronous management:*** the temporal pertinence of a schema version and the temporal pertinence of the corresponding data are independent, so data can be updated and retrieved through any schema version.

So the interaction between intensional and extensional data:

- is always *asynchronous* along orthogonal dimensions;
- is always *synchronous* along transaction time and
- can be *synchronous* or *asynchronous along valid time.*

3.3 Completed Schema and Null Values

The completed schema [Roddick94] is a schema with relations containing the union of attributes that have ever been defined for them. When the single-pool solution is used, extensional data are recorded according to their completed schema.

Example: consider the following history for a Student valid time relation:

→ Version 1: [01/01/1996 – 12/31/1996] Student (Register Name, Address, Phone)
→ Version 2: [01/01/1997 – 12/31/1997] Student (Register, Name, Phone)
→ Version 3: [01/01/1998 - ∞] Student (Register, Name, Phone, Course)

The completed schema for this relation is: Student (Register,Name,Address,Phone,Course).

When the concept of completed schema is used, the attributes (or relations) undefined at any schema version should be replaced by null values. Roddick [Roddick94] proposes a 7-valued null logic. For example, for attributes not currently defined their value depend on the reason behind their non-existence. The interpretation of these values is given in table 1.

	Attribute is defined		Attribute is not defined	
	Value is Known	Value is Unknown	Value is Known	Value is Unknown
Attribute is Applicable	value	ω_1	ω_4	ω_5
Attribute is Inapplicable		ω_2		ω_6
Applicability is Unknown	ω_3		ω_7	

Table 1 - Roddick's Null Values

3.4 Schema Valid Time and Schema Transaction Time

The support for schema versioning requires the introduction of two new timestamps: schema valid time (indicating the validity of the schema version in the reality) and schema transaction time (indicating the temporal pertinence of the schema version in the database system). These timestamps should be connected to each data to inform to which schema version the data belong.

3.5 Design Alternatives

The types of schema and data versioning, the solutions for the storage of extensional data and the interaction between schema and data versioning, were analysed with the goal of identifying all possibilities for the formation of a generalised temporal database system. Based on this analysis, this paper identified 38 kinds of database systems.

It is important to reinforce the independence between schema and data versioning. The choice of the types of schema and data versioning depends on the application requirements of historical recording at intensional and extensional levels. The choice of the storage solution depends on the storage space, if space is limited then single-pool should be used. The choice of the type of interaction should consider the level of independence needed. Thus, the definition of the database system depends on the application's requirements.

Table 2 shows the alternatives for the construction of a generalised temporal database system. The first column specifies the kind of schema versioning, the second column lists the kind of data versioning, the third column tells which solution is used for the storage of extensional data and in the fourth column there is the interaction between schema and data versioning. There are some empty spaces in the fourth column, this happens because the type of interaction can only be determined if both intensional and extensional data are temporal and known.

SCHEMA VERSIONING	DATA VERSIONING	STORAGE	INTERACTION
1. Transaction Time	Snapshot	Single-pool	
2. Transaction Time	Snapshot	Multi-pool	
3. Transaction Time	Transaction Time	Single-pool	Synchronous
4. Transaction Time	Transaction Time	Multi-pool	Synchronous
5. Transaction Time	Valid Time	Single-pool	Asynchronous
6. Transaction Time	Valid Time	Multi-pool	Asynchronous
7. Transaction Time	Bitemporal	Single-pool	Synchronous/Asynchronous
8. Transaction Time	Bitemporal	Multi-pool	Synchronous/Asynchronous
9. Transaction Time	Multitemporal	Single-pool	
10. Transaction Time	Multitemporal	Multi-pool	
11. Valid Time	Snapshot	Single-pool	
12. Valid Time	Snapshot	Multi-pool	
13. Valid Time	Transaction Time	Single-pool	Asynchronous
14. Valid Time	Transaction Time	Multi-pool	Asynchronous
15. Valid Time	Valid Time	Single-pool	Synchronous
16. Valid Time	Valid Time	Single-pool	Asynchronous
17. Valid Time	Valid Time	Multi-pool	Synchronous
18. Valid Time	Valid Time	Multi-pool	Asynchronous
19. Valid Time	Bitemporal	Single-pool	Asynchronous/Synchronous
20. Valid Time	Bitemporal	Single-pool	Asynchronous/Asynchronous
21. Valid Time	Bitemporal	Multi-pool	Asynchronous/Synchronous
22. Valid Time	Bitemporal	Multi-pool	Asynchronous/Asynchronous
23. Valid Time	Multitemporal	Single-pool	
24. Valid Time	Multitemporal	Multi-pool	
25. Bitemporal	Snapshot	Single-pool	
26. Bitemporal	Snapshot	Multi-pool	
27. Bitemporal	Transaction Time	Single-pool	Synchronous
28. Bitemporal	Transaction Time	Multi-pool	Synchronous
29. Bitemporal	Valid Time	Single-pool	Synchronous
30. Bitemporal	Valid Time	Single-pool	Asynchronous
31. Bitemporal	Valid Time	Multi-pool	Synchronous
32. Bitemporal	Valid Time	Multi-pool	Asynchronous
33. Bitemporal	Bitemporal	Single-pool	Synchronous/Synchronous
34. Bitemporal	Bitemporal	Single-pool	Synchronous/Asynchronous
35. Bitemporal	Bitemporal	Multi-pool	Synchronous/Synchronous
36. Bitemporal	Bitemporal	Multi-pool	Synchronous/Asynchronous
37. Bitemporal	Multitemporal	Single-pool	
38. Bitemporal	Multitemporal	Multi-pool	

Table 2 - Design Alternatives

4 QUERIES TO THE GENERALISED TEMPORAL DATABASE SYSTEM

With the fact that multiple schema versions coexist in the same database, the facility of querying these versions is desirable.

In the last years several temporal query languages have been proposed, but the one that has achieved most popularity is TSQL2 [Snodgrass95] – because it is a consensual extension of the SQL2 standard developed by a group composed by academic and specialists. Among the features of TSQL2 there is the support for transaction time schema versioning (according to Roddick's proposal). In [DeCastro95] it is extended to support valid time schema versioning as well.

In TSQL2 the specification of the schema version to be used in the query answer can be implicit (always the current schema) or explicit, as shown below:

Example 1:
```
SET SCHEMA DATE '1998-01-01'
SELECT * FROM STUDENT
WHERE Name = "Mary Brown"
```

The schema version is selected by "SET SCHEMA" statement. This query retrieves data using the schema active on 01/01/1998. Note that this statement only selects the schema version not affecting data selection.

Example 2:
```
SELECT * FROM STUDENT
WHERE SCHEMA (STUDENT) PRECEEDS DATE '1997-06-01'
```

In this case the schema version selector is added to the WHERE clause.

According to the extension proposed by De Castro [DeCastro95], TSQL2 also supports valid time and bitemporal schema versioning, so SET SCHEMA statement should be modified by the addition of a VALID and/or a TRANSACTION clause, as shown below:

Example 3:
```
SET SCHEMA VALID DATE '1998-01-01'
SELECT * FROM STUDENT
WHERE VALID(STUDENT) OVERLAPS DATE '1997-01-07'
```

This query retrieves data from table Student, which were valid at 01/01/1997 using the schema valid at 01/01/1998. There is a conflict between De Castro and Roddick's proposals about the processing of this kind of queries. According to De Castro, the constant "01/01/1998" would only be used to select the schema version if asynchronous management is employed, however, according to Roddick, extensional data can be restored through any schema version, so the constant "01/01/1998" would be used for any case. In the approach used by this paper, the constant "01/01/1998" would be used if single-pool is employed (because data are stored under the completed schema) or if asynchronous multi-pool is used (because data are updated in all pools formatted according to the corresponding schema version). When synchronous multi-pool is used it is not possible to query data through an old schema definition, because data is only updated in the current data pool. Thus, only in this case, the time constant 01/07/1997 will be used to select both data and schema along valid time.

Lets consider the most complex case, in which data and schema are bitemporal:

Example 4:
```
SET SCHEMA VALID DATE '1998-01-01'
     AND TRANSACTION DATE '1997-01-01'
SELECT * FROM STUDENT
WHERE VALID(STUDENT) OVERLAPS DATE '1995-01-07'
     AND TRANSACTION (STUDENT) OVERLAPS DATE "1996/01/01"
```

For this case, if synchronous management or single-pool are used, the constant "01/01/1998" will be used to select the schema version by valid time, the constant "01/01/1997" will be used to select the schema version by transaction time, the constant "01/07/1995" will be used to select the data by valid time and the constant "01/01/1996" will select the data by transaction time. If synchronous multi-pool is employed the temporal selection conditions on intensional and extensional data would conflict, so the conditions in the WHERE clause should prevail. Another problem could happen if the selection conditions for the schema do not refer to the same version, or if the selection conditions for the data do not refer to the same information, in both cases the query would be rejected.

Example 5: Consider a valid time relation "Student" belonging to a valid time schema, composed by the following three versions:

→ Version 1: [01/01/1996 – 12/31/1996] Student (Register Name, Address, Phone)
→ Version 2: [01/01/1997 – 12/31/1997] Student (Register, Name, Phone)
→ Version 3: [01/01/1998 - ∞] Student (Register, Name, Phone, Course)

The extensional data are recorded according to the single-pool solution, synchronous management and can be represented by the following table:

Register	Name	Address	Phone	Course	Valid Time	Schema Valid Time
1	John Hill	Flower's Road 23	3102225	*null*	01/01/1996 - ∞	01/01/1996 – 12/31/1996
2	Mary Brown	King's Street 99	3255878	*null*	06/15/1996 – 06/01/1998	01/01/1996 – 12/31/1996
3	Paul White	*null*	3252558	*null*	02/01/1997 - ∞	01/01/1997 – 12/31/1997
4	Jane Moore	*null*	3789967	Law	07/30/1998 - ∞	01/01/1998 - ∞
2	Mary Brown*	*null*	3250000	English	06/02/1998 - ∞	01/01/1998 - ∞

Thus, the following query:

```
SET SCHEMA VALID PERIOD '[1998-01-01 - 1998-10-10]'
SELECT * FROM STUDENT
WHERE VALID(STUDENT) OVERLAPS DATE '1997-07-07'
```

Can be answered as:

Register	Name	Phone	Course	Valid Time	Schema Valid Time
1	John Hill	3102225	*null*	01/01/1996 - ∞	01/01/1996 – 12/31/1996
2	Mary Brown	3255878	*null*	06/15/1996 - 06/01/1998	01/01/1996 – 12/31/1996
3	Paul White	3252558	*null*	02/01/1997 - ∞	01/01/1997 – 12/31/1997

The query selects the schema valid between 01/01/1998 and 10/10/1998 and data valid at 1997/07/07. Note that the attribute "Address", which was defined for the first version, is not present in the answer because it is not defined for the selected version. The rows "(Jane Moore, 3789967, Law)" and "(Mary Brown, 3250000, English)" are not present in the answer because their validity does not overlap the selected interval.

Example 6: Considering the same intensional and extensional data presented for Example 5, the following query is proposed:

```
SET SCHEMA VALID PERIOD '[1996-01-01 - 1997-12-12]'
SELECT Name, Address, Course FROM STUDENT
WHERE VALID(STUDENT) PRECEEDS DATE '1998-30-12'
```

In example 5 only one schema version qualifies for the query processing, however for this case two schema versions are defined within the interval selected by the query. This kind of queries are called multi-schema and cannot be answered in a simply way. When several versions qualify for the temporal selection conditions we have two alternatives: reject the query or answer it using all selected schemas (multi-schema answer).

4.1 The Proposed Strategy

The strategy proposed by this paper includes the following features:

- in order to support legacy applications, queries dealing with the current schema version should not need changes in the syntax of the DML when dealing with the current schema version. Thus, if the schema version is not specified, the current version should be used.
- in order to provide more semantics to the query answer, Roddick's null values (see table 1) should be used in substitution of attributes undefined at a given point in time.

* Note that there are two rows for Mary Brown. When the first one was inserted version 1 was valid. In 06/02/1997 her phone has changed, as we are dealing with a temporal database, we could not just replace the row. So another row is inserted, formatted according version 3, which was the current schema version.

- all queries should be answered following a predetermined format, shown below:

ATTRIBUTES	TIMESTAMPS		SCHEMA VERSION	
	Transaction	Valid	Transaction	Valid

- sometimes it may be desirable to force that the answer is given by means of the multi-schema answer, so the clause "`using all versions`" should be used.

So, using the strategy proposed, considering data presented for Example 5, the query answer would be:

ATTRIBUTES			TIMESTAMPS	SCHEMA VERSION
Name	**Address**	**Course**	**Valid Time**	**SchemaValid Time**
John Hill	Flower's Road 23	ω_5	01/01/1996 - ∞	01/01/1996 – 12/31/1996
Mary Brown	King's Street 99	ω_5	06/15/1996 – 06/01/1998	01/01/1996 – 12/31/1996
Paul White	ω_5	ω_5	02/01/1997 - ∞	01/01/1997 – 12/31/1997
Jane Moore	ω_5	Law	07/30/1998 - ∞	01/01/1998 - ∞
Mary Brown	ω_4	English	06/02/1998 - ∞	01/01/1998 - ∞

The null value ω_5 was used to inform that the attribute "Course" was not defined under version 1, its value is unknown and its applicable. The same happened to the attribute "Address" in version 2, it was nor defined, its value is unknown and it is applicable. The null value ω_4 was used to indicate that the attribute "Address" for Mary Brown was not defined in version 3, its value is known (the attribute Address was already defined for Mary, in version 1) and it is applicable.

Once the schema versions have been selected they will be used do access the underlying data. If the solution used for the storage of extensional data is single-pool then the application the proposed strategy is direct. However if multi-pool is used, a new schema composed by all attributes needed to answer the query must be constructed. The processing of queries in multi-pool involves several data-pools, and requires some filtering. From each data pool should only be recovered the extensional data stored under the schema version to which the pool belongs. Multi-schema queries may be performed on all 38 types of temporal databases supporting schema versioning.

Example 7: Consider a bitemporal relation "Employee" and a valid time relation "Dept", belonging to a valid time schema, composed by the following three versions:

→ Version 1: [01/01/1995 – 12/31/1996]Employee (Name, Salary, Phone, Function)
→ Version 2: [01/01/1997 – 12/31/1998]Employee (Name, Salary, Phone)
→ Version 3: [01/01/1999 – ∞] Employee (Name, Salary, Dept)
Dept (Code, Name)

The extensional data are recorded according to the multi-pool solution, asynchronous management and can be represented by Table 3.

Note that although the rows "(Lisa Fields, 850, 3257888)" and "(Jerry Simon, 1200, 3899988)" were inserted when version 2 was valid, they were also recorded under version 1 due to the asynchronous management. The same happened to the rows "(Tom Sands, 960,1)"and "(Bob Jones, 1700, 2)" that were inserted when version 3 was current and were also recorded under versions 1 and 2.

Based on the intensional and extensional data presented above, the query:

```
SET SCHEMA VALID PERIOD [06-01-1996 - 06-01-1999]
SELECT Name, Function, Phone FROM EMPLOYEE
WHERE VALID (EMPLOYEE) PRECEEDS DATE '02-01-1999'
```

can be answered as:

ATTRIBUTES			TIMESTAMPS		SCHEMA VERSION
Name	**Function**	**Phone**	**Transaction Time**	**Valid Time**	**Schema Valid Time**
James Lane	Salesman	3125588	01/01/1995 - ∞	01/01/1996 - ∞	01/01/1995 – 12/31/1996
Suzy Blair	Manager	3478999	02/25/1996 - ∞	03/01/1996 - ∞	01/01/1995 – 12/31/1996
Lisa Fields	ω_5	3257888	02/01/1997 - ∞	02/01/1997 - ∞	01/01/1997 – 12/31/1998
Jerry Simon	ω_5	3899988	07/30/1998 - ∞	08/01/1998 - ∞	01/01/1997 – 12/31/1998
Tom Sands	ω_5	ω_5	01/01/1999 - ∞	01/01/1999- ∞	01/01/1999 – ∞

Note that the row "(Bob Jones, 1700, 2)" does not qualify for the temporal selection clause, so it is not presented in the answer.

Version 1: [01/01/1995 – 12/31/1996]

Employee (Name, Salary, Function)

Name	**Salary**	**Phone**	**Function**	**Transaction Time**	**Valid Time**	**Schema Valid Time**
James Lane	1,000	3125588	Salesman	01/01/1995 - ∞	01/01/1996 - ∞	01/01/1995 – 12/31/1996
Suzy Blair	2,050	3478999	Manager	02/25/1996 - ∞	03/01/1996 - ∞	01/01/1995 – 12/31/1996
Lisa Fields	850	3257888	*null*	02/01/1997 - ∞	02/01/1997 - ∞	01/01/1997 – 12/31/1998
Jerry Simon	1,200	3899988	*null*	07/30/1998 - ∞	08/01/1998 - ∞	01/01/1997 – 12/31/1998
Tom Sands	960	*null*	*null*	01/01/1999 - ∞	01/01/1999- ∞	01/01/1999 – ∞
Bob Jones	1,700	*null*	*null*	02/25/1999 - ∞	03/01/1999- ∞	01/01/1999 – ∞

Version 2: [01/01/1997 – 12/31/1998]

Employee (Name, Salary, Phone)

Name	**Salary**	**Phone**	**Transaction Time**	**Valid Time**	**Schema Valid Time**
James Lane	1,000	3125588	01/01/1995 - ∞	01/01/1996 - ∞	01/01/1995 – 12/31/1996
Suzy Blair	2,050	3478999	02/25/1996 - ∞	03/01/1996 - ∞	01/01/1995 – 12/31/1996
Lisa Fields	850	3257888	02/01/1997 - ∞	02/01/1997 - ∞	01/01/1997 – 12/31/1998
Jerry Simon	1,200	3899988	07/30/1998 - ∞	08/01/1998 - ∞	01/01/1997 – 12/31/1998
Tom Sands	960	*null*	01/01/1999 - ∞	01/01/1999- ∞	01/01/1999 – ∞
Bob Jones	1,700	*null*	02/25/1999 - ∞	03/01/1999- ∞	01/01/1999 – ∞

Version 3: [01/01/1999 – ∞]

Employee (Name, Salary, Dept)

Name	**Salary**	**Dept**	**Transaction Time**	**Valid Time**	**Schema Valid Time**
James Lane	1,000	*null*	01/01/1995 - ∞	01/01/1996 - ∞	01/01/1995 – 12/31/1996
Suzy Blair	2,050	*null*	02/25/1996 - ∞	03/01/1996 - ∞	01/01/1995 – 12/31/1996
Lisa Fields	850	*null*	02/01/1997 - ∞	02/01/1997 - ∞	01/01/1997 – 12/31/1998
Jerry Simon	1,200	*null*	07/30/1998 - ∞	08/01/1998 - ∞	01/01/1997 – 12/31/1998
Tom Sands	960	1	01/01/1999 - ∞	01/01/1999- ∞	01/01/1999 – ∞
Bob Jones	1,700	2	02/25/1999 - ∞	03/01/1999- ∞	01/01/1999 – ∞

Dept (Code, Name)

Code	**Name**	**Valid Time**
1	Sales	01/01/1997 - ∞
2	Accountancy	01/01/1997 - ∞
3	Administration	01/01/1997 - ∞

Table 3 – Intensional and extensional data for Employee and Dept

Example 8: Considering the intensional and extensional data presented in Table 3, the query:

```
SELECT * FROM EMPLOYEE, DEPT
WHERE EMPLOYEE.Dept = DEPT.Code AND (TRANSACTION) EMPLOYEE
OVERLAPS DATE '1997-01-01' USING ALL VERSIONS
```

Can be answered as:

ATTRIBUTES			TIMESTAMPS		ATTRIBUTES		TIMESTAMPS	SCHEMA VERSION
Name	**Salary**	**Function**	**Transaction Time**	**Valid Time**	**Code**	**Name**	**Valid Time**	**Schema Valid Time**
James Lane	1,000	Salesman	01/01/1995 - ∞	01/01/1996 - ∞	ω_5	ω_5	ω_5	01/01/1995 – 12/31/1996.
Suzy Blair	2,050	Manager	02/25/1996 - ∞	03/01/1996 - ∞	ω_5	ω_5	ω_5	01/01/1995 – 12/31/1996
Lisa Fields	850	ω_5	02/01/1997 - ∞	02/01/1997 - ∞	1	Sales	01/01/1997 - ∞	01/01/1997 – ∞
Jerry Simon	1,200	ω_5	07/30/1998 - ∞	08/01/1998 - ∞	2	Accountancy	01/01/1997 - ∞	01/01/1997 – ∞

Example 9: Considering the data presented in Table 3, the following query:

```
SET SCHEMA VALID '1996-01-01'
SELECT EMPLOYEE.NAME, SALARY, DEPT.NAME
FROM EMPLOYEE, DEPT
WHERE EMPLOYEE.DEPT = DEPT.CODE
```

Would be rejected because the relation "Dept" is not defined for the selected schema version.

5 CONCLUSIONS

In order to obtain a complete representation of a relevant part of the real world in a database system, it is necessary to keep not only all defined data values (past, present and future), but also all schema versions. Having this in mind we presented the generalised temporal database system which achieves this objective by integrating temporal databases concepts and schema versioning mechanisms. The generalised temporal database purpose is to manage schema evolution similarly to the management of data evolution already employed in temporal databases. The construction of a system with these characteristics raises many complex aspects such as the storage and the recovery of the several schema versions and its associated data.

This paper's main concern was to study the processing of queries that involve two or more schema versions and to propose a strategy of answering such queries. Our strategy can be used by any of the 38 types of temporal database that support schema versioning. It has predefined format in order to facilitate its understanding by a compiled application. The strategy proposed is compliant to TSQL2 principles.

REFERENCES

[Angelakis94] ANGELAKIS, Dimitrios. ERT-SQL/SE: Incorporating Schema Evolution in a Temporal Query Language. MSc Thesis – University of Manchester – UK, 1994

[Ariav91] ARIAV, Gad. Temporally Oriented Data Definitions: Managing Schema Evolution in Temporally Oriented Databases In: Data & Knowledge Engineering v. 6, n. 6,Oct. 1991, p.451-467

[Clifford87] CLIFFORD, J.; CROKER A. The Historical Relational Data Model (HRDM) and Algebra based on Lifespans. In: 3rd IEEE International Conference on Data Engineering, Proceedings... Los Angeles,CA – 1987. pp. 528-537.

[DeCastro95] De CASTRO, Cristina; GRANDI, Fabio; SCALAS, Maria R., On Schema Versioning in Temporal Databases. In: Recent Advances in Temporal Databases, CLIFFORD, J.; TUZHILIN, A. (Eds.) Great Britain: Springer, 1995. p.272-291

[DeCastro97] CASTRO, Cristina; GRANDI, Fabio; SCALAS, Maria R., Schema Versioning For Multitemporal Relational Databases. Information Systems vol. 22, no. 5, 1997, p.249-290.

[Edelweiss 95] EDELWEISS, N.; CASTILHO, J.M.V.; OLIVEIRA, J.Palazzo M.Temporal Aspects of Conceptual Schema Evolution. In: International Conference Of The Chilean Computer Science Society, 15. Nov. 1-3, 1995, Arica, Chile. Proceedings..., p.187-197.

[Goralwalla97] GORALWALLA, Iqbal A.; SZAFRON, Duane; ÖSZU M. Tamer; Managing Schema Evolution Using a Temporal Object Model In: 16th International Conference on Conceptual Modeling (ER'97). Proceedings... Nov. 1997.

[Jensen94] JENSEN, C.S. et al. A Consensus Glossary of Temporal Database Concepts. SIGMOD RECORD, v.23, n.1, p.53-63, Mar.1994.

[Jensen98] JENSEN, C.S. et al. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. Temporal Databases Research and Practice . O. Etzion, S. Jajodia and S. Sripada (eds.) Springer-Verlag. Berlin Heidelberg 1998. pp. 367-405.

[Laine79] LAINE, H.; MAANAVILJA, O. and PELTOLA, E. "Grammatical Database Model" Information Systems, v. 4, p. 257-267. 1979

[MaKenzie90] McKENZIE, Edwin & SNODGRASS, Richard. Schema Evolution and the Relational Algebra. Information Systems v.15, n.2, p.207-232. 1990

[Martin87] MARTIN, N.G.; NAVATHE, S.B., AHMED, R. Dealing With Temporal Schema Anomalies in History Databases. In: 13th International Conference on Very Large Databases, Proceedings... Sept 1-4, 1987, Brighton, England, p.177-184

[Peters94] PETERS, R.J. TIGUKAT: A Uniform Behavioural Objectbase Management System. PhD Thesis University of Alberta, 1994

[Roddick92] RODDICK, John F., Schema Evolution in Database Systems - An Annotated Bibliography. ACM SIGMOD Record vol. 21 no. 4, December, 1992.

[Roddick94] RODDICK, J.F., A Model for Temporal Inductive Inference and Schema Evolution in Relational Database Systems . Ph.D. Thesis, Department of Computer Science and Computer Engineering, La Trobe University, 1994.

[Roddick95] RODDICK, J. F. & SNODGRASS, R. T., Schema Versioning In: The TSQL2 Temporal Query Language, Kluwer Academic Publishers, Noewell-MA, 1995

[Scala93] SCALAS, Maria Rita, CAPPELLI, Alessandro; CASTRO Cristina De; A Model for Schema Evolution in Temporal Relational Databases. In: 7th Conference Computers in Design, Manufacturing and Production IEEE (COMPEURO'93), Proceedings ...Paris, Evry, France,1993.

[Snodgrass95] SNODGRASS, R.T., et al., The TSQL2 Temporal Query Language, Kluwer Academic Publishers, Noewell-MA, 1995.

[Tansel93] TANSEL, A.U.; CLIFFORD, J.; GADIA, S.; JAJODIA, S.; SEGEV, A.; SNODGRASS, R. Temporal Databases - Theory, Design and Implementation . Redwood City: Benjamin/Cummings, 1993. 633p

[Theodoulidis91] THEODOULIDIS, C., LOUCOPOULUS, P. and WANGLER, B. A Conceptual Modelling Formalism for Temporal Database Applications", Information Systems, v. 16, n. 4, p. 401-416, 1991.