TVQL - Temporal Versioned Query Language

Mirella Moura Moro
Aglaê Pereira Zaupa
Nina Edelweiss
Clesio Saraiva dos Santos

Instituto de Informática - UFRGS - BRAZIL

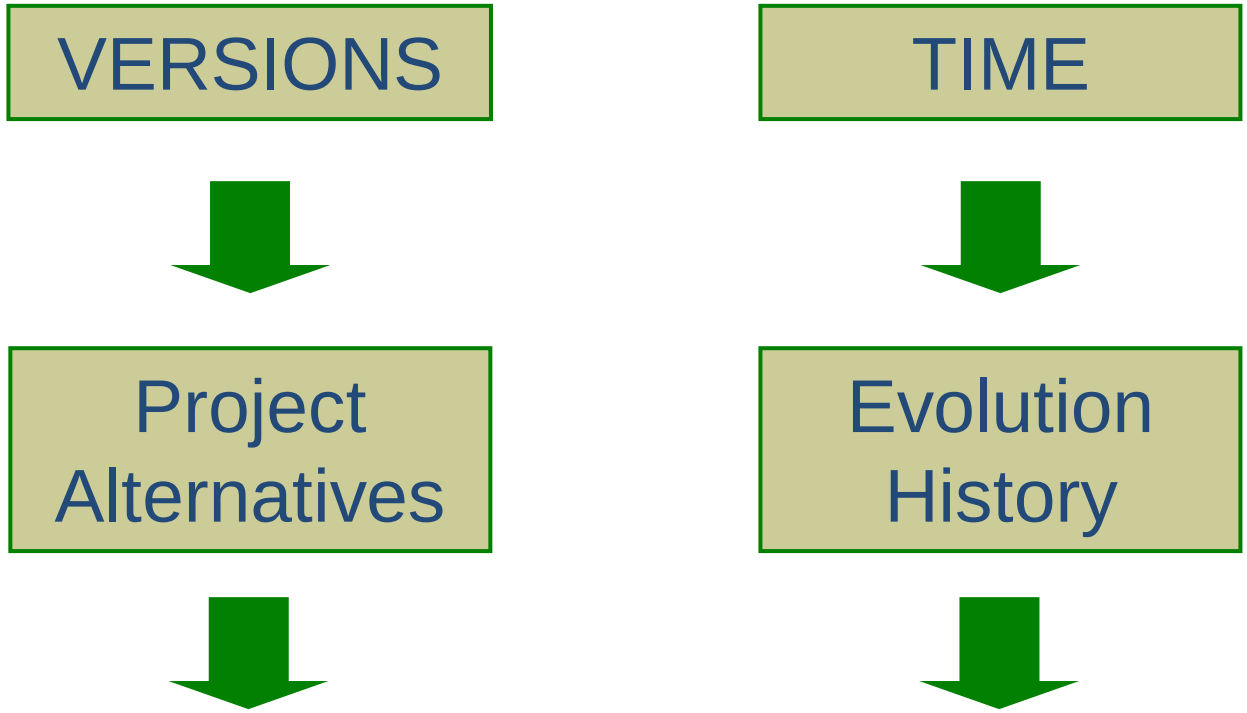
Contents

- Introduction
- Objective
- Example
- Temporal Versions Model
- Temporal Versioned Language
- Implementing TVQL
- Concluding Remarks

Introduction

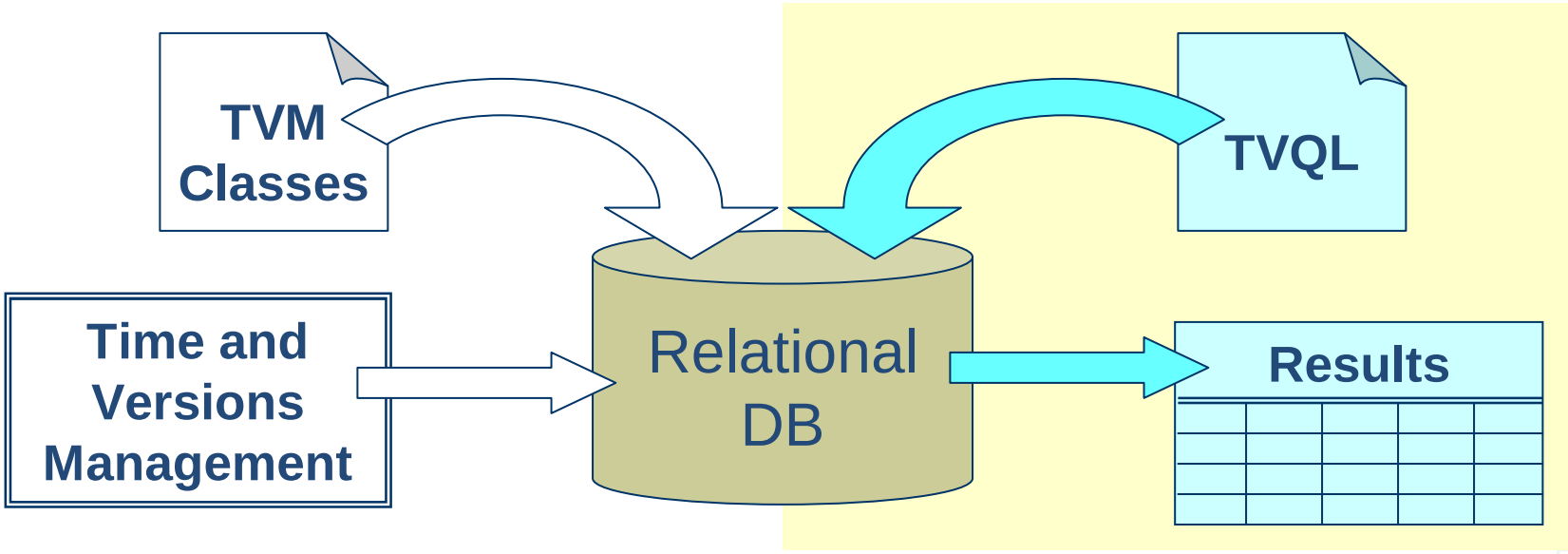
- Version: describes an object in a period of time or from a certain point of view
- Temporal Model: specifies both static and dynamic aspects of the application by associating temporal labels

Introduction

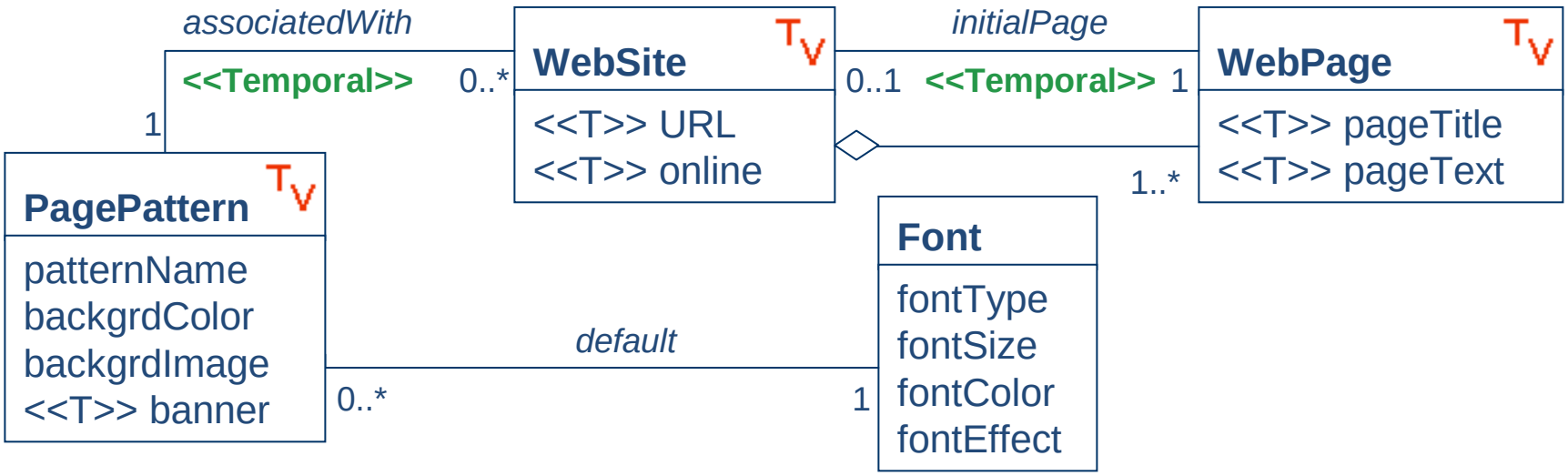


Objective

- TVM Project
  - implementing an integrated environment for class specification, object versioning, versions management, query, and visualization
  - Interface for temporal versioned class specification



Example

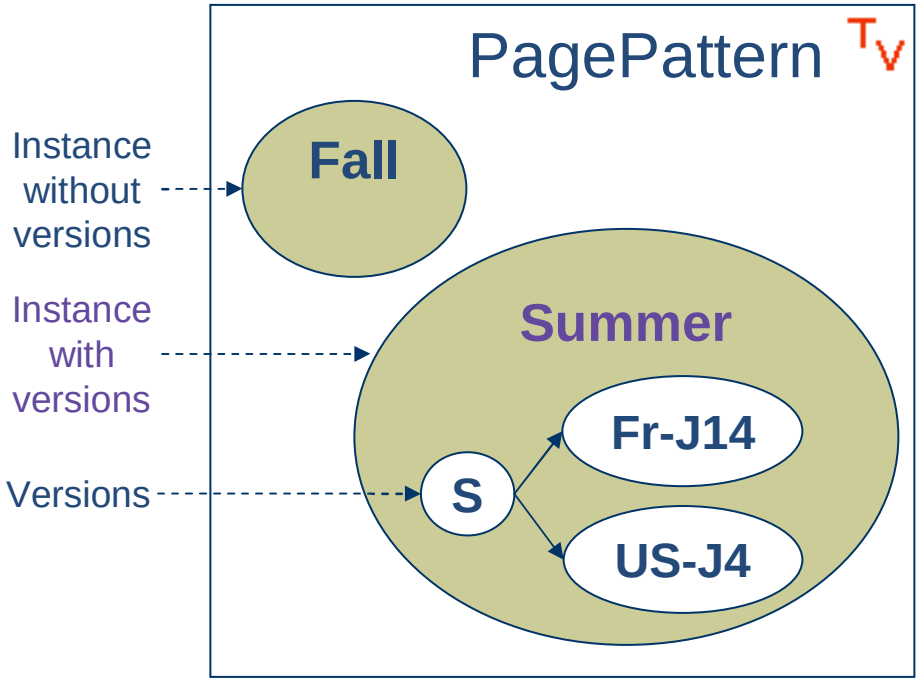

associatedWith
<<Temporal>>
0..*
1
WebSite
<<T>> URL
<<T>> online
initialPage
0..1
<<Temporal>>
1
WebPage
<<T>> pageTitle
<<T>> pageText
1..*
PagePattern
patternName
backgrdColor
backgrdImage
<<T>> banner
default
0..*
1
Font
fontType
fontSize
fontColor
fontEffect

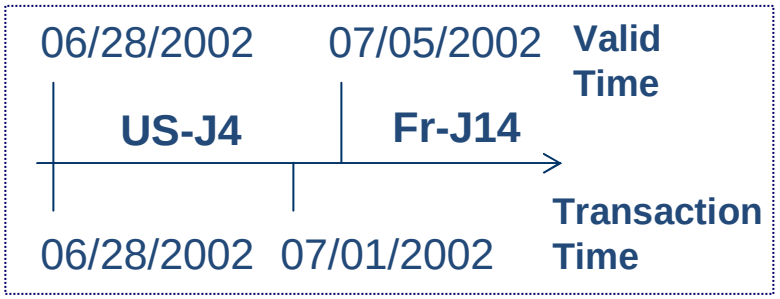
Temporal Versions Model

- TVM:
  - OO data model
  - adds the temporal dimension to the instance versioning level
- Design alternatives + data evolution
  - storage of different designed versions +
  - all updates in values of those attributes and relationships defined as temporal
- Usual classes + temporal versioned classes

Temporal Versions Model

- Instances of temporal versioned class PagePattern
- Temporal behavior of an attribute





TVQL Temporal Versioned Query Language

```
query ::=           SELECT [ EVER ] [ DISTINCT ]  targetC { , targetC }
    FROM identificC { , identificC }      [ WHERE [ EVER ] searchC ]
    [ GROUP BY groupC { , groupC }  [ HAVING logicalExpr ] ]
    [ ORDER BY orderC { , orderC }  [ setOp query ] ;
targetC ::=          ( * | propertyName | aggregationFunctions | preDefInterval
                     | preDefInstant )  [ AS identifier ]
identificC ::=       className [ .VERSIONS ] [ aliasName ]
searchC ::=          logicalExpr | tempExpr
groupC ::=           propertyName | preDefInterval | preDefInstant
logicalExpr ::=      AnyLogicalExpression
tempExpr ::=         logicalExpr | PRESENT ( logicalExpr )
orderC ::=           groupC [ ASC | DESC ]
setOp ::=                  UNION | INTERSECTION | DIFFERENCE
preDefInterval ::= [ propertyName. ] ( tInterval | vInterval )
preDefInstant ::=  [ propertyName. ] ( tiInstant | tfInstant | viInstant | vfInstant )
                     | [ className. ] ( iLifeTime | fLifeTime )
```

TVQL - Queries

Query	Meaning
SELECT property FROM class WHERE condition	Retrieves **current data** based on current data
SELECT property FROM class WHERE **EVER** condition	Retrieves **current data** based on all data history
SELECT property FROM class WHERE temporalVersionCondition	Retrieves **current data** based on temporal and versions condition
SELECT **EVER** property FROM class WHERE condition	Retrieves all data history based on all data history
SELECT **EVER** property FROM class WHERE **PRESENT**(condition)	Retrieves all data history based on current data
SELECT **EVER** property FROM class WHERE temporalVersionCondition	Retrieves all data history based on temporal and versions condition

TVQL - Predefined properties

- In order to retrieve and consider the temporal labels, TVQL specifies pre-defined temporal properties:
  - tInterval (transaction time interval)
  - vInterval (valid time interval)
  - tiInstant (initial transaction time)
  - tfInstant (final transaction time)
  - viInstant (initial valid time)
  - vfInstant (final valid time)
- There are also two pre-defined properties related to the object lifetime:
  - iLifetime (initial lifetime of the object)
  - fLifeTime (final lifetime of the object)

Implementing TVQL

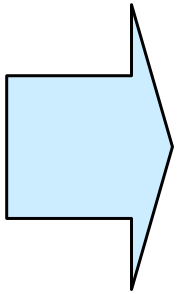
- Existing commercial DBMSs may be used for implementing TVM as long as a proper mapping from the model to the data model underlying the adopted DBMS is provided
- Mapping from TVQL to SQL based on a mapping from TVM to a relational database
- Mapping =
  representing the class model's hierarchy
  implementing the application classes
  mapping of the TVQL to SQL

Implementing TVQL
MAPPING TVM

- Standard classes: a table (main table) with the same name of the class
  - attributes => columns
  - primary key = tvoid

Standard class:

Font
fontType fontSize fontColor fontEffect



Relational Table:

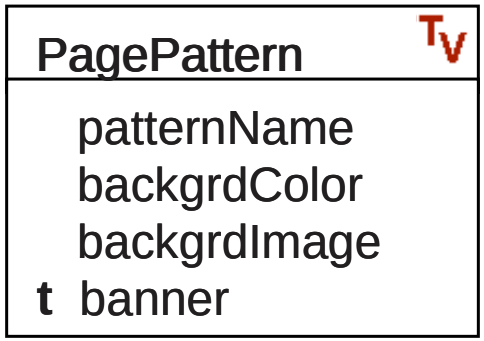
Font
tvoid fontType fontSize fontColor fontEffect

Main Table

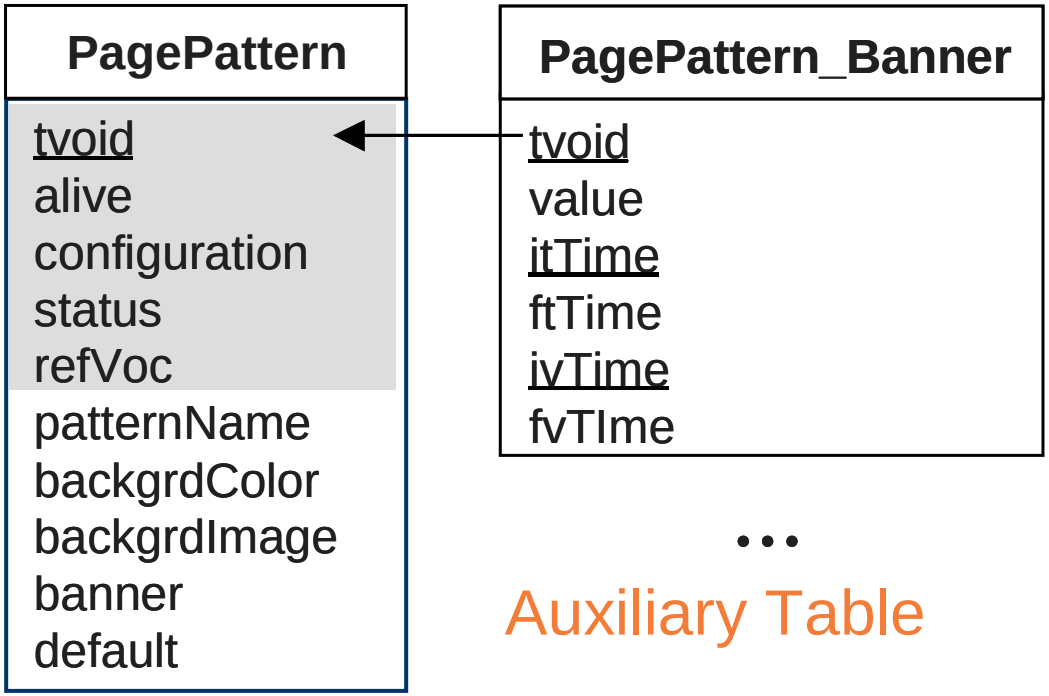
Implementing TVQL

MAPPING TVM - EXAMPLE

TVM class:



Relational Tables:



Auxiliary Table

Main Table

Implementing TVQL
MAPPING TVQL TO SQL

1. Mapping of classes, attributes and associations
2. Mapping of data restrictions

- There is no special rule for mapping the general TVQL to SQL (keyword distinct; count, sum, ...)
  - Classes and properties involved in these cases are mapped to the respective tables implemented

Implementing TVQL
MAPPING TVQL TO SQL

1. Mapping Classes, Attributes, Associations

- The first step is to define in which tables of the database the classes and properties are stored
- In *from* clause, classes are mapped to the main tables, and receive an alias (t1, t2, and so on)
- In *select* clause, attributes are mapped to the respective columns in the tables defined in *from*

CURRENT DATA:

TVQL	**SQL**
SELECT fontType, fontSize FROM Font;	SELECT t1.fontType, t1.fontSize FROM Font t1;

Implementing TVQL
MAPPING TVQL TO SQL

DATA HISTORY:

TVQL

```
SELECT EVER banner
FROM PagePattern.versions;
```

SQL

```
SELECT t1.value
FROM PagePattern_Banner t1
WHERE t1.ftTime = '12/31/9999';
```

- "List all banners in the history of all versions"
- If the keyword *ever* is used after *select*, the auxiliary tables are also considered
- As there is no temporal data restriction, this query considers the current database state, which has final transaction time with value null or '12/31/9999'

Implementing TVQL
MAPPING TVQL TO SQL

2. Mapping Data Restrictions: lot of possibilities

- "Retrieve the current banner from versions of page patterns that have banner name beginning with letter A in its history"

TVQL

```
SELECT banner
FROM PagePattern.versions
WHERE EVER banner LIKE  'A%';
```

SQL

```
SELECT t1.banner
FROM PagePattern t1, PagePattern_Banner t2
WHERE t1.tvoid = t2.tvoid and t2.ftTime = '12/31/9999'
        and t2.value like 'A%';
```

- Get current value
- Based on all history
- From current database state

Implementing TVQL
MAPPING TVQL TO SQL

- “Retrieve the current banner from versions of page patterns that have banner beginning with letter A as the database state on last December 31”

TVQL

```
SELECT banner
FROM PagePattern.versions
WHERE EVER banner LIKE  'A%'
and '12/31/2001'into tInterval;
```

SQL

```
SELECT t1.banner
FROM PagePattern t1, PagePattern_Banner t2
WHERE t1.tvoid = t2.tvoid  and t2.value like 'A%'
and t2.itTime <= '12/31/2001' and t2.ftTime >=  '12/31/2001';
```

- Get current value
- Based on history
- From past database state

Implementing TVQL
MAPPING TVQL TO SQL

- "Retrieve the history of banners that currently begin with letter A"

TVQL

```
SELECT EVER banner
FROM PagePattern.versions
WHERE PRESENT
        (banner LIKE  'A%');
```

SQL

```
SELECT t2.value
FROM PagePattern t1, PagePattern_Banner t2
WHERE t1.tvoid = t2.tvoid  and t2.ftTime = '12/31/9999'
        and t1.banner like 'A%';
```

- Get history
- Based on current values
- From current database state

Implementing TVQL
MAPPING TVQL TO SQL

- "Retrieve the url of online pages over the history"
- INTERSECT avoids a Cartesian Product (histories of url and online)

TVQL

```
SELECT EVER url
FROM WebPage.versions
WHERE online = true and
url.vInterval INTERSECT online.vInterval;
```

- Get history
- Based on history
- From current database state

SQL

```
SELECT t1.value
FROM WebSite_URL t1, WebSite_Online t2
WHERE t1.tvoid = t2.tvoid and t2.value = true and
t1.ftTime = '12/31/9999' and t2.ftTime = '12/31/9999' and
t1.ivTime<= t2.fvTime and t1.fvTime>= t2.ivTime;
```

Concluding Remarks

- This work presents
  - Query language (TVQL) for time-evolving systems with versions support, modeled using TVM
  - The mapping from TVQL to SQL
- TVQL
  - allows usual SQL statements
  - adds new properties to SQL clauses: retrieve past, current, and future data values, considering current and past database states
  - the same query statement may consider current and historic values, by combining keywords ever and present
  - all pre-defined properties are defined to hide storage details from the user

Concluding Remarks

- TVM Environment
- TVQL to SQL
- Under development
  - DML
  - Schema versioning and evolution

TVQL - Temporal Versioned Query Language

Contact author e-mail

nina@inf.ufrgs.br