

# Managing multiple time dimensions in clinical databases

Carlo Combi

Department of Computer Science, University of Verona

joint work with Angelo Montanari and Giuseppe Pozzi



# Outline

- 1 Introduction
- 2 A multidimensional temporal data model
  - Valid and Transaction Times
  - Event Time
  - Availability Time
  - Temporal relations
- 3 T4SQL: a multidimensional temporal query language
  - The Clause SEMANTICS
  - The Clause SELECT
  - The Clause FROM
  - The Clauses WHERE and WHEN
  - The Clauses GROUP BY and HAVING
- 4 Conclusions



# Motivation

- The temporal database community made considerable efforts in studying and proposing data models and query languages dealing with the well-known *valid and transaction times*.
- However, meaningful temporal aspects of facts in clinical information systems cannot be naturally modeled by means of valid and transaction times.



# Motivation

- Significant efforts have been undertaken to create database models that can capture the rich semantics of time needed for temporal querying and temporal reasoning in clinical decision support, mainly using the valid time dimension.
- Given the importance of querying time for clinical applications and the lack of standard temporal features in SQL, the specification of a standard, rich temporal relational query language for clinical data remains an open challenge.



## Goal of the work

- Refine the notion of *event time*.
- Introduce a further temporal dimension, called *availability time*.
- Propose a new temporal query language, named T4SQL, that extends, and is fully compatible with, SQL and deals with different temporal semantics (atemporal aka non-sequenced, current, sequenced, next) with respect to every temporal dimension.



# Valid and Transaction Times

**Valid time:** the *valid time* (VT) of a fact is the time when the fact is true in the modeled reality.

**Transaction time:** the *transaction time* (TT) of a fact is the time when the fact is current in the database and may be retrieved.



# Features

- Valid time is usually provided by database users, while transaction time is system-generated and supplied.
- Valid and transaction times are orthogonal dimensions: each of them can be independently recorded or not and has specific properties.
- Valid and transaction times can be related to each other in different ways. Jensen and Snodgrass propose several classifications (*specializations*) of (bi)temporal relations, based on the relationships between valid and transaction times of timestamped facts.



# A concrete situation

## Example

On August 10, 1998, the physician prescribes a bupivac-based therapy from 10:00 to 14:00. Data about the therapy is entered into the database at 9:00. Due to the unexpected evolution of the patient state, the bupivac infusion is stopped at 11:15 and replaced by a diazepam-based therapy from 11:25 to 14:00. The new facts are entered at 12:00. (Bupivac and diazepam are drugs commonly used in anesthesia.)

Drug	VT	TT
bupivac	[98Aug10;10:00, 98Aug10;14:00)	[98Aug10;9:00, 98Aug10;12:00)
bupivac	[98Aug10;10:00, 98Aug10;11:15)	[98Aug10;12:00, <i>uc</i> )
diazepam	[98Aug10;11:25, 98Aug10;14:00)	[98Aug10;12:00, <i>uc</i> )





# Event Time

**Event time:** the *event time* (ET) of a fact is the occurrence time of the real-world event that generates the fact.

## Example

- an on-time promotion event, that increases the rank of a physician, occurs on October 1, 1997, but its effects are recorded in the database on November 1 (delayed update);
- a retroactive promotion event, whose effects are immediately recorded in the database, occurs on November 1, but the increase in rank of the physician becomes valid since October 1 (retroactive update).



## Valid and Event Times

- *On-time events*: the validity interval starts at the occurrence time of the event (e.g., hospitalization starts immediately after the family doctor decision).
- *Retroactive events*: the validity interval starts before the occurrence time of the event (e.g., on December 21, 1997, the manager of the hospital decided an increase of 10% of the salary of the physicians of the Pathology Department, starting from December 1, 1997).
- *Proactive events*: the validity interval starts after the occurrence time of the event (e.g., on January 29, 1998, the family doctor decides the patient hospitalization on February 15, 1998).



## Transaction and Event Times

- *On-time update*: the transaction time coincides with the event time. This situation happens when data values are inserted in the database as soon as they are generated.
- *Delayed update*: the transaction time is greater than the event time. This is the case when data values are inserted some time after their generation.
- *Anticipated update*: the transaction time is less than the event time. This is the case when data values are entered into the database before the occurrence time of the event that generates them. Such a notion of anticipated update is useful to model hypothetical courses of events.



# One Event Time is not enough

- **IF** there are gaps in temporal validity (a relation modeling admissions to hospital, which only records information about inpatients),
- **OR** only incomplete information about the effects of an event is available (we know that an event that changes the rank of the physician occurred, but we do not know if it is a promotion or a downgrading event),
- **OR** the expected termination of a validity interval is (must be) revised, while its initiation remains unchanged (a prescribed therapy needs to be stopped due to an unexpected evolution of the patient state),
- **THEN** we need to distinguish between the occurrence times of the two events that respectively initiate and terminate the validity interval of the fact.



# A concrete situation

## Example

On August 10, 1998, at 8:00, the physician prescribes a bipuvac-based therapy from 10:00 to 14:00. Data about the therapy is entered into the database at 9:00. Due to the unexpected evolution of the patient state, at 11:00 the physician decides a change in the patient therapy. Accordingly, the bipuvac infusion is stopped at 11:15 and replaced by a diazepam-based therapy from 11:25 to 14:00. The new facts are entered at 12:00.



# Two (incorrect) solutions

## Example

Drug	VT	ET	TT
bipuvac	[98Aug10;10:00, 98Aug10;14:00)	98Aug10;8:00	[98Aug10;9:00, 98Aug10;12:00)
bipuvac	[98Aug10;10:00, 98Aug10;11:15)	98Aug10;8:00	[98Aug10;12:00, <i>uc</i> )
diazepam	[98Aug10;11:25, 98Aug10;14:00)	98Aug10;11:00	[98Aug10;12:00, <i>uc</i> )

Drug	VT	ET	TT
bipuvac	[98Aug10;10:00, 98Aug10;14:00)	98Aug10;8:00	[98Aug10;9:00, 98Aug10;12:00)
bipuvac	[98Aug10;10:00, 98Aug10;11:15)	98Aug10;11:00	[98Aug10;12:00, <i>uc</i> )
diazepam	[98Aug10;11:25, 98Aug10;14:00)	98Aug10;11:00	[98Aug10;12:00, <i>uc</i> )



# Event time revisited

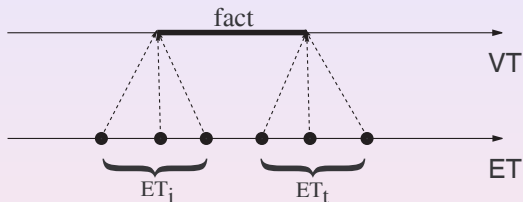
**Event time:** the *event time* of a fact is the occurrence time of a real-world event that either initiates or terminates the validity interval of the fact.

## Example

Drug	VT	$ET_i$	$ET_t$	TT
bipuvac	[98Aug10;10:00, 98Aug10;14:00)	98Aug10;8:00	98Aug10;8:00	[98Aug10;9:00, 98Aug10;12:00)
bipuvac	[98Aug10;10:00, 98Aug10;11:15)	98Aug10;8:00	98Aug10;11:00	[98Aug10;12:00, <i>uc</i> )
diazepam	[98Aug10;11:25, 98Aug10;14:00)	98Aug10;11:00	98Aug10;11:00	[98Aug10;12:00, <i>uc</i> )



## Some remarks



- This distinction between initiating and terminating event times comes from ideas underlying classical formalisms in the area of **reasoning about actions and change**.
- The choice of adding **event time(s) as a separate temporal dimension** has been extensively debated in temporal databases.





# Time and Information Systems

- By **information system** we mean the set of information flows of an organization and the human and computer-based resources that manage them.
- From such a point of view, we may need to model **the time at which** (someone/something within) **the information system becomes aware of a fact** as well as the time at which the fact is stored into the database. While the latter temporal aspect is captured by the transaction time, the former has never been explicitly modeled.



## A concrete situation

### Example

Due to a trauma that occurred on September 15, 1997, Mary suffered from a severe headache starting from October 1. On October 7, Mary was visited by a physician. On October 9, the physician administered her a suitable drug. The day after, the physician entered acquired information about Mary's medical history into the database. On October 15, the patient told the physician that her headache stopped on October 14; the physician entered this data into the database on the same day. Due to an insertion mistake (or to an imprecision in Mary's talk), the trauma has been registered as happened on September 5, 1997. Only on October 20, the mistake was discovered. The day after, the physician entered the correct data into the database.



# Availability Time

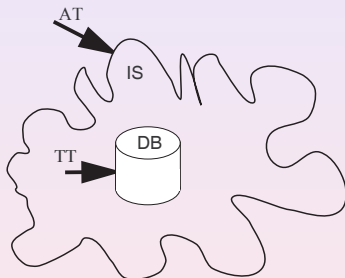
**Availability Time:** the *availability time* (AT) of a fact is the time interval during which the fact is known and believed correct by the information system.

## Example

symptom	VT	$ET_j$	$ET_t$	AT	TT
headache	[97Oct1, now)	97Sept5	null	[97Oct7, 97Oct15)	[97Oct10, 97Oct15)
headache	[97Oct1, 97Oct14)	97Sept5	97Oct9	[97Oct15, 97Oct20)	[97Oct15, 97Oct21)
headache	[97Oct1, 97Oct14)	97Sept15	97Oct9	[97Oct20, uc)	[97Oct21, uc)



## Transaction and Availability Times



- The availability time can be viewed as **the transaction time of the information system.**

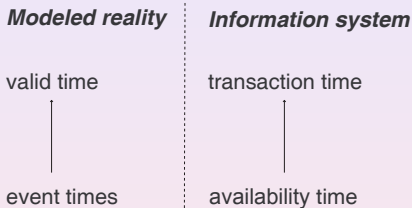


## Transaction and Availability Times

- If the information system outside the database is considered a database (in general, this is not the case; in our medical examples, for instance, the information system includes both databases and physicians), then the availability time of a fact is when the fact is inserted into (deleted from) the information system.
- This parallel between availability time and transaction time makes it immediately clear why the availability time has to be an interval.
- This point of view on the availability time resembles the notion of temporal generalization which allows several levels, and thus several transaction times, in a database.



# The complete picture



- Transaction time is append-only.
- Valid and event times, being related to times of the represented real world, can be located either in the past or in the future, and they can be modified freely.



## The complete picture (cont'd)

- The availability time is append-only in its nature, because facts previously known and believed correct by the information system cannot be changed.
- However, from the point of view of the database system, availability time would be really append-only only if there were no errors in data entry.
- Since we cannot exclude such a possibility, previous states of the information system, according to the availability time, can be revised by entering new data.
- Furthermore, even assuming data entry without errors, database and transaction times may “append” facts according to two different orders.



# The temporal data model

- The T4SQL data model is a straightforward extension of the relational model, where ***every relation is equipped with the four temporal dimensions.***
- The T4SQL temporal data model encompasses a single type of key constraints, namely, the *snapshot key* constraint (*key* for short).





# Snapshot key

Given a temporal relation  $R$ , defined on a set of (atemporal) attributes  $X$  and on the special attributes  $VT$ ,  $TT$ ,  $AT$ ,  $ET_i$ , and  $ET_t$ , and a set  $K \subseteq X$  of its atemporal attributes,  $K$  is a *snapshot key* for  $R$  if the following conditions hold:

- $\forall a \in K \forall t \in R (t[a] \neq \text{null})$
- $\forall t_1, t_2 \in R ((t_1 \neq t_2 \wedge t_1[VT, TT, AT] \cap t_2[VT, TT, AT] \neq \emptyset) \Rightarrow t_1[K] \neq t_2[K])$



# Example: a clinical database

## Pat\_Therapy

$P\_id$	$Therapy$	$Dosage$	$VT$	$ET_j$	$ET_t$	$AT$	$TT$
p2	Paracetamol	dose1	[2006-07-01, 2006-07-12)	2006-06-28	2006-06-28	[2006-06-28, 2006-09-06)	[2006-06-29, 2006-09-08)
p1	Streptomycin	dose3	[2006-05-08, 2006-11-15)	2006-05-05	2006-11-12	[2006-11-13, <i>uc</i> )	[2006-12-01, <i>uc</i> )
p2	Paracetamol	dose2	[2006-07-01, 2006-07-10)	2006-06-28	2006-07-10	[2006-09-07, <i>uc</i> )	[2006-09-08, <i>uc</i> )

## Pat\_Symptoms

$P\_id$	$Symptom$	$Sev\_Level$	$VT$	$ET_j$	$ET_t$	$AT$	$TT$
p2	Fever	1	[2006-06-25, <i>now</i> )	2006-06-23	null	[2006-06-27, 2006-07-11)	[2006-06-28, 2006-07-11)
p2	Fever	2	[2006-06-25, 2006-07-11)	2006-06-23	2006-06-28	[2006-07-12, <i>uc</i> )	[2006-07-15, <i>uc</i> )
p1	Fever and dry cough	4	[2006-05-01, 2006-10-21)	2006-03-23	2006-05-05	[2006-11-13, <i>uc</i> )	[2006-11-15, <i>uc</i> )



## Semantics

The main features of T4SQL include the following semantics:

- *current*, which considers only current tuples;
- *sequenced*, which corresponds the homonymous SQL3 semantics;
- *atemporal*, which is equivalent to the SQL3 non-sequenced one;
- *next*, which allows one to link consecutive states when evaluating a query.



# The syntax of T4SQL

```
SEMANTICS <sem> [ON] <dim> [[TIMESLICE] <ts_exp>]
      , <sem> [ON] <dim> [[TIMESLICE] <ts_exp>]
SELECT <sel_element_list>
      [WITH <w_exp> [AS] <dim> , <w_exp> [AS] <dim> ]
      [TGROUPING] [WEIGHTED]
FROM <tables>
WHERE <cond>
WHEN <t_cond>
GROUP BY <list_group_element>
HAVING <list_group_element> ::= <group_element>
      , <group_element>
<group_element> ::= <attribute> |
      <temp_attribute> USING <part_size>
<sem> ::= 'ATEMPORAL' | 'CURRENT' | 'SEQUENCED'
      | 'NEXT(number)'
<dim> ::= 'VALID' | 'TRANSACTION' | 'AVAILABILITY' |
      'INITIATING_ET' | 'TERMINATING_ET'
```



## Managing multiple semantics

### Example

We want to extract all the patients that suffered from “Fever” and the respective period during which the fact happened. In a relational DBMS where temporal dimensions are managed explicitly by the user, the query considering all the temporalities may look like the following (assuming that the DBMS is able to deal with periods):

```
SELECT      P_id, pp.valid
FROM        Pat_Symptom AS pp
WHERE       Symptom = 'Fever' AND
            pp.transaction OVERLAPS DATE 'uc'
```



## Managing multiple semantics

### Example

The obtained code is more readable, less complex, and possibly, with a reduced number of errors. The same query of above is expressed in T4SQL as follows:

```
SEMANTICS SEQUENCED ON VALID
SELECT    P_id
FROM      Pat_Symptom
WHERE     Symptom = 'Fever'
```



## Managing multiple semantics

The temporal dimension of VT is interpreted according to the use of the keyword `SEQUENCED`, while the temporal dimension of TT is automatically managed to consider only current tuples. If instead we want to consider all the patients that suffered from fever during the year 2005, the `TIMESLICE` token helps us and the T4SQL query is the following:

### Example

```
SEMANTICS SEQUENCED ON VALID TIMESLICE
          PERIOD '[2005-01-01 - 2005-12-31]'
```

```
SELECT   P_id
FROM     Pat_Symptom
WHERE    Symptom = 'Fever'
```



## A TEMPORAL Semantics

If the **A TEMPORAL** semantics is adopted, the corresponding attribute(s) is dealt with as atemporal (timeless), providing the user with the highest level of freedom in managing temporal dimensions, even though any support from the system is disabled.





## ATEMPORAL Semantics

### Example

If we want to retrieve all the patients where the symptom “Fever” has never been observed, the query does not require any temporal dimension, but requires to span over the entire temporal axis. The resulting T4SQL query is the following:

```
SEMANTICS ATEMPORAL ON VALID
SELECT    P_id
FROM      Pat_Symptom P
WHERE     P.P_id NOT IN (
    SEMANTICS ATEMPORAL ON VALID
    SELECT  P_id
    FROM    Pat_Symptom
    WHERE   Symptom = 'Fever' )
```



## CURRENT Semantics

The `CURRENT` semantics, applied to a temporal dimension  $d$ , considers only the tuples where the value associated to  $d$  includes the current date.

The `CURRENT` semantics may assume a different meaning according to the temporal data type associated to the considered temporal dimension:

- if the specified data type is a period, such as the VT, the TT or the AT, the query considers only the tuples satisfying the condition  $t(d) \text{ CONTAINS DATE 'uc'}$  ('now' for VT);
- if the specified data type is a date, as for the ET, the query considers all the tuples satisfying the condition  $t(d) = \text{DATE 'now'}$ .



## CURRENT Semantics

### Example

We want to retrieve all the patients from `Pat_Symptom`, as currently stored. The T4SQL query is:

```
SEMANTICS CURRENT ON TRANSACTION, ATEMPORAL ON VALID
SELECT      P_id
FROM        Pat_Symptom
```



## Example

The above query can be expressed by an ATEMPORAL semantics (the complexity of the query increases if the user has to explicitly manage all the temporal dimensions) as:

```
SEMANTICS ATEMPORAL ON TRANSACTION,  
                                     ATEMPORAL ON VALID  
  
SELECT      P_id  
FROM        Pat_Symptom AS pp  
WHERE       TRANSACTION(pp) CONTAINS DATE 'uc'
```

The above query can be rewritten as:

```
SEMANTICS ATEMPORAL ON TRANSACTION  
          TIMESLICE DATE 'uc', ATEMPORAL ON VALID  
  
SELECT      P_id  
FROM        Pat_Symptom AS pp
```



## SEQUENCED Semantics

The **SEQUENCED** semantics forces a *step by step* evaluation of the statement. A step by step evaluation of a given temporal dimension  $d$  considers all the tuples of the relations in the **FROM** clause where there exists a date  $i$  belonging to all the periods of the dimension  $d$ .



## SEQUENCED Semantics

### Example

If we want to retrieve the patients who suffered from fever, considering the information that the database in its current state believed correct on December 1, 2006, the query is:

```
SEMANTICS SEQUENCED ON VALID,  
  CURRENT ON TRANSACTION, SEQUENCED ON AVAILABLE  
  TIMESLICE PERIOD `[2006-12-01 - 2006-12-01]`  
SELECT    P_id  
FROM      Pat_Symptom  
WHERE     Symptom = `Fever`
```

The result is a temporal relation with one explicit attribute (*P\_id*) and two implicit attributes (VT and AT) managed by the system.



## SEQUENCED Semantics

### Example

In this case, too, the query can be translated to a different one with the **ATEMPORAL** semantics, as follows:

```
SEMANTICS ATEMPORAL ON VALID,  
ATEMPORAL ON TRANSACTION, ATEMPORAL ON AVAILABLE  
SELECT    P_id WITH VALID(pp) AS VALID,  
          AVAILABLE(pp) AS AVAILABLE  
FROM      Pat_Symptom AS pp  
WHERE     symptom = 'Fever' AND  
          TRANSACTION(pp) OVERLAPS DATE 'uc' AND  
          AVAILABLE(pp) OVERLAPS DATE '2006-12-01'
```



## NEXT Semantics

The **NEXT** semantics enables the user to retrieve information about the same object as observed in two subsequent dates over the order of the temporal dimension.

### Example

Starting from the `Pat_Therapy` relation, we want to retrieve for every patient the period elapsed between two subsequent administrations. The corresponding T4SQL query is:

```
SEMANTICS NEXT ON VALID
SELECT    P_id, Therapy,
          (BEGIN (VALID (NEXT (t))) - END (VALID (t))) DAY
FROM      Pat_Therapy AS t
```





## NEXT Semantics

### Example

We want to retrieve all the patients who received a therapy 'Paracetamol' moving from a dosage 'dose1' to a dosage 'dose2'. The query in the T4SQL query language is the following:

```
SEMANTICS NEXT(0) ON VALID
SELECT    P_id
FROM      Pat_Therapy AS t
WHERE     t.Therapy = 'Paracetamol' AND
          t.Dosage = 'dose1' AND
          NEXT(t).Dosage = 'dose2'
```



## Default Values

If no information or no temporal dimension is specified for the SEMANTICS clause, default values apply. Compatibility of T4SQL towards SQL92 drives the choices for default values.

### Example

From the `Pat_Symptom` relation, we want to retrieve patients who are suffering from “Fever”. The SQL92 query has no condition over temporal aspects and it is:

```
SELECT    P_id
FROM      Pat_Symptom
WHERE     Symptom = 'Fever'
```

The result is an atemporal relation considering only information in the database and valid at the current time.



## Default Values

### Example

The equivalent T4SQL code is:

```
SEMANTICS CURRENT ON VALID,  
           CURRENT ON TRANSACTION,  
           CURRENT ON AVAILABLE,  
           ATEMPORAL ON INITIATING_ET,  
           ATEMPORAL ON TERMINATING_ET  
  
SELECT    P_id  
FROM      Pat_Symptom  
WHERE     Symptom = 'Fever'
```



## Specifying temporal dimensions

T4SQL introduces the token `WITH` in the `SELECT` clause, to separate the specification of explicit attributes from that of implicit attributes. Any statement before the token `WITH` is evaluated according to the SQL92 semantics for projection: any statement following the token `WITH` computes the temporal dimension(s) to be included in the result relation.



# Specifying temporal dimensions

## Example

We want to retrieve the patients who suffered from symptom '*Fever*', received the drug '*Paracetamol*', and resolved their disease within 5 days from the beginning of the therapy. Every element in the result relation must come with the VT, which is included within the beginning of the symptom and the end of the therapy. The resulting T4SQL query is:

```
SEMANTICS SEQUENCED ON VALID
SELECT    P_id WITH PERIOD (BEGIN (VALID (pp)),
                END (VALID (pt))) AS VALID
FROM      Pat_Symptom AS pp, Pat_Therapy AS pt
WHERE     pp.Symptom = 'Fever' AND
          pt.Therapy = 'Paracetamol' AND
          pp.P_id = pt.P_id AND
          (END (VALID (pp)) - BEGIN (VALID (pt))) DAY
          < INTERVAL '5' DAY
```



## Joins and temporal joins

The `FROM` clause of SQL92 identifies the relations used to find the attributes and/or to perform join operations. In SQL92 several `JOIN` criteria can be defined: `INNER` join, which is the default value, `LEFT OUTER`, `RIGHT OUTER` and `FULL OUTER` joins. The token `ON` specifies the selection conditions of the `FROM` clause. T4SQL adopts the same `FROM` clause as SQL92. Additionally, besides the token `JOIN` of SQL92, we introduced also the token `TJOIN` (*temporal join*).



## temporal join

### Example

We want to select, according to the current state of the database, all the patients who showed any symptom during the assumption of a therapy that lasted for a period greater than 10 days. The T4SQL query is:

```
SEMANTICS SEQUENCED ON VALID
SELECT    P_id, Therapy, Symptom
FROM      Pat_Therapy AS pt TJOIN
          Pat_Symptom AS pp ON
          (pt.P_id = pp.P_id AND
           (END (VALID (pp)) - BEGIN (VALID (pp))) DAY
           > INTERVAL '10' DAY )
```



## Specifying select conditions

In SQL92, the `WHERE` clause evaluates selection predicates over tuples from the relations in the `FROM` clause. The `WHERE` clause of T4SQL extends the SQL92 clause possibly including some temporal conditions. As temporal conditions may turn out to be very complex, the user can optionally separate temporal conditions from atemporal conditions by using the `WHEN` clause.





## Specifying select conditions

### Example

Consider again the query of the previous example. It can be defined by using the clause `WHEN` as follows:

```
SEMANTICS SEQUENCED ON VALID
SELECT    P_id, Therapy, Symptom
FROM      Pat_Therapy AS pt, Pat_Symptom AS pp
WHERE     pt.P_id = pp.P_id
WHEN      (END (VALID (pp)) - BEGIN (VALID (pp))) DAY
          > INTERVAL '10' DAY
```



## Temporal grouping

In a temporal query language, the `GROUP BY` clause can be used to implement a *temporal grouping*, rather than a punctual grouping over atemporal attributes. The selection of the tuples to be grouped together (we have one group for every partitioning element) is performed according to the value of the considered temporal attribute(s) of the tuple, according to the periods associated with the partition, and according to the temporal comparison operator used in the temporal grouping.



## Temporal grouping

### Example

We want to perform a temporal grouping considering the VT and the initial event time. The resulting query is the following:

```
SELECT TGROUPING (VALID (...) AS V,  
                 INITIATING_ET (...) AS IE), ...  
FROM ...  
GROUP BY  
  VALID (...) USING ...,  
  INITIATING_ET (...) USING ...
```



## Temporal grouping

### Example

Let us assume that we want to retrieve for every year the average duration of prescribed therapies. The corresponding T4SQL query is the following:

```
SELECT      TGROUPING (VALID (t) AS period),  
            AVG (CAST (INTERVAL (VALID (t) DAY) )  
                AS INTEGER)  
FROM        Pat_Therapy AS t  
GROUP BY   VALID (t) USING YEAR
```



## Temporal grouping

### Example

We want to compute from the table `Pat_Symptom` the average level of weighted severity (`Sev_Level` is in a range 1÷10) for patients who suffered from fever and dry cough every month, returning only the tuples of patients where the average is greater than 3. The T4SQL query is:

```
SELECT      TGROUPING (VALID(t) AS period),  
            WEIGHTED AVG(Sev_Level)  
FROM        Pat_Symptom AS p  
WHERE       Symptom = 'Fever and dry cough'  
GROUP BY   VALID(p) USING MONTH  
HAVING     WEIGHTED AVG(Sev_Level) > 3
```



## Final remarks

- We proposed a new, fully symmetric **conceptual data model** with multiple temporal dimensions, that revises and extends existing temporal models.
- We first described a **refinement of the concept of event time**, that replaces the single event time by a pair of *initiating* and *terminating event times*.
- Then, we introduced **a new temporal dimension** (*availability time*), which captures the time interval during which a fact is known and believed correct by the information system.



## Final remarks

- The availability time can be exploited, for example, to analyze the **quality of decision making in hospital information systems** where data insertion is performed in batches and thus some delay is possible between data availability and data insertion.
- The new T4SQL query language over multidimensional temporal relations supports the temporal dimensions of *valid*, *transaction*, *availability*, and *event* times.
- T4SQL performs queries over temporal relations provided with some/all the supported temporal dimensions according to different semantics, ranging from the already known `CURRENT`, `ATEMPORAL`, and `SEQUENCED` semantics, to the new `NEXT` semantics.



## Future Work

- Apply the data model and query language to some real complex clinical domains.
- Merge the management of multiple temporal dimensions for clinical data with process-oriented medical records.





## References on temporal dimensions of data



Carlo Combi and Angelo Montanari.

Data models with multiple temporal dimensions:  
Completing the picture.

In K. R. Dittrich, A. Geppert, and M. C. Norrie, eds, *CAiSE*,  
volume 2068 of *LNCS*, pages 187–202. Springer, 2001.



Sharma Chakravarthy and Seung-Kyum Kim.

Resolution of time concepts in temporal databases.

*Inf. Sci.*, 80(1-2):91–125, 1994.





Christian S. Jensen et Al.

The consensus glossary of temporal database concepts -  
february 1998 version.

In *Temporal Databases*, *Dagstuhl*, pages 367–405, 1998.



## References on temporal data models

-  Christian S. Jensen and Richard T. Snodgrass.  
Temporal specialization and generalization.  
*IEEE Trans. Knowl. Data Eng.*, 6(6):954–974, 1994.
-  Gultekin Özsoyoglu and Richard T. Snodgrass.  
Temporal and real-time databases: A survey.  
*IEEE Trans. Knowl. Data Eng.*, 7(4):513–532, 1995.



## References on temporal extensions to SQL



John Bair, Michael H. Böhlen, Christian S. Jensen, and Richard T. Snodgrass.

Notions of upward compatibility of temporal query languages.

*Wirtschaftsinformatik*, 39(1):25–34, 1997.



Richard T. Snodgrass.

*Developing Time-Oriented Database Applications in SQL*.  
Morgan Kaufmann Publishers, 2000.






Cindy Xinmin Chen, Jiejun Kong, and Carlo Zaniolo.

Design and implementation of a temporal extension of SQL.

In U. Dayal, K. Ramamritham, and T. M. Vijayaraman, eds,  
*ICDE*, pages 689–691. IEEE Computer Society, 2003.



## References on temporal extensions of SQL

-  Richard T. Snodgrass, Michael H. Böhlen, Christian S. Jensen, and Andreas Steiner.  
Transitioning temporal support in TSQL2 to SQL3.  
*In Temporal Databases, Dagstuhl*, pages 150–194, 1997.
-  Jose Ramon Rios Viqueira and Nikos A. Lorentzos.  
SQL extension for spatio-temporal data.  
*VLDB J.*, 16(2):179–200, 2007.
-  Esteban Zimányi.  
Temporal aggregates and temporal universal quantification  
in standard SQL.  
*SIGMOD Record*, 35(2):16–21, 2006.



## References on time in clinical information systems



Yuval Shahar and Carlo Combi.

Intelligent Temporal Information Systems in Medicine.

*J. Intelligent Information Systems, Special Issue*, 13(1-2):  
5-8, 1999.



Carlo Combi and Giuseppe Pozzi.

Temporal representation and reasoning in medicine.

*Artificial Intelligence in Medicine, Special Issue: Temporal  
representation and reasoning in medicine.*, 38(2):97-100,  
2006.



Juan Carlos Augusto.

Temporal reasoning for decision support in medicine.

*Artificial Intelligence in Medicine*, 33(1):1–24, 2005.



## References on time in clinical information systems



Yuval Shahar and Carlo Combi.

Temporal reasoning and temporal data maintenance in medicine: Issues and challenges.

*Computer in Biology and Medicine, Special Issue: Time-Oriented Systems in Medicine*, 27(5):353–368, 1997.



Carlo Combi, Giorgio Cucchi, and Francesco Pinciroli.

Applying object-oriented technologies in modeling and querying temporally-oriented clinical databases dealing with temporal granularity and indeterminacy.

*IEEE Transactions on Information Technology in Biomedicine*, 1:100–127, 1997.

