Querying Data with Multiple Temporal Dimensions

Carlo Combi, Dipartimento di Informatica,

Università degli Studi di Verona,

Ca' Vignal 2, strada le Grazie, 15 — I-37134 Verona, Italy,

combi@sci.univr.it

Angelo Montanari, Dipartimento di Matematica e Informatica,

Università degli Studi di Udine,

Via delle Scienze, 206 — I-33100 Udine, Italy

montana@dimi.uniud.it

Summary. *We focus on the problem of designing a query language for data models provided with four temporal dimensions:* valid time, transaction time, event time, and availability time.

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.

Example 1. On August 10, 1998, the physician prescribes a bipuvac-based therapy **from 10:00 to 14:00**. Data about the therapy is entered **at 9:00**. Due to the unexpected evolution of the patient state, 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**. (Bipuvac and diazepam are drugs commonly used in anesthesia.)

Drug	VT	TT
bipuvac	[98Aug10;10:00, 98Aug10;14:00)	[98Aug10;9:00, 98Aug10;12:00)
bipuvac	[98Aug10;10:00, 98Aug10;11:15)	[98Aug10;12:00, ∞)
diazepam	[98Aug10;11:25, 98Aug10;14:00)	[98Aug10;12:00, ∞)

Event Time

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

Example 1 (revisited). On August 10, 1998, **at 8:00**, the physician prescribes a bipuvac-based therapy from 10:00 to 14:00. Data is entered 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.

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, ∞)
diazepam	[98Aug10;11:25, 98Aug10;14:00)	98Aug10;11:00	98Aug10;11:00	[98Aug10;12:00, ∞)

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 2. 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. **On October 10**, 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 **immediately** entered this data into the database.

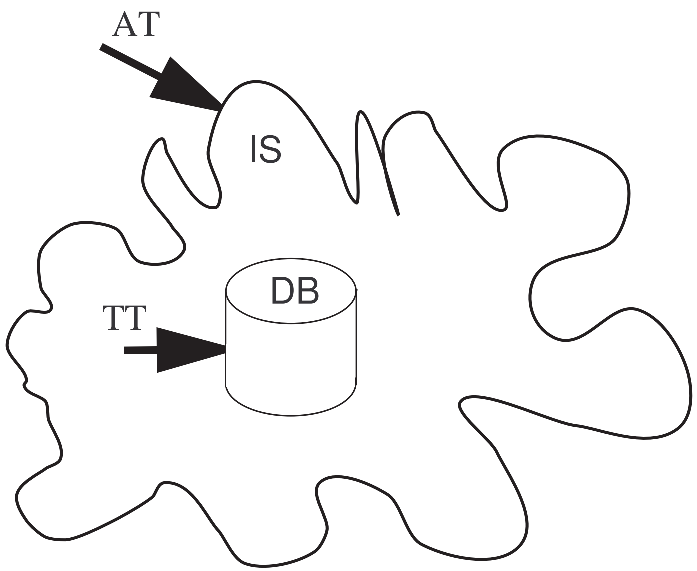
Availability Time (cont'd)

Example 2 (cont'd). 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. **On October 21**, the physician entered the correct data into the database.

symptom	VT	ET_i	ET_t	AT	TT
headache	[97Oct1, ∞)	97Sept5	null	[97Oct7, 97Oct15)	[97Oct10, 97Oct15)
headache	[97Oct1, 97Oct14)	97Sept5	97Oct9	[97Oct15, 97Oct20)	[97Oct15, 97Oct21)
headache	[97Oct1, 97Oct14)	97Sept15	97Oct9	[97Oct20, ∞)	[97Oct21, ∞)

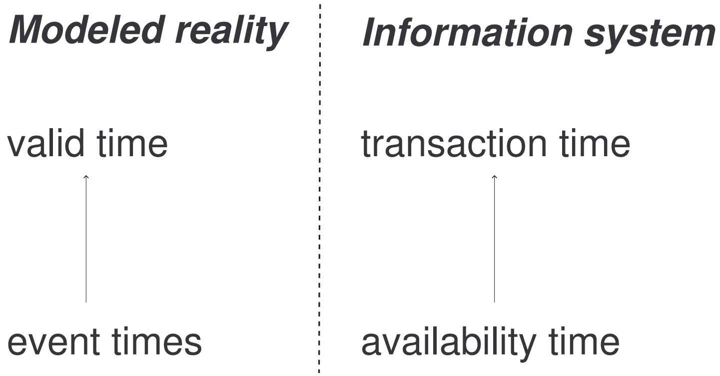
Transaction and Availability Times

Availability (AT) and transaction (TT) times and their relationships with the database (DB) and information systems (IS):



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

The complete picture



- **Transaction time** is append-only, while **valid** and **event times** can be freely modified.
- 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, and we cannot exclude such a possibility. Furthermore, even assuming data entry without errors, availability and transaction times may "append" facts according to two different orders.

Querying data with multiple temporal dimensions

The management of multiple temporal dimensions has an impact on many components of query languages.

We restrict our attention to those aspects specifically related to queries involving the four temporal dimensions.

Example: a clinical database

Tables *pat_sympt* (patient symptoms) and *pat_ther* (patient therapy):

pat_sympt(P_id, symptom, VT, ET_i, ET_t, AT, TT)
pat_ther(P_id, therapy, VT, ET_i, ET_t, AT, TT)

An instance of the clinical database

P_id	symptom	VT	ET_i	ET_t	AT	TT
1	headache	[97Oct1, ∞)	97Sept5	null	[97Oct7, 97Oct15)	[97Oct10, 97Oct15)
2	vertigo	[97Aug8, 97Aug15)	97Aug7	97Aug12	[97Sept3, 97Oct17)	[97Oct15, 97Oct21)
2	vertigo	[97Aug10, 97Aug15)	97Aug7	97Aug12	[97Oct19, ∞)	[97Oct21, ∞)
1	headache	[97Oct1, 97Oct14)	97Sept5	97Oct9	[97Oct15, 97Oct20)	[97Oct15, 97Oct21)
1	headache	[97Oct1, 97Oct14)	97Sept15	97Oct9	[97Oct20, ∞)	[97Oct21, ∞)

P_id	therapy	VT	ET_i	ET_t	AT	TT
1	aspirin	[96Oct1, 96Oct20]	96Sept25	96Oct16	[97Oct7, ∞)	[97Oct10, ∞)
2	paracetamol	[97Aug11, 97Aug12]	97Aug11	97Aug12	[97Sept3, ∞)	[97Oct15, ∞)

Checking temporal conditions: WHERE and WHEN

Comparisons between temporal dimensions can be supported in two alternative ways: the first one is to deal with **temporal and non-temporal data** in the `WHERE` clause; the second one is to constrain **purely temporal conditions** to appear in an ad-hoc `WHEN` clause.

Query 1. Determine all and only the patient's symptoms that appeared no later than 3 days after their initiating event.

```
SELECT symptom
FROM pat_sympt S
WHEN BEGIN(VALID(S)) - INITIATING_ET(S) ≤ 3
```

Querying past states: AS OF and AS KNOWN

By default, a query is evaluated with respect to the current state of the database and to the knowledge currently available to the information system. However, there must be the possibility of evaluating it over **past states** of the database and/or of the information system.

Query 2. Determine which data was available to the physician on October 18, according to the database contents on October 20:

```
SELECT *
FROM pat_sympt S
AS OF 97Oct20
AS KNOWN 97Oct18
```

P_id	symptom	VT	ET_i	ET_t	AT	TT
1	headache	[97Oct1, 97Oct14)	97Sept5	97Oct9	[97Oct15, 97Oct20)	[97Oct15, 97Oct21)

Temporal joins

Besides the standard (atemporal) join, the language must support some forms of **temporal join**, where the join condition allows the user to join tuples taking into account different temporal features.

A common distinction: **sequenced** vs. **non-sequenced semantics**

Query 3. Determine the symptoms of patients for which the valid time intervals of symptoms and therapies intersect.

```
SELECT symptom, S.P_id
FROM pat_sympt S, pat_ther T
WHEN NOT(VALID(S) BEFORE VALID(T)) AND
  NOT(VALID(T) BEFORE VALID(S))
WHERE S.P_id = T.P_id
```

Defining temporal dimensions of query results

The language must provide some mechanisms to allow the user to define the (values of the) temporal dimensions of query results.

- **Transaction** and **availability** times can be obtained as the intersection of the transaction and availability times of the involved tuples, respectively.
- Some default criteria must be specified to determine the values of the **valid** and **event times** of the resulting relation, whenever the user does not provide any explicit rule (we expect that, in most cases, the user explicitly defines the way in which the valid and event times of the resulting relation must be determined).

Default vs. user-defined criteria

```
SELECT symptom, S.P_id     # default criteria
FROM pat_sympt S TJOIN pat_ther T ON
   S.P_id = T.P_id AND NOT(VALID(S) BEFORE
   VALID(T)) AND NOT(VALID(T) BEFORE VALID(S))
```

P_id	symptom	VT	ET_i	ET_t	AT	TT
2	vertigo	[97Aug11, 97Aug12]	97Aug11	97Aug12	[97Oct19, ∞)	[97Oct21, ∞)

```
SELECT symptom, S.P_id     # user-defined criteria
   WITH VALID(S) AS VALID, INITIATING_ET(S) AS
   INITIATING, TERMINATING_ET(S) AS TERMINATING
FROM pat_sympt S TJOIN pat_ther T ON ...
```

P_id	symptom	VT	ET_i	ET_t	AT	TT
2	vertigo	[97Aug10, 97Aug15]	97Aug7	97Aug12	[97Oct19, ∞)	[97Oct21, ∞)

Conclusions

Further temporal features (e.g., **TIME SLICE** and **MOVING WINDOWS** clauses) are reported in the paper.

The **key point** of our approach to the problem of querying data with multiple temporal dimensions can be summarized as follows.

Whether or not query languages for temporal databases must provide a special support for event and availability times is **debatable**; in the past years, there has been a dispute even about the inclusion of built-in supports for dealing with valid and transaction times.

However, the design issues involved in the definition of a query language for temporal databases show that **all the four temporal dimensions must be taken into account** starting from the high-level tasks of requirement engineering and conceptual design (also in the case in which one decides to manage them by using a classical atemporal database system).