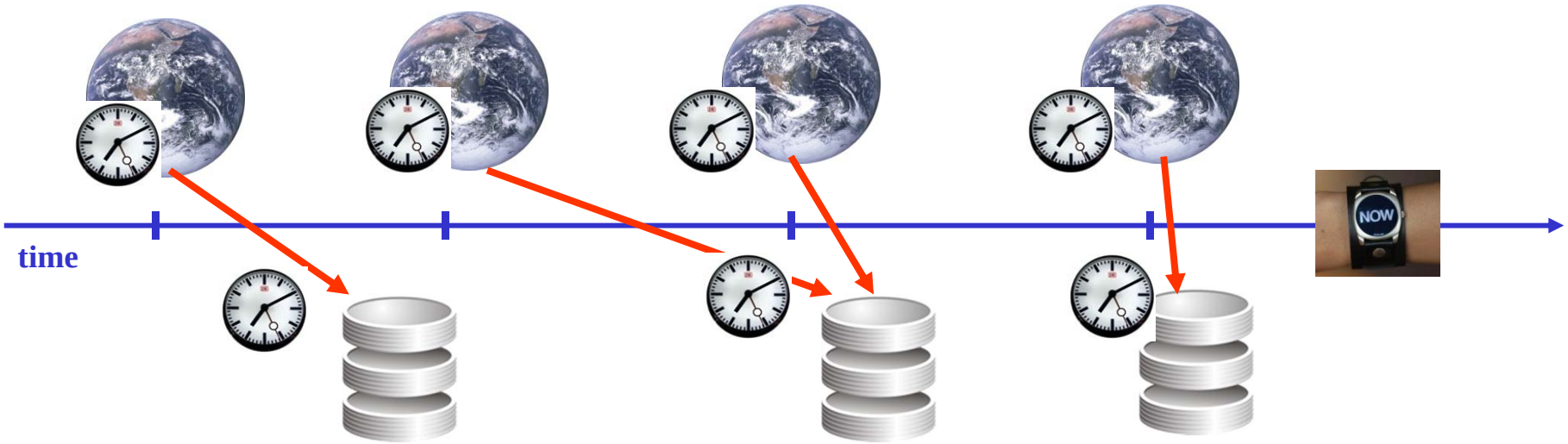
Temporal Information Systems

SS 2015

Bi-Temporal Databases – Managing History in Two Dimensions

Chapter 5

Bi-Temporal Data Management

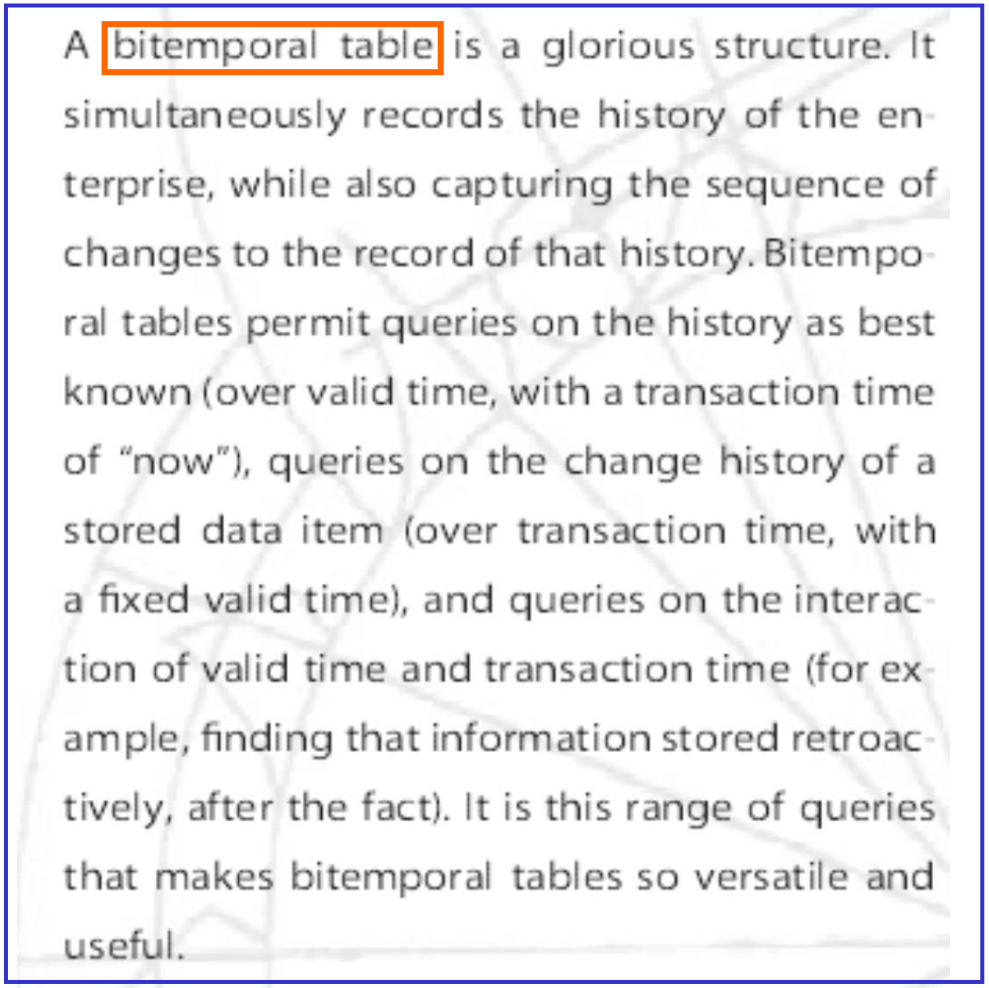


- In this chapter, we will discuss databases where certain tables contain temporal data (i.e., leep histories) about both, the real world (valid time) and the database (transaction time).

- Such tables are called bi-temporal tables (from lat. „bi“=„two times, twice“) .

- In some research contributions, another „temporal dimension“ is proposed: Columns containing temporal data elements for which it is not known (or irrelevant) whether they refer to the „outside world“ or are maintained „inside the DBMS“ are called user-defined time columns.

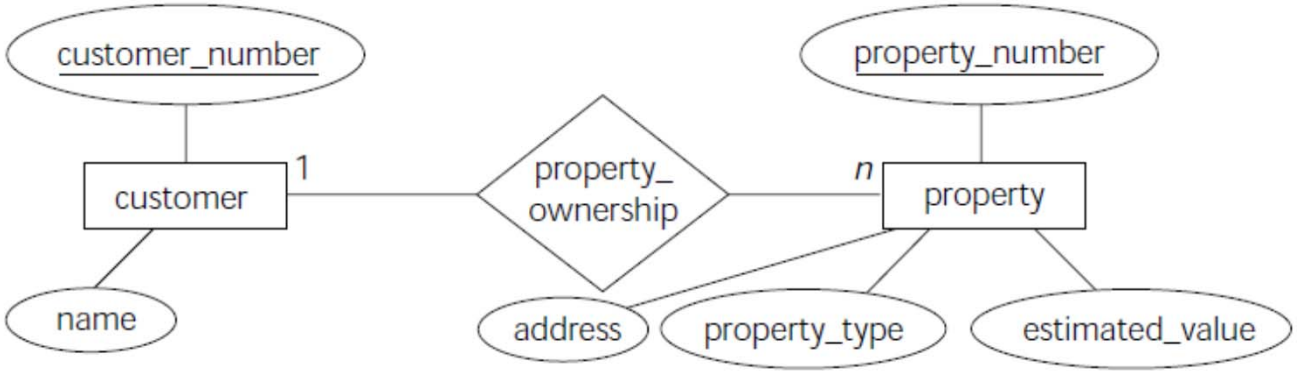
In Praise of Bitemporality



A bitemporal table is a glorious structure. It simultaneously records the history of the enterprise, while also capturing the sequence of changes to the record of that history. Bitemporal tables permit queries on the history as best known (over valid time, with a transaction time of "now"), queries on the change history of a stored data item (over transaction time, with a fixed valid time), and queries on the interaction of valid time and transaction time (for example, finding that information stored retroactively, after the fact). It is this range of queries that makes bitemporal tables so versatile and useful.

(from: R. Snodgrass „Developing Time-Oriented DB Applications in SQL", p. 276)

Running Example on Bitemporal Data Management



Owner (customer, property, VT_Begin, VT_End, TT_Start, TT_Stop)

Customer (name, VT_Begin, VT_End, TT_Start, TT_Stop)
Property (nr, address, type, value, VT_Begin, VT_End, TT_Start, TT_Stop)

(VT: valid time, TT: transaction time)

- The example and the following discussion is once again based on the book by Richard Snodgrass, this time on chapter 10 (relational schema slightly modified).
- We are going to discuss modifications and queries, in order to illustrate typical problems of maintaining bitemporality.

Modifications of Bitemporal Tables

- On the following slides, we are going to discuss how to properly modify bitemporal tables (by means of recording several property sales transactions in the example DB).

- All of these modifications are current modifications as far as transaction time is concerned, thus it is not allowed to the user of the database to
  - … logically modify any of the TT timestamps once generated, or to
  - … logically delete any records once created. Only the DBMS is allowed to do so!

- Nevertheless, valid time information may be changed, if „better information“ about the past becomes available (sequenced modifications for VT).

- We do not consider any VT entries for future states here, different from Snodgrass!

- Thus, we again start with logical changes (corresponding to non-temporal versions of the modification) and derive suitable physical implementations from them.

- Snodgrass recommends to proceed in two stages:
  - First, express the valid time implementation (ignoring TT for the moment).
  - Then consider transaction time in a second step.

Current Insertion (1)

1) On January 10, 1998, Eva buys property 7797. This fact is recorded in the DB on that day.

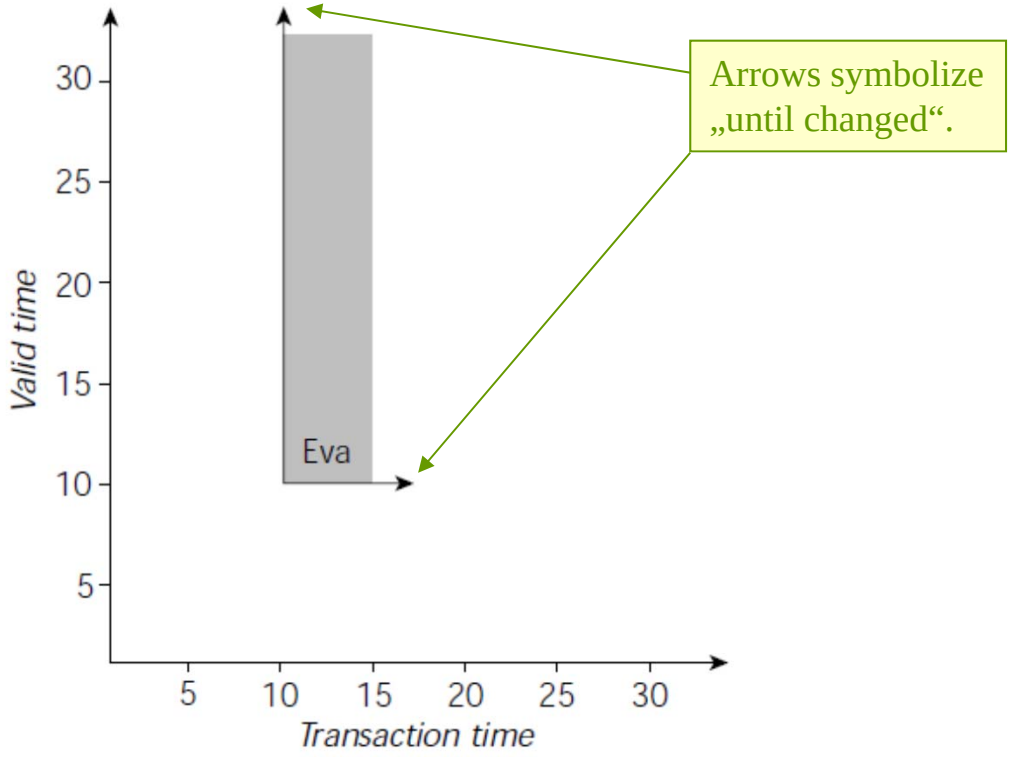
```
INSERT INTO Owner (customer, property, VT_Begin, VT_End, TT_Start, TT_Stop)
VALUES ("Eva", 7797, CURRENT_DATE, DATE '9999-12-31', CURRENT_DATE, DATE '9999-12-31')
```

(blue: logical insertion, red: physical after adding VT, green: physical with TT)

Corresponding

Bitemporal Time Diagram

(technique introduced by C. Jensen)



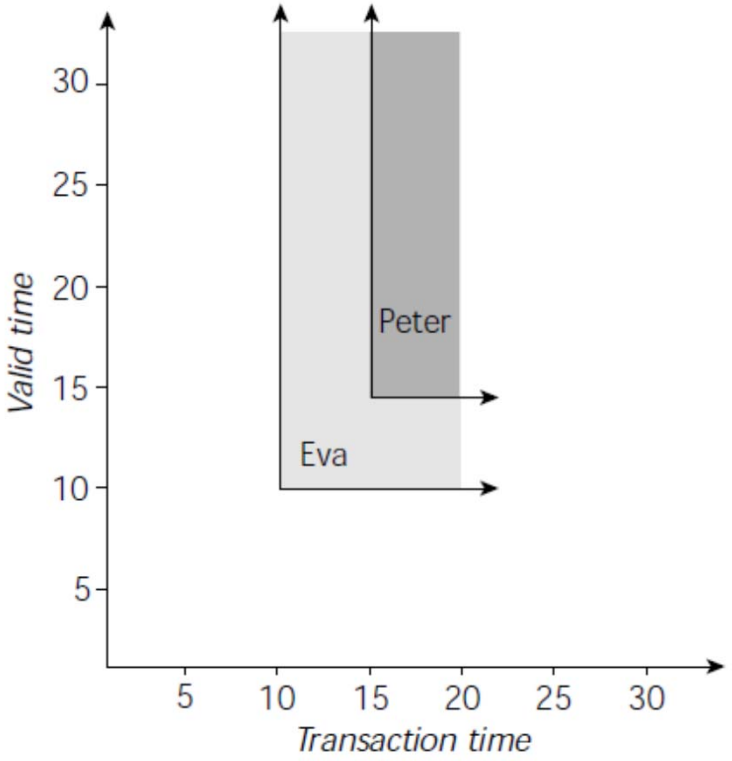
(CURRENT_DATE: 1998-01-**10**)

Current Update (1)

2) On January 15, Eva sells property 7797 to Peter.

logical update (non-temporal):

```
UPDATE  Owner
SET     customer = "Peter"
WHERE   property = 7797
```



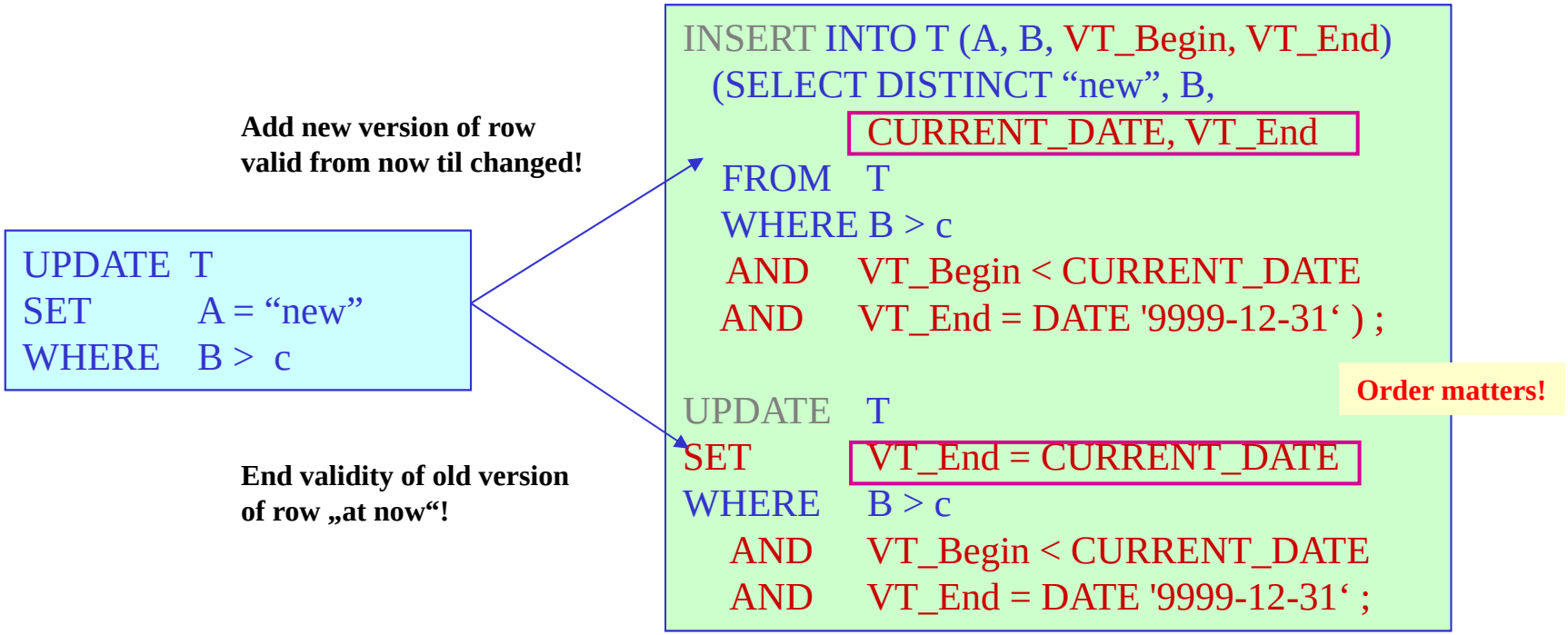
Again assume immediate recording in the DB:

(CURRENT_DATE: 1998-01-**15**)

Reminder: Mapping Current Updates in General

- In Chap. 3, we already discussed translating logical current updates into physical current updates for TT tables. Remember: We deviated slightly from Snodgrass‘ translation!

- Now, we deal with VT current updates, but the general idea is the same (keeping in mind that the result of translation is not yet physical, but a logical intermediate step). Therefore we will have to deviate from Snodgrass again. Here is the general format:



Current Update (2)

1st step:
Consider current valid time modifications only, applying the general technique from previous slide!

```
UPDATE   Owner
SET      customer = "Peter"
WHERE    property = 7797
```

(still logical)

```
INSERT INTO Owner
  (SELECT  DISTINCT  "Peter", property,
                     CURRENT_DATE, VT_End
   FROM      Owner
   WHERE     property = 7797
      AND    VT_Begin < CURRENT_DATE
      AND    VT_End = DATE '9999-12-31' )
```

(still logical, order of execution matters)

```
UPDATE   Owner
SET      VT_End = CURRENT_DATE
WHERE    property = 7797
   AND   VT_Begin < CURRENT_DATE
   AND   VT_End = DATE '9999-12-31'
```

Applied to old version only (with owner Eva)!

(CURRENT_DATE: 1998-01-**15**)

Current Update (3)

```
INSERT INTO Owner
  (SELECT DISTINCT   "Peter", property,
                CURRENT_DATE, VT_End
    FROM        Owner
    WHERE       property = 7797
        AND     VT_Begin < CURRENT_DATE
        AND     VT_End = DATE '9999-12-31'
```

(logical + VT)

```
INSERT INTO Owner
  (SELECT DISTINCT "Peter", property,
                CURRENT_DATE, VT_End,
                CURRENT_DATE, TT_Stop
   FROM         Owner
   WHERE        property = 7797
       AND      VT_Begin < CURRENT_DATE
       AND      VT_End = DATE '9999-12-31'
       AND      TT_Stop = DATE '9999-12-31' )
```

(physical)

2nd step:

Add TT modifications, too!

Treat logical UPDATE as before!

```
UPDATE      Owner
SET         VT_End = CURRENT_DATE
WHERE       property = 7797
    AND     VT_Begin < CURRENT_DATE
    AND     VT_End = DATE '9999-12-31'
```

(logical + VT)

Eva!

```
INSERT INTO Owner
  (SELECT DISTINCT  customer, property,
                VT_Begin, CURRENT_DATE,
                CURRENT_DATE, TT_Stop
   FROM         Owner
   WHERE        property = 7797
       AND      VT_Begin < CURRENT_DATE
       AND      VT_End = DATE '9999-12-31'
       AND      TT_Stop = DATE '9999-12-31' )
```

```
UPDATE      Owner
SET         TT_Stop = CURRENT_DATE
WHERE       property = 7797
    AND     VT_Begin < CURRENT_DATE
    AND     VT_End = DATE '9999-12-31'
    AND     TT_Stop = DATE '9999-12-31'
```

(CURRENT_DATE: 1998-01-**15**)

Current Update (4)

State of table „Owner“ before implementing the sales from Eva to Peter:

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	9999-12-31

1st step: Add information about the new owner valid from now till changed in both, VT and TT:

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	9999-12-31
Peter	7797	1998-01-15	9999-12-31	1998-01-15	9999-12-31

2nd step: Add new version (TT now) of old owner row, a copy of old Eva row with VT validity ended:

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	9999-12-31
Peter	7797	1998-01-15	9999-12-31	1998-01-15	9999-12-31
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31

3rd step: End TT validity of old Eva row:

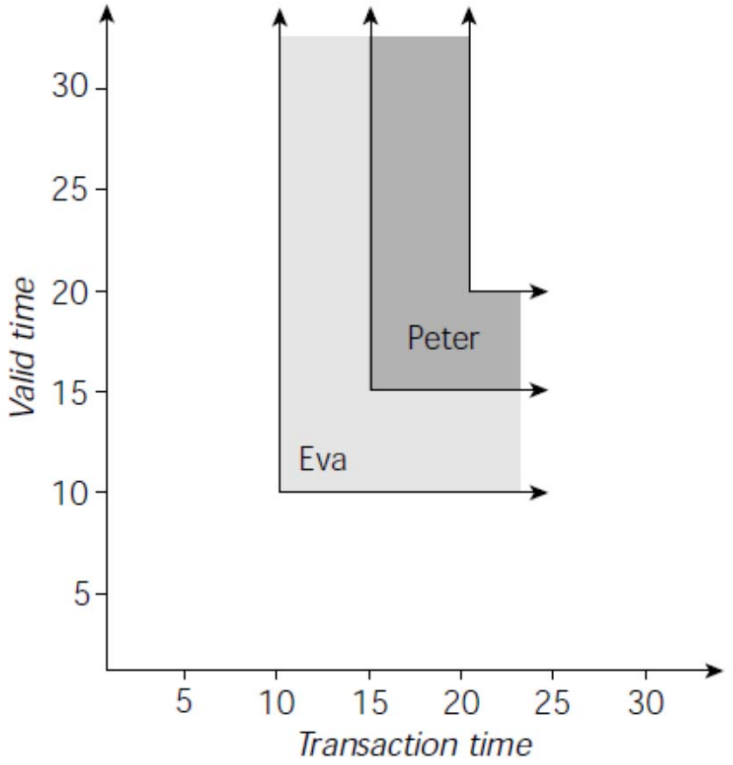
customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Peter	7797	1998-01-15	9999-12-31	1998-01-15	9999-12-31
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31

Current Deletion (1)

3) Peter sells the flat on January 20
(to somebody who is not in our customer table, thus this flat is no longer visible to us).

logical update (non-temporal):

```
DELETE FROM Owner
WHERE  property = 7797
```



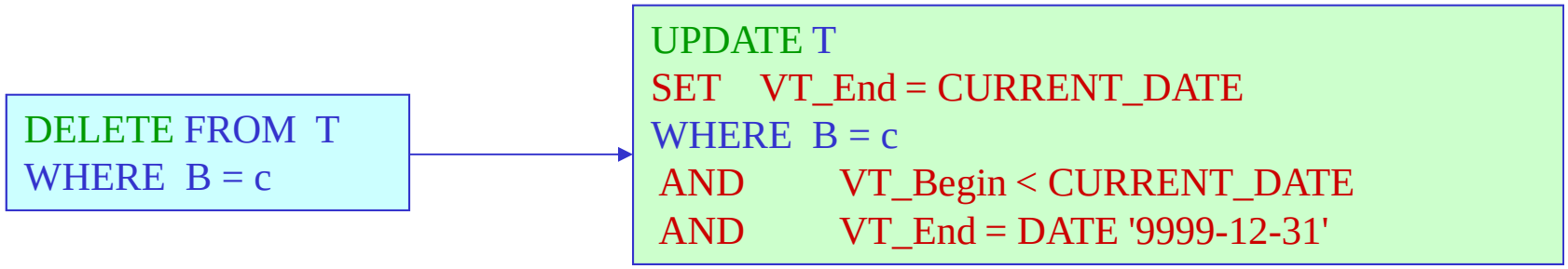
(CURRENT_DATE: 1998-01-**20**)

Reminder: Mapping Current Deletions in General

- As for mapping logical deletions to physical updates, we deviated from Snodgrass‘ proposal in chapter 3 – and have to do the same here, thus deviating from his bi-temporal proposal as well.

- Again, we first perform the translation previously applied to TT only to the VT part of the intended change (and then map the resulting „in between“ update to its physical TT counterpart).

- As a reminder, see the principle of mapping logical deletions to physical updates again:



Current Deletion (2)

Now apply this technique to the 3rd event in our „sales story“:

Peter sells the flat on January 20.

logical deletion
(non-temporal)

```
DELETE FROM Owner
WHERE property = 7797
```


still logical update
(including VT changes)

```
UPDATE Owner
SET    VT_End = CURRENT_DATE
WHERE  property = 7797
   AND VT_Begin < CURRENT_DATE
   AND VT_End = DATE '9999-12-31'
```

Current Deletion (3)

Now let the TT dimension follow:

```
UPDATE Owner
SET       VT_End = CURRENT_DATE
WHERE     property = 7797
   AND    VT_Begin < CURRENT_DATE
   AND    VT_End = DATE '9999-12-31'
```

(logical + VT)

```
INSERT INTO Owner
  (SELECT  customer, property,
           VT_Begin, CURRENT_DATE,
           CURRENT_DATE, DATE '9999-12-31'
   FROM    Owner
   WHERE   property = 7797
     AND   VT_Begin < CURRENT_DATE
     AND   VT_End = DATE '9999-12-31'
     AND   TT_Stop = DATE '9999-12-31' )

UPDATE    Owner
SET       TT_Stop = CURRENT_DATE
WHERE     property = 7797
   AND    VT_Begin < CURRENT_DATE
   AND    VT_End = DATE '9999-12-31'
   AND    TT_Stop = DATE '9999-12-31'
```

(physical)

Current Deletion (4)

State of table „Owner“ before implementing the sales from Peter to the anonymous buyer (on Jan. 20):

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Peter	7797	1998-01-15	9999-12-31	1998-01-15	9999-12-31
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31

1st step: Add information about the new end date of validity of Peter‘s ownership in VT (do so TT now):

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Peter	7797	1998-01-15	9999-12-31	1998-01-15	9999-12-31
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31
Peter	7797	1998-01-15	1998-01-20	1998-01-20	9999-12-31

2nd step: End TT validity of old Peter row:

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Peter	7797	1998-01-15	9999-12-31	1998-01-15	1998-01-20
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31
Peter	7797	1998-01-15	1998-01-20	1998-01-20	9999-12-31

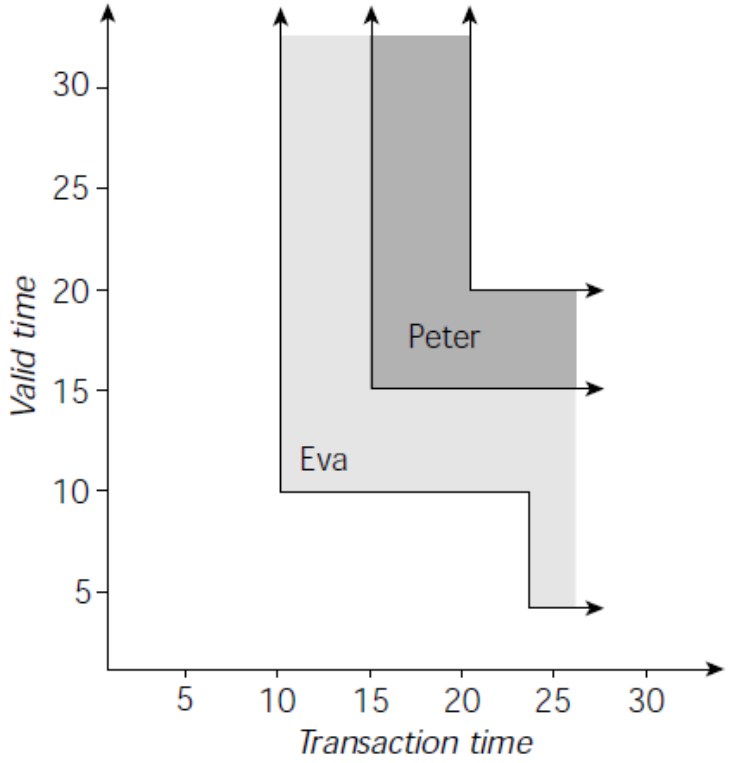
VT Sequenced Insertion

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31
Peter	7797	1998-01-15	9999-12-31	1998-01-15	1998-01-20
Peter	7797	1998-01-15	1998-01-20	1998-01-20	9999-12-31
Eva	7797	1998-01-03	1998-01-10	1998-01-23	9999-12-31

4) On January 23 we find out that Eva had actually purchased the flat already on January 3 rather than on 10, so one week earlier – we add the additional week.

logically: VT sequenced insertion

physically: TT current insertion



VT Sequenced Deletion

5) On January 26 we learn that Eva bought the flat not on January 10, nor on January 3, but (really?) on January 5. This can be viewed as a sequenced deletion for Jan. 3 and 4.

logically: VT sequenced deletion
physically: TT current insertion + TT current update

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31
Peter	7797	1998-01-15	9999-12-31	1998-01-15	1998-01-20
Peter	7797	1998-01-15	1998-01-20	1998-01-20	9999-12-31
Eva	7797	1998-01-03	1998-01-10	1998-01-23	9999-12-31

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Eva	7797	1998-01-10	1998-01-15	1998-01-15	9999-12-31
Peter	7797	1998-01-15	9999-12-31	1998-01-15	1998-01-20
Peter	7797	1998-01-15	1998-01-20	1998-01-20	9999-12-31
Eva	7797	1998-01-03	1998-01-10	1998-01-23	1998-01-26
Eva	7797	1998-01-05	1998-01-10	1998-01-26	9999-12-31

VT Sequenced Update

6) On January 28 we learn that Peter bought the flat from Eva on January 12, not 15.

logically: VT sequenced update
physically: 2 TT current insertions + 3 TT current updates

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Eva	**7797**	**1998-01-10**	**1998-01-15**	**1998-01-15**	**9999-12-31**
Peter	7797	1998-01-15	9999-12-31	1998-01-15	1998-01-20
Peter	**7797**	**1998-01-15**	**1998-01-20**	**1998-01-20**	**9999-12-31**
Eva	7797	1998-01-03	1998-01-10	1998-01-23	1998-01-26
Eva	7797	1998-01-05	1998-01-10	1998-01-26	9999-12-31

customer	property	VT Begin	VT End	TT Start	TT Stop
Eva	7797	1998-01-10	9999-12-31	1998-01-10	1998-01-15
Eva	7797	1998-01-10	1998-01-15	1998-01-15	1998-01-28
Peter	7797	1998-01-15	9999-12-31	1998-01-15	1998-01-20
Peter	7797	1998-01-15	1998-01-20	1998-01-20	1998-01-28
Eva	7797	1998-01-03	1998-01-10	1998-01-23	1998-01-26
Eva	7797	1998-01-05	1998-01-10	1998-01-26	1998-01-28
Eva	7797	1998-01-05	1998-01-12	1998-01-28	9999-12-31
Peter	7797	1998-01-12	1998-01-20	1998-01-28	9999-12-31

… and once more: Slides from Kulkarni‘s tutorial on SQL:2011 and time

SQL:2011: System-Versioned Application Time Period Tables (1)

```
CREATE TABLE employees
(emp_name VARCHAR(50) NOT NULL PRIMARY KEY,
dept_id VARCHAR(10),
start_date DATE NOT NULL,
end_date DATE NOT NULL,
system_start TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
System_end TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
PERIOD FOR emp_period (start_date, end_date),
PERIOD FOR SYSTEM_TIME (system_start, system_end),
PRIMARY KEY (emp_name, emp_period WITHOUT OVERLAPS),
FOREIGN KEY (dept_id, PERIOD emp_period) REFERENCES
                    departments (dept_id, PERIOD dept_period)
) WITH SYSTEM VERSIONING;
```

Reminder!:
There are no constraints for system time columns as these are automatically set by the system (and thus can‘t be prevented)!

(example from K. Kulkarni „Temporal Features in SQL Standard“)

SQL:2011: System-Versioned Application Time Period Tables (2)

Inserting rows into a (formerly called) bitemporal table – application time values this time are future dates (planning), system time values are current and generated by the system:

On 11/01/1995, *employees* table was updated to show that John and Tracy will be joining the departments J13 & K25, respectively, starting from 11/15/1995.

```
INSERT INTO employees (emp_name, dept_id, start_date, end_date)
VALUES ('John', 'J13', DATE '1995-11-15', DATE '9999-12-31'),
       ('Tracy','K25', DATE '1995-11-15', DATE '9999-12-31')
```

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J13	11/15/1995	12/31/9999	11/01/1995	12/31/9999
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

(example from K. Kulkarni „Temporal Features in SQL Standard“)

SQL:2011: System-Versioned Application Time Period Tables (3)

old state:

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J13	11/15/1995	12/31/9999	11/01/1995	12/31/9999
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

On 11/10/1995, it was discovered that John was assigned to the wrong department.
It was changed to department J15 on that day.

```
UPDATE employees
SET dept_id = 'J15"
WHERE emp_name = 'John'
```

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	11/15/1995	12/31/9999	11/10/1995	12/31/9999
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

(example from K. Kulkarni „Temporal Features in SQL Standard“)

SQL:2011: System-Versioned Application Time Period Tables (4)

old state:

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	11/15/1995	12/31/9999	11/10/1995	12/31/9999
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

On 12/15/1997, John is loaned to Dept M12 starting from 1/1/1998 to 7/1/1998.

```
UPDATE employees FOR PORTION OF emp_period FROM
    DATE '1998-01-01' TO DATE '1998-07-01'
SET dept_id = 'M12' WHERE emp_name = 'John'
```

employees

Emp_name	dept_id	etart_date	end_date	system_start	system_end
John	J15	07/01/1998	12/31/9999	12/15/1997	12/31/9999
John	M12	01/01/1998	07/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	01/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

(example from K. Kulkarni „Temporal Features in SQL Standard“)

SQL:2011: System-Versioned Application Time Period Tables (5)

old state:

employees

Emp_name	dept_id	etart_date	end_date	system_start	system_end
John	J15	07/01/1998	12/31/9999	12/15/1997	12/31/9999
John	M12	01/01/1998	07/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	01/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

On 12/15/1998, John is approved for a leave of absence from 1/1/1999 to 1/1/2000.

```
DELETE FROM employees FOR PORTION OF emp_period FROM
    DATE '1999-01-01' TO DATE '2000-01-01'
WHERE emp_name = 'John'
```

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	01/01/2000	12/31/9999	12/15/1998	12/31/9999
John	J15	07/01/1998	01/01/1999	12/15/1998	12/31/9999
John	M12	01/01/1998	07/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	01/01/1998	12/15/1997	12/31/9999
John	J15	07/01/1998	12/31/9999	12/15/1997	12/15/1998
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

(example from K. Kulkarni „Temporal Features in SQL Standard“)

SQL:2011: System-Versioned Application Time Period Tables (6)

old state:

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	01/01/2000	12/31/9999	12/15/1998	12/31/9999
John	J15	07/01/1998	01/01/1999	12/15/1998	12/31/9999
John	M12	01/01/1998	07/01/1998	12/15/1997	12/31/9999
John	J15	11/15/1995	01/01/1998	12/15/1997	12/31/9999
John	J15	07/01/1998	12/31/9999	12/15/1997	12/15/1998
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

On 6/1/2000, John resigns from the company.

```
DELETE FROM employees
WHERE emp_name = 'John'
```

employees

emp_name	dept_id	start_date	end_date	system_start	system_end
John	J15	01/01/2000	12/31/9999	12/15/1998	06/01/2000
John	J15	07/01/1998	01/01/1999	12/15/1998	06/01/2000
John	M12	01/01/1998	07/01/1998	12/15/1997	06/01/2000
John	J15	11/15/1995	01/01/1998	12/15/1997	06/01/2000
John	J15	07/01/1998	12/31/1999	12/15/1997	12/15/1998
John	J15	11/15/1995	12/31/9999	11/10/1995	12/15/1997
John	J13	11/15/1995	12/31/9999	11/01/1995	11/10/1995
Tracy	K25	11/15/1995	12/31/9999	11/01/1995	12/31/9999

(example from K. Kulkarni „Temporal Features in SQL Standard“)

Temporal Queries on Bitemporal Tables

- We now turn to bitemporal queries, where the temporal aspects of the query condition refer to either VT, or TT, or to both. Thus, we ask queries about what happened in the „world“ and/or what the DB knows (or better: knew when) about these events.

- We first look at several examples of time-slice queries, asking about a particular day in history: VT only, TT only, and VT+TT.

- We then go through a range of bitemporal queries that are classified similarly to the classification used in chapter 3 (when discussing temporal queries against just TT tables):
    - current queries
    - sequenced queries
    - nonsequenced queries

Time Slice Query (1)

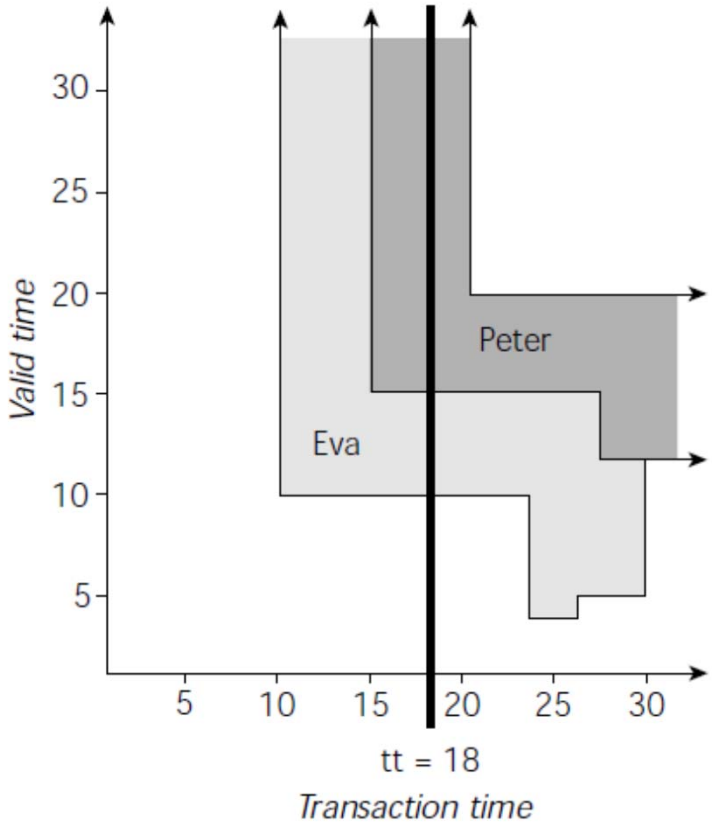
What was known about the history of flat 7797 on January 18, 1998?

TT time slice query:

```
SELECT   customer, VT_Begin, VT_End
FROM     Owner
WHERE    property = 7797
    AND  TT_Start <= DATE '1998-01-18'
    AND  DATE '1998-01-18' < TT_Stop
```

VT state table as answer:

customer	VT Begin	VT End
Eva	1998-01-10	1998-01-15
Peter	1998-01-15	9999-12-31



Time Slice Query (2)

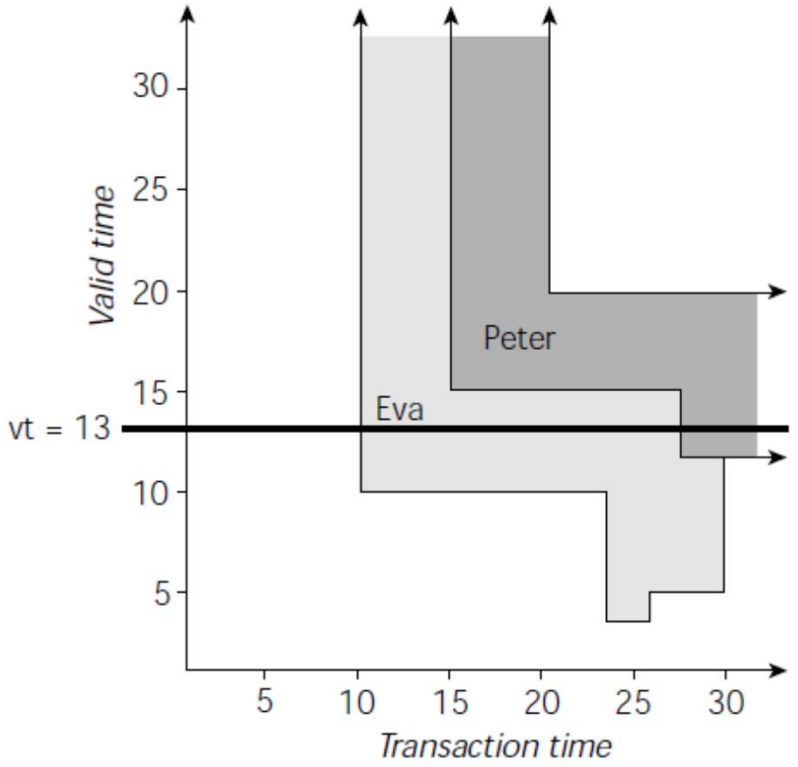
When was information about the owner of flat 7797 on January 13 recorded in the DB?

VT time slice query:

```
SELECT   customer, TT_Start, TT_Stop
FROM     Owner
WHERE    number = 7797
   AND   VT_Begin <= DATE '1998-01-13'
   AND   DATE '1998-01-13' < VT_End
```

TT state table as answer:

customer	TT Start	TT Stop
Eva	1998-01-10	1998-01-15
Eva	1998-01-15	1998-01-28
Peter	1998-01-28	9999-12-31

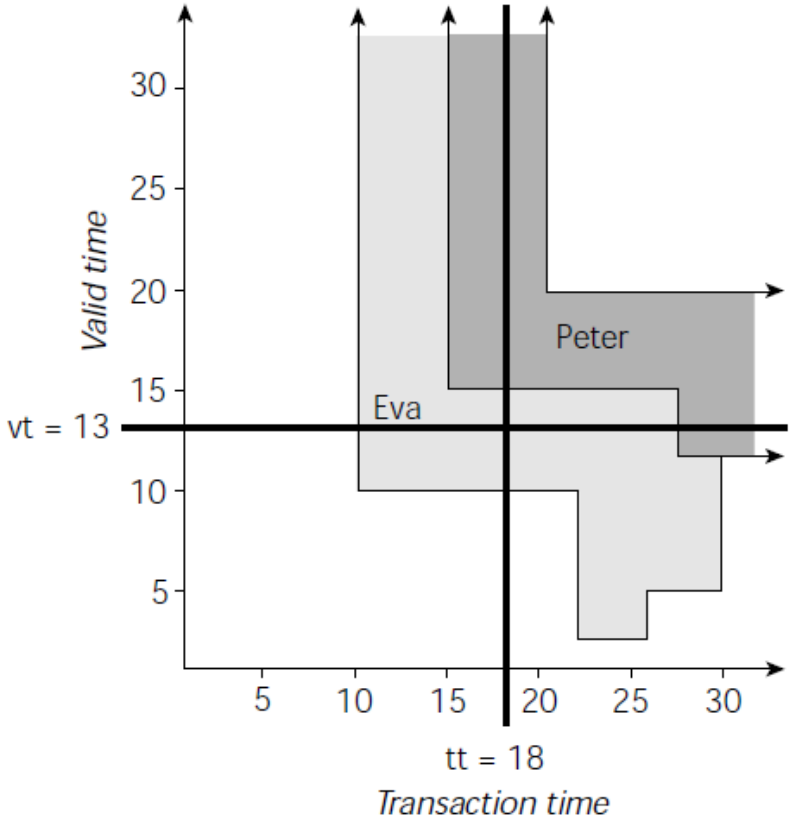


Time Slice Query (3)

Give the ownership history of flat 7797 on January 13, as stored in the DB on January 18!

Bitemporal time slice query:

```
SELECT   customer
FROM     Owner
WHERE    property = 7797
    AND  VT_Begin <= DATE '1998-01-13'
    AND  DATE '1998-01-13' < VT_End
    AND  TT_Start <= DATE '1998-01-18'
    AND  DATE '1998-01-18' < TT_Stop
```



The resulting (bitemporal) snapshot table (obviously omitting the timestamps) just contains the customer name „Eva“.

Spectrum of Bitemporal Queries Illustrated

- We conclude the discussion of bitemporality with a set of variants of an example query using different combinations of time reference for the two dimensions:
  - time-slice, in this example always „now“, i.e., current query
  - sequenced
  - nonsequenced

- If applying these query attributes to valid time, the meaning is as follows:
  - current: valid now
  - sequenced: history of validity
  - nonsequenced: valid at some time (ignoring historical sequence)

- If referring to transaction time, this is the meaning:
  - current: as best known today
  - sequenced: as recorded in the database (retaining history)
  - nonsequenced: as recorded at some time (ignoring history)

- Exploring all possible combinations thus means to look at nine different variants.