Temporal Database Concepts

April 16,2002

Michael Minock

Temporal Databases

We have allowed for *time* and *date* attributes in our schemas.

So we already have some experience with temporal databases.

Still it is necessary to develop some general concepts around the deeper modeling of temporal data.

Applications

- Health-care - does this patient have any record of heart ailments? (What about a family history?)

- Insurance - when is a policy in effect - is a particular accident covered?

- Reservation Systems - when does a reservation expire?

- Scientific Databases - when was a measurement taken?

Maintaining (and Querying) Histories

A unifying requirement for many temporal applications is the need to maintain an entire history of the changes to a tuple or object.

We need to be able to support queries that enable us to view 'the state' at any time in the past.

We might also like to support more complex time based queries as well.

Time Representation

Time: an ordered sequence of *points* of some *granularity*.

For example a granularity may be a *second*.

Some, who want to respect the continuity of time, will speak in terms of the inter-point durations (termed a *chronon*) instead of points.

Events occurring at the same point (or within the same chronon) will be seen as *simultaneous*.

Calendar

Need of a reference point from which to start measure.

A *calendar* is relative to a reference point and organizes time into different time time units for convenience. (Gregorian, Chinese, Islamic, Hindu, Jewish, Coptic, etc.)

In SQL2 we have:

- DATE (YYYY-MM-DD)
- TIME (HH:MM:SS)
- TIMESTAMP DATE and TIME together
- INTERVAL: A duration of time
- PERIOD: An anchored duration of time

Build your own types to get *milli*, *micro*, *nano*, and *pico* seconds. Or to get *centuries*,*millenia*, *millions*, and *billions* of years.

Event versus Duration Information

Point events are associated with a single time point.

Duration events are associate with a period of time and have a start and an end time-points.

Durations consist of a set of chronos that lie between the beginning and end time points (inclusive). Hence, based on the granularity this may be quite a few chronos. But it will be finite.

Valid Time and Transaction Time Dimensions

Given an association between a fact and particular point or duration event, there are two possible interpretations:

- It could be that this is the time the fact occurs or the period over which the fact is true *in the real world*. This is the notion of **valid time**.

- It could be that this is the time the fact is entered (or the period for which it is held) *i*n the database. This is the notion of **transaction time**.

Some databases use the first notion (*valid time database*), some use the second (*transaction time database*), and finally some use both time dimensions (*bitemporal databases*).

Figure 23.1 A simplified COMPANY database used for active rule examples.

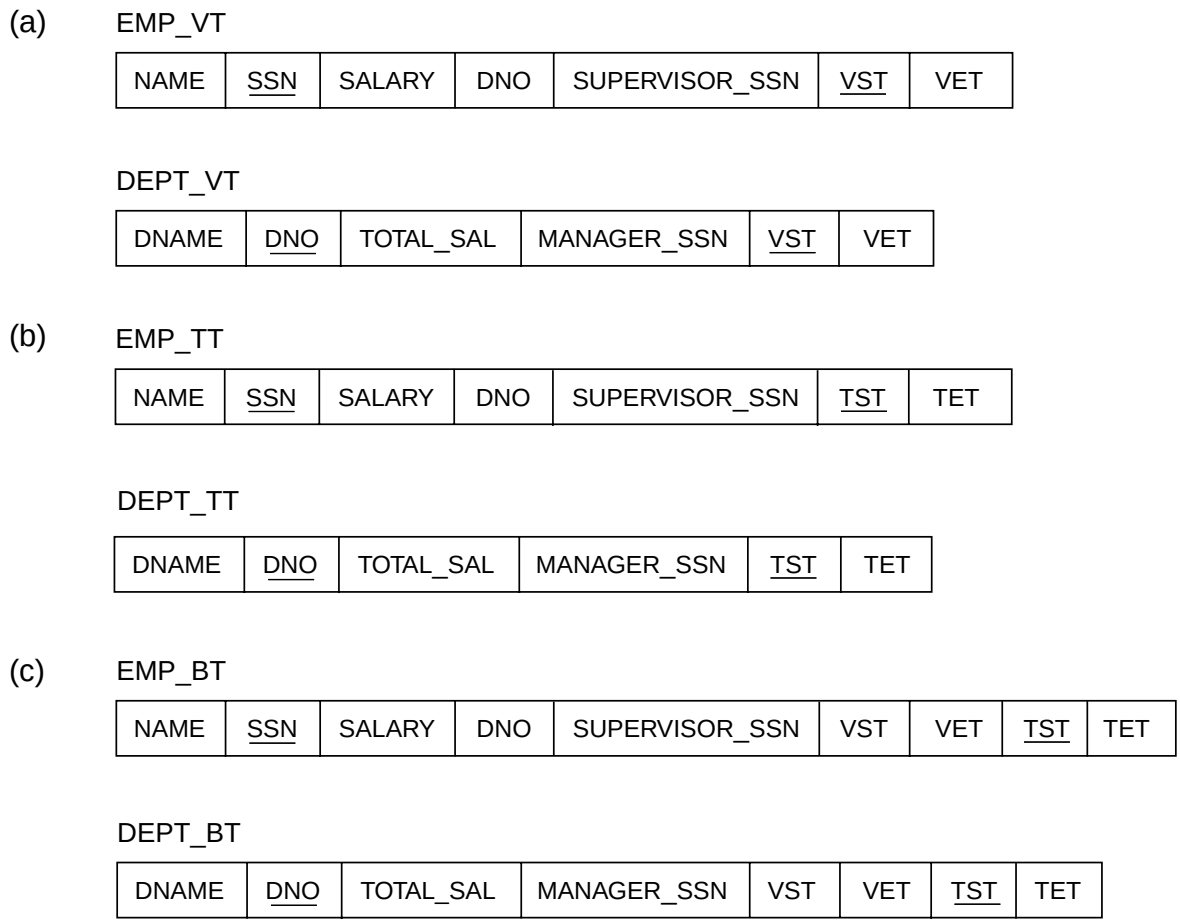
EMPLOYEE

NAME	SSN	SALARY	DNO	SUPERVISOR_SSN

DEPARTMENT

DNAME	DNO	TOTAL_SAL	MANAGER_SSN

Figure 23.6 Different types of temporal relational databases. (a) Valid time database schema. (b) Transaction time database schema. (c) Bitemporal database schema.



Valid Time Relations

- Add date types for valid start time and valid end-time.
- Each tuple represents a version of the employee.
- The current version has its valid end-time set to the special temporal variable `now`.
- The total key now includes the time period values.
- Non intersecting valid time points constraint may need to be observed (non-temporal keys may need to hold at every time point).

Figure 23.7 Some tuple versions in the valid time relations EMP_VT and DEPT_VT.

EMP_VT

NAME	SSN	SALARY	DNO	SUPERVISOR_SSN	VST	VET
Smith	123456789	25000	5	333445555	1997-06-15	1998-05-31
Smith	123456789	30000	5	333445555	1998-06-01	now
Wong	333445555	25000	4	999887777	1994-08-20	1996-01-31
Wong	333445555	30000	5	999887777	1996-02-01	1997-03-31
Wong	333445555	40000	5	888665555	1997-04-01	now
Brown	222447777	28000	4	999887777	1996-05-01	1997-08-10
Narayan	666884444	38000	5	333445555	1998-08-01	now

. . .

DEPT_VT

DNAME	DNO	MANAGER_SSN	VST	VET
Research	5	888665555	1996-09-20	1997-03-31
Research	5	333445555	1997-04-01	now

. . .

Updates, Deletes, and Inserts

- On updates the system should *close* the current version and create a new version with the updated values.
- Note that the user often needs to state the valid time of the update. Relying on the current date is often impractical.
- More often an update is a *proactive update* or *retroactive update*, but rarely it is simultaneous update.
- On deletes - just close the current version.
- On inserts - create the first version of the tuple.
- We require a *surrogate key* if the non-temporal primary key might change.

Transaction Time Relations

In a transaction time database, the actual state of the database through time may be recovered - unlike a valid time database.

Such databases are the most useful in applications where we have simultaneous updates. This is true in many financial domains.

The treatment is similar to a valid time database, but instead of `now` we have the variable `uc` (until changed).

These databases are also called *rollback databases*.

Bitemporal Relations

Sometimes we need to represent both transaction and valid time.

Tuples with `TET` as `uc`, represent currently valid information.

The current version has `TET` = `uc` and `VET` = `now`.

Updates

No attributes get changed in any tuple except `TET` on tuples with `TET = uc`.

Say we change current salary of `v`. Need `newSalary` and the `vst` when the new salary becomes effective.

```
changeSalary(v, newSalary){
  v2 = copy (v);
  v2.vet = vst-;  //immediately prior to VT
  v2.tst = ts(t); //transaction time
  v2.tet = uc;
  insert v2; // Valid old version with old salary

  v3 = copy (v);
  v3.vst = vt;
  v3.vet = now;
  v3.salary = newSalary;
  v3.tst =  ts(t);
  v3.tet = uc;
  insert v3; // new version

  v.tet = ts(t); // old state of affairs  }
```

Note that this how we handle this simple transaction. Think about more complex transactions.

Figure 23.8 Some tuple versions in the bitemporal relations EMP_BT and DEPT_BT.

EMP_BT

	NAME	SSN	SALARY	DNO	SUPERVISOR_SSN	VST	VET	TST	TET
v1:	Smith	123456789	25000	5	333445555	1997-06-15	now	1997-06-08,13:05:58	1998-06-04,08:56:12
v2:	Smith	123456789	25000	5	333445555	1997-06-15	1998-05-31	1998-06-04,08:56:12	uc
v3:	Smith	123456789	30000	5	333445555	1998-06-01	now	1998-06-04,08:56:12	uc
v4:	Wong	333445555	25000	4	999887777	1994-08-20	now	1994-08-20,11:18:23	1996-01-07,14:33:02
v5:	Wong	333445555	25000	4	999887777	1994-08-20	1996-01-31	1996-01-07,14:33:02	uc
v6:	Wong	333445555	30000	5	999887777	1996-02-01	now	1996-01-07,14:33:02	1997-03-28,09:23:57
v7:	Wong	333445555	30000	5	999887777	1996-02-01	1997-03-31	1997-03-28,09:23:57	uc
v8:	Wong	333445555	40000	5	888665555	1997-04-01	now	1997-03-28,09:23:57	uc
v9:	Brown	222447777	28000	4	999887777	1996-05-01	now	1996-04-27,16:22:05	1997-08-12,10:11:07
v10:	Brown	222447777	28000	4	999887777	1996-05-01	1997-08-10	1997-08-12,10:11:07	uc
v11:	Narayan	666884444	38000	5	333445555	1998-08-01	now	1998-07-28,09:25:37	uc

. . .

DEPT_VT

	DNAME	DNO	MANAGER_SSN	VST	VET	TST	TET
v12:	Research	5	888665555	1996-09-20	now	1996-09-15,14:52:12	1996-03-28,09:23:57
v13:	Research	5	888665555	1996-09-20	1997-03-31	1997-03-28,09:23:57	uc
v14:	Research	5	333445555	1997-04-01	now	1997-03-28,09:23:57	uc

. . .

Deletes and Inserts

To delete `v`, create a copy `v2` of `v`.

Set the `VET` to the proper end-time and the `TST` to the time of the transaction and the `TET` to `uc`.

Then simply set `TET` on `v` from `uc` to the time of the delete transaction.

To insert, create the tuple and set `TST` to the current time, and the `VST` to whatever the valid start time should be, and the `TET` to `uc` and the `VET` to `now`. Then insert the tuple.

Bitemporal Database Implementation and Other Issues

We could keep all the information is a single table, or we could keep the currently valid information in one table and the history a second.

Note that a history of all the changes are computed. So I can, one year later, knock Smith's salary up for the last year. And the record of me doing this, and the previous and now updated values of Smith's salary will be kept.

Bitemporal databases are termed *append only databases*.

We have not (yet) talked about complex queries over these histories.

Nor have we talked (yet) about some of the rather heavy integrity constraints that are implicit in this model. All the writers to the database must follow the rules.

Allen's Interval Algebra

The interval x is anchored at x^- and ends at x^+. $(x^- < x^+)$

The interval y is anchored at y^- and ends at y^+. $(y^- < y^+)$

Relation	Symbol	Example	Conditions
x before y y after x	$<$ $>$	XXX YYY	$x^+ < y^-$
x meets y y met-by x	m $m^\smile$	XXXX YYYY	$x^+ = y^-$
x overlaps y y overlaped-by x	o $o^\smile$	XXXX YYYY	$x^- < y^- < x^+$, $x^+ < y^+$
x during y y includes x	d $d^\smile$	XX YYYY	$x^- > y^-$, $x^+ < y^+$
x starts y y started-by x	s $s^\smile$	XXXX YYYYYYYY	$x^- = y^-$, $x^+ < y^+$
x finishes y y finished-by x	f $f^\smile$	XXXX YYYYYYYY	$x^+ = y^+$ $x^- > y^-$
x equals y	$\equiv$	XXXX YYYY	$x^- = y^-$, $x^+ = y^+$

The general problem of determining the satisfiability of a set of n temporal relation statements in Allen's Algebra is intractable.

It is polynomial under certain restrictions.

Refutation procedure enables deduction.

Temporal Elements

Require that the system is able to reason over *temporal elements*.

Unique representation of temporal elements.

Sequences of distinct durations.

This insures that the operations compose over a closed domain of temporal elements.

TSQL2

TSQL2 allows temporal tables to be specified in the DDL. The implementation of the temporal aspects are thus largely hidden from the user.

Time Series Data

Time series records data-values over a predetermined sequence of data points on a calendar.

Queries are often *temporal aggregation queries* over larger granularities than the time series calendar.

Traditionally such data has not been handled in databases. But this is changing.

Conclusion

We touched on simple valid time and transaction time databases.

We treated bitemporal databases

Things are in reality much more complex with respect to transactions over bitemporal databases.

Remember we only covered one transaction type.

We also mentioned querying temporal DBs, Allen's algebra, and we very briefly mentioned time series data.