Temporal databases

Youri Baeyens

July 25, 2011

1. Concepts
   - Non temporal databases
   - Valid time databases
   - Transaction time databases
   - Bi-temporal databases
2. Implementation
   - How to implement t-uple versioning in the DWH?
3. Readings

Non temporal relations

Id	Name	State	Capital
1	MobileTelCo	Active	1500
2	BankSpec	Active	1600

- Very common in relational models
- Each t-uple (record) represents a fact that is currently true. There is only one snapshot available: the "current snapshot".
- As time goes by, the content of the table changes. Some t-uples are added, some other are modified or deleted.
- With such a model, we are loosing information.

Valid time databases

Id	Name	State	Capital	dt_vld_strt	dt_vld_end
1	MobileTelCo	Active	1500	2004-12-15	2006-01-01
1	MobileTelCo	Active	1555	2006-01-01	2008-01-01
1	MobileTelCo	Active	1590	2008-01-01	9999-12-31
2	BankSpec	Active	1600	2008-02-15	2009-02-01
2	BankSpec	Bankrupt	1600	2009-02-01	9999-12-31

- Validity periods of t-uples are delimited by two instants: Dt_vldt_strt and Dt_vldt_end.
- Usually, the periods are delimited by [Dt_vldt_strt, Dt_vldt_end[.

Time of validity

- Valid time periods are managed by humans.
- Validity dates often have an "administrative" impact.
- It is possible to set a period "pro-actively" by using future dates.
- It is also possible to change the vision we have of the past, by correcting the timeline.
- All in all ... We can manage the timeline the way we want.

Time of validity

- Validity dates are managed by humans ... and humans can make mistakes. You could observe incoherences in timelines (overlaps, gaps)
- A good practice would be to enforce integrity constraints at database levels, to avoid incoherences (not so simple).
- Check your timelines !!!

Id	Name	State	Capital	dt_vld_strt	dt_vld_end
1	MobileTelCo	Active	1500	2004-12-15	2006-01-15
1	MobileTelCo	Active	1555	2006-01-01	2008-01-01
1	MobileTelCo	Active	1590	2008-01-01	9999-12-31
2	BankSpec	Active	1600	2008-02-15	2009-02-01
2	BankSpec	Bankrupt	1600	2010-02-15	9999-12-31

Time of validity

Exercise

What was "officialy" the capital of BankSpec on 2008-04-03?

```
proc sql;
 select Capital
 from LU_SDS_EMPL.TU_SDS_EMPL
 where Name='BankSpec'
       and dt_vldt_strt<='03MAR2008'D<dt_vldt_end;
quit;
```

Transaction time databases

- Transaction time databases can be considered as a common non-temporal database that supports complete backup of changes.
- We say that transaction time tables are "archives".

Transaction time databases

- This concept is closely related to the concept of "T-uple versioning".
- The systematic archiving of all changes make it possible to "rollback", "rebuild" the non-temporal database to any point in time.
- You can address questions like "What was the non-temporal database saying on 1st january 2008?"

T-uple versioning

- T-uple versioning (also called point-in-time) is a mechanism used to store past states of a non-temporal table.
- It makes possible to retrieve the state of t-uples at a given time. In other words, it makes it possible to retrieve "snapshots".
- Very interesting for statisticians: it makes it possible to reproduce your results.
- For example: to be able to find back the population of statistical units that were used to create a sample, you only need to remember the snapshot date !

Transaction periods

- A period is associated to each version of the record in the archive.
- The period consist in a transaction start date (dt_trn_strt) and transaction end time (dt_trn_end) indicating when the t-uple version appeared in the database (new or modified version of a t-uple) and when it disappeared (deleted or replaced by another version).
- Versions are sequenced. As a consequence, the end time of a record (the old version of the t-uple) often correspond to the start time of another record (the new version of the t-uple).
- Transaction time periods are delimited by [Dt_trn_strt, Dt_trn_end[.

T-uple versioning

- Transaction time "resemble" valid time databases but ...
- ... the semantic of valid time differs from transaction time

Id	Name	State	Capital	dt_trn_strt	dt_trn_end
1	MobileTelCo	Active	1500	2004-12-15	2006-01-01
1	MobileTelCo	Active	1555	2006-01-01	2008-01-01
1	MobileTelCo	Active	1590	2008-01-01	9999-12-31
2	BankSpec	Active	1600	2006-02-15	2008-02-01
2	BankSpec	Bankrupt	1600	2008-02-01	9999-12-31

Current values

- Current t-uples have a transaction end time set to "true until changed" (UC).
- "True until changed" does not exist neither in DB2 nor in SAS.
- Our department uses 31/12/9999 (in DB2 and SAS) as a surrogate for "true until changed".

Properties

- Transaction time is managed by "programs", not by humans.
- We can easily trace the different version of a t-uple, looking at its timeline.
- The timeline of a t-uple is always correct as it is managed by "programs". There are no incoherences like gaps or overlaps.
- For the computer scientists: Code snippets, SAS/DI transfomations (form of "slowly changing dimension"), ... exist that ease the implementation of t-uple versioning.

Semantics

Transaction dates

- You never correct transaction time dates!! It doesn't make sense.
- A transaction date is bounded by the database creation date.
- A transaction date cannot be a "date in the future".

Valid time dates

- It makes sense to correct valid time dates (administrative changes).
- A valid time date is bounded by the big bang.
- A valid time date is bounded by the big crunch.

Retrieving a snapshot

Exercise

Retrieve the t-uples one could see in the database on 2008-04-03?

```
proc sql;
 select Capital
 from LU_SDS_EMPL.TU_SDS_EMPL
 where dt_trn_strt<='03MAR2008'D<dt_trn_end;
quit;
```

Bi-temporal databases

- Bi-temporal databases are valid time databases with full t-uple versioning.
- Bi-temporal databases combines valid time and transaction time (for t-uple versioning); they make it possible to retrieve validity timelines at any point in time.

Example of Bi-temporal table (extract)

Id	Name	State	Capital	dt_vldt_strt	dt_vldt_end	dt_trn_strt	dt_trn_end
5	Leo	Active	1600	2004-12-15	9999-12-31	2004-12-20	2006-12-01
5	Leo	Active	1600	2004-12-15	2006-12-15	2006-12-01	9999-12-31
5	Leo	Active	1650	2006-12-15	9999-12-31	2006-12-01	9999-12-31
8	FredAndCo	Active	1600	2004-12-15	9999-12-31	2004-12-20	2006-12-01
8	FredAndCo	Active	1600	2004-12-15	2006-12-15	2006-12-01	9999-12-31
8	FredAndCo	Active	1650	2006-12-15	9999-12-31	2006-12-01	2008-12-01
8	FredAndCo	Active	1650	2006-12-15	2008-12-15	2008-12-01	9999-12-31
8	FredAndCo	Active	1700	2008-12-15	9999-12-31	2008-12-01	9999-12-31

How to use a bi-temporal database?

1. Select first a snapshot of the valid time database.
2. Take a look at the valid time line you get.

Example of Bi-temporal table (extract)

Id	Name	State	Capital	dt_vldt_strt	dt_vldt_end	dt_trn_strt	dt_trn_end
8	FredAndCo	Active	1600	2004-12-15	9999-12-31	2004-12-20	2006-12-01
8	FredAndCo	Active	1600	2004-12-15	2006-12-15	2006-12-01	9999-12-31
8	FredAndCo	Active	1650	2006-12-15	9999-12-31	2006-12-01	2008-12-01
8	FredAndCo	Active	1650	2006-12-15	2008-12-15	2008-12-01	9999-12-31
8	FredAndCo	Active	1700	2008-12-15	9999-12-31	2008-12-01	9999-12-31

Considering the knwoledge we had on 2008-11-15, what salary could Fred was for 2006-12-31?

```
/* Get the snapshot of 2008-11-15 for Fred*/
proc sql;
create table T_snapshot_20081115 as
select *
from LU_SDS_EMPL.TU_SDS_EMPL_ARC
where name="Fred"
      and dt_trn_strt<='15NOV2008'D<dt_trn_end;
quit;

/* Take a look at the valid time line */
proc print data=T_snpshot_20081115 noobs;
var Capital dt_vldt_strt dt_vldt_end;
run;
```

Answer to exercise

Name	State	Capital	dt_vldt_strt	dt_vldt_end
Fred	Active	1600	2004-12-15	2006-12-15
Fred	Active	1650	2006-12-15	9999-12-31

On 2008-11-15, we knew that Fred earned 1650 euros as from 2006-12-15.

It is also possible to answer the question in one step

```
/* Get the snapshot of 2008-11-15 for Fred and take a look at its official salary on 2006-12-31*/
proc sql;
select "Fred's salary was: ", Capital
from LU_SDS_EMPL.TU_SDS_EMPL_ARC
where name="Fred"
      and dt_trn_strt<='15NOV2008'D<dt_trn_end
      and dt_vldt_strt<='31DEC2006'D<dt_vldt_end;
quit;
```

Tables

- Let T = a table for which we want t-uple versioning
- Let T_arc = the archived version of T

Tables

In a data-warehouse context, T will be typically be refreshed "at night". T_arc will be refreshed just after T. It is possible to retrieve the "previous" version of T from the archive. This version will compared with the "current" version of T.

- Let $Snp.LastKnown = \{tuple \in T\_arc | dt\_trn\_end = UC\}$
- Let $Snp.Current = \{tuple \in T\}$
- Let $Altered = \{tuples \in Snp.LastKnown \setminus Snp.Current\}$
- Let $New = \{tuples \in Snp.Current \setminus Snp.LastKnown\}$

Macro

```
data T01_class;
set sashelp.class;
where name>'M';
run;
```

Project Designer | T01_CLASS (read-only)

	Name	Sex	Age	Height	Weight
1	Mary	F	15	66.5	112
2	Philip	M	17	72	150
3	Robert	M	12	64.8	128
4	Ronald	M	15	67	133
5	William	M	15	66.5	112
6	Maurice	M	19	70	115

Macro

```
%let variables=name,sex,age,height,weight;

data T02_class_fp / view=T02_class_fp;
length TX_FINGERPRINT $32;
set T01_class;
TX_FINGERPRINT=put(md5(catx("|",&variables)),hex32.);
run;
```

Project Designer | T02_CLASS_FP (read-only)

	TX_FINGERPRINT	Name	Sex	Age	Height	Weight
1	09ED12DC35159AFFD	Mary	F	15	66.5	112
2	E9592BED9DE79E4F0	Philip	M	17	72	150
3	0DF47829CDDAFA1FE	Robert	M	12	64.8	128
4	3A70056E534778BDEF	Ronald	M	15	67	133
5	6A034EC9BBE1DB07B	William	M	15	66.5	112
6	F3F3F372F10E9431D0	Maurice	M	19	70	115

Archive creation

Archive creation

The structure of the snapshot and archive tables are the same, if we except the presence of transaction dates.

```
/* DB creation */

proc sql;
create table T10_class_arc like T01_class;
alter table T10_class_arc
  add DT_TRN_STRT     num format=datetime.
  add DT_TRN_STOP     num format=datetime.
  add TX_FINGERPRINT char(32);
quit;
```

Macro

```
/* Initial load */

proc sql;
insert into T10_class_arc(&variables,
                          DT_TRN_STRT,
                          DT_TRN_STOP,
                          TX_FINGERPRINT )
select &variables,
       datetime(),
       "31DEC9999:00:00:00"DT,
       TX_FINGERPRINT
from T02_class_fp;
quit;
```

Project Designer | T10_CLASS_ARC (read-only)

	Name	Sex	Age	Height	Weight	DT_TRN_STRT	DT_TRN_STOP	TX_FINGERPRINT
1	Mary	F	15	66.5	112	25JUL11:16:01:26	31DEC99:00:00:00	09ED12DC35159AFFD
2	Philip	M	16	72	150	25JUL11:16:01:26	31DEC99:00:00:00	A4B51359FEE1F8A931
3	Robert	M	12	64.8	128	25JUL11:16:01:26	31DEC99:00:00:00	0DF47829CDDAFA1FE
4	Ronald	M	15	67	133	25JUL11:16:01:26	31DEC99:00:00:00	3A70056E534778BDEF
5	Thomas	M	11	57.5	85	25JUL11:16:01:26	31DEC99:00:00:00	6B143D502A47A22D49
6	William	M	15	66.5	112	25JUL11:16:01:26	31DEC99:00:00:00	6A034EC9BBE1DB07B

Macro

```
%macro archive_maintenance(archive=,current=);
%let maintenance_start_time=%sysfunc(datetime());
proc sql;
  /* Closures */
  update &archive
  set dt_trn_stop=&maintenance_start_time
  where tx_fingerprint not in ( select distinct tx_fingerprint from &current )
        and dt_trn_stop="31DEC9999:00:00:00"DT;

  /* Inserts */
  create table T20_new_tuples as
  select * from &current
  where tx_fingerprint not in ( select distinct tx_fingerprint
                                from &archive
                                where dt_trn_stop="31DEC9999:00:00:00"DT );
  insert into &archive(&variables, dt_trn_strt ,dt_trn_stop, tx_fingerprint)
  select &variables, &maintenance_start_time, "31DEC9999:00:00:00"DT, tx_fingerprint
  from T20_new_tuples;
  drop table T20_new_tuples;
quit;
%mend;
```

Macro

```
proc sql;
/* A t-uple delete */
delete from T01_class where name="Thomas";

/* A t-uple update */
update T01_class set age=17 where name="Philip";

/* A t-uple creation */
insert into T01_class(name,sex,age,height,weight) values ("Maurice","M",19,70,115);
quit;

%archive_maintenance(archive=T10_CLASS_ARC,current=T02_CLASS_FP);
```

Project Designer | T10_CLASS_ARC (read-only)

	Name	Sex	Age	Height	Weight	DT_TRN_STRT	DT_TRN_STOP	TX_FINGERPRINT
1	Mary	F	15	66.5	112	25JUL11:15:46:40	31DEC99:00:00:00	09ED12DC35159AFFD
2	Philip	M	16	72	150	25JUL11:15:46:40	25JUL11:15:47:12	A4B51359FEE1F8A931
3	Robert	M	12	64.8	128	25JUL11:15:46:40	31DEC99:00:00:00	0DF47829CDDAFA1FE
4	Ronald	M	15	67	133	25JUL11:15:46:40	31DEC99:00:00:00	3A70056E534778BDEF
5	Thomas	M	11	57.5	85	25JUL11:15:46:40	25JUL11:15:47:12	6B143D502A47A22D49
6	William	M	15	66.5	112	25JUL11:15:46:40	31DEC99:00:00:00	6A034EC9BBE1DB07B
7	Philip	M	17	72	150	25JUL11:15:47:12	31DEC99:00:00:00	E9592BED9DE79E4F0
8	Maurice	M	19	70	115	25JUL11:15:47:12	31DEC99:00:00:00	F3F3F372F10E9431D0

Web site

Web site of Richard T. Snodgrass

http://www.cs.arizona.edu

Many articles, books, code snippets, ... for free. All below mentionned publications are available on the site. Ask Youri for SAS code samples.

References I

Christian Jensen, Curtis Dyreson, Michael Böhlen, James Clifford, Ramez Elmasri, Shas hi Gadia, Fabio Grandi, Pat Hayes, Sushil Jajodia, Wolfgang Käfer, Nick Kline, Nikos Lorentzos, Yannis Mitsopoul os, Angelo Montanari, Daniel Nonen, Elisa Peressi, Barbara Pernici, John Roddick, Maria Sarda, Nandlal an d Scalas, Arie Segev, Richard Snodgrass, Mike Soo, Abdullah Tansel, Paolo Tiberio, and Gio Wiederhold.
The consensus glossary of temporal database concepts — february 1998 version.
Temporal Databases: Research and Practice, pages 367–405, 1998.

References II

Richard T. Snodgrass.
Managing temporal data: A five-part series.
TimeCenter TR-28, September, 1998.

Richard Thomas Snodgrass.
Developing time-oriented database applications in SQL.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.