tDB2TemporalMergeELT

Purpose

http://www.cimt-ag.de

This component carries out the creation and fill procedure for Temporal Tables in the IBM DB2.
Temporal Tables are introduced with DB2 v10 and provides mechanism to keep historical data in two time possible time dimensions:
1. Business Time period is the time range within the data a valid from a business perspective. The business time period is optional.
2. System Time period is the time range, which reflects the technical creation and modification of the data. The system time period will be automatically defined by timestamps of the database management system and is mandatory.
If a table use both time periods it's called bi-temporal table.
To get a better idea about this concept please refer to this web resource:

http://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/

Talend-Integration

This component can be found in the palette under Database -> DB2 and
Business Intelligence -> DB SCD.

Parameters

Connection configuration:

Property	Content
Property Type	Choose the database connection from the Metadata or use the build-in mode to setup individual configurations. (Only if use want to establish a dedicated connection for this component)
Use existing connections	True: choose an existing connection component in your job False: configure its own connection.
Host	Host (IP address or hostname) of your DB2 server. ***Required***
Port	Port where the instance is listening. Default is 50000.
Additional JDBC Parameters	Set here semicolon separated list of key=value pairs with JDBC parameters. The default pair in this component is: retrieveMessagesFromServerOnGetMessage=true which cause in case of errors a readable error message instead of getting only the SQLCODE.
Database	The database you want to work with
Source DB Schema	The database schema containing the source table.
Username	User name
Password	Password of the user

Source configuration:

Property	Content
Source table	Name (without schema) of the source table. The name will be set automatically if you drag and drop the component from a metadata table.
Where condition	The condition to select the source datasets. The keyword *where* is not needed here (it will be added in the SQL code generation in case of this attribute is not empty).
Schema from input table	Talend Schema chooser for the input table.
Use self defined source key	The component can take the keys from the input schema. In case of there are no keys defined in this schema (e.g. the source table is probably a view) it is possible to define the keys which are part of the source key. It is necessary to have at least one input field declared as source key. Source key field will not update and will be used to find a single unique dataset.
Track updates for all columns	If this option is checked, all datasets which source key already exists in the target table causes an update and a versioning of the previous dataset (regardless if it was a real change or not). It is recommended using this option if the source provides only changed datasets (e.g. with a source selection by last modified timestamps).
Supplementary Source Column Config	Per source column you can specify: *Ignore:* If true the column will be ignored. This is helpful if you prefer using a schema from the repository and it contains not needed columns. *Track real changes for*: This cause the component to track real changes of this field. Visible only if the option "Track updates for all columns" is switched off. *Is source key*: Check all columns, which identifies a unique source dataset. Visible only of option "Use self defined source key" is enabled.

Target configuration:

Property	Content
Target DB Schema	Database schema for the target table.
Target table	Name (without schema) of the target table.
Create table if not exists	If the table (and its historical counterpart) does not exist, it will be created.
Create indexes for source key, bus-tp, sys-tp	In addition to the creation of the table, also indexes on the source keys, on the business time period columns (if needed) and on the system time period columns for the target table will be created.
Use a specific table space	If the target table should created within a specific table space, this table space will be created if needed in case of the table have to be created.
Tablespace name	Name of the table space in which the target table should be created.
Surrogate Key column	If set, a surrogate key will be added to the target table and configured as auto increment column with type BIGINT.
Use Business Timeperiod	If this option is switch on, the table and fill method a carry out a bi-temporal versioning of data.
Business Timeperiod is of data type Date	If switched on the business time period columns will work with days and ignores the time (set the time to 00:00:00)
Business Timeperiod Start Column	Name of the column holding the start date/timestamp of the business time period. Typically in data warehouse projects it is needed to follow project specific naming conventions for those kinds of fields.
Business Timeperiod End Column	Name of the column holding the end date/timestamp of the business time period. Typically in data warehouse projects it is needed to follow project specific naming conventions for those kinds of fields.
Value Start BUS time	The value (as literal or a context variable) which defines the start of the business time period (timestamp or day). The value has to be of a data type inheriting java.util.Date.
Value End BUS time (insert)	The value (as literal or a context variable), which defines the end of the business, time period (timestamp or day) in case of insert a new data set. The value has to be of a data type inheriting java.util.Date.

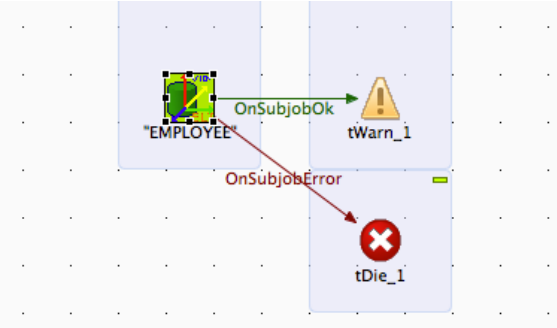
	In case of days, this has typically 2 different values depending the situation: The value should point to a date in the far future (like a SCD end value to specify the current valid dataset).
Value End BUS time (update)	The value (as literal or a context variable), which defines the end of the business, time period (timestamp or day) in case of update an existing data set. The value has to be of a data type inheriting java.util.Date. In case of days, this has typically 2 different values depending the situation: The value should contains the next time slice (e.g. the next day) after the start value of the business time period.
System Timeperiod Start Column	Name of the column for the system time period start.
System Timeperiod End Column	Name of the column for the system time period end.
Set history table append on	With this option the history table can be set as append on. This option increases the performance in case of many updates.
Switch Off Versioning at the End	Especially in data ware house applications it is a common task to update foreign key fields in the target table or converting / transforming special values. To avoid creating historical datasets by this post processing it is possible to switch off the versioning feature at the end. Every new run of this component will switch on the versioning in case of it is switched off.
Additional Columns in Target Table	If additional columns are needed (e.g. meta data columns like job instance keys, processing information and so on) it is possible to specify them here and they values. As values can be used literals or context variables matching the data type of the added column. It is possible to enable the column for the insert or update statement.
Delete condition	If the source data contains info to mark the data for delete you can specify the condition here. It will cause an additional part of the merge statement to delete if the dataset exists and the delete condition is true.

Return values

Return value	Content
ERROR_MESSAGE	Last error message
COUNT_MERGED	Number of datasets actually merged by the component

Scenario: Bi-temporal processing of the EMPLOYEE table from the DB2 examples

The component creates its own database connection and carries out a bi-temporal versioning.



Connection Configuration

Property Type: Repository — DB (IBM_DB2):DB2_SAMPLES

Use an existing connection

Host: "on-0337-jll.local" Port: "50001"

Additional JDBC Parameters: "retrieveMessagesFromServerOnGetMessage=true"

Database: "SAMPLE" Source DB Schema: "DB2INST2"

Username: "db2inst2" Password: **********

Source Configuration

Source Table: "EMPLOYEE"

Where Condition (without keyword where): "SEX='M'"

Schema from Input Table: Repository — DB (IBM_DB2):DB2_SAMPLES - EMPLOYEE — Edit schema

Use self defined source key | Track updates for all columns (except for source keys)

Supplementary Source Column Config

Column	Ignore Column	Track real changes for	Is source key
EMPNO	☐	☐	☑
FIRSTNME	☐	☑	☐
MIDINIT	☐	☑	☐
LASTNAME	☐	☑	☐
WORKDEPT	☐	☑	☐
PHONENO	☐	☑	☐
HIREDATE	☐	☑	☐
JOB	☐	☐	☐
EDLEVEL	☐	☐	☐
SEX	☐	☐	☐
BIRTHDATE	☐	☑	☐
SALARY	☐	☑	☐
BONUS	☐	☑	☐

Target Configuration

Target DB Schema: "DWH_ODS" Target Table: "EMPLOYEE" Alternative History Table: ""

Create table if not exists | Create indexes for src-key, bus-tp, sys-tp | Create primary key

Use specific tablespace Tablespace name (max. 18 characters): "TBLSP_EMPLOYEE"

Surrogate Key: "SKEY"

Use Business Timeperiod | Business Timeperiod is of Date type

Business Timeperiod Start Column: "DWH_BUS_VALID_FROM" Value Start BUS Time: context.reportDate

Business Timeperiod End Column: "DWH_BUS_VALID_TO"

Value End BUS Time (insert): "'9999-12-30'" Value End BUS Time (update): "'9999-12-30'"

System Timeperiod Start Column: "DWH_SYS_VALID_FROM" System Timeperiod End Column: "DWH_SYS_VALID_TO"

Addition columns in target table

Name	Type	Length	Precision	Nullable	For Insert	For Update	Value
"JOB_INSTANCE_ID	BIGINT	20	0	☑	☑	☑	context.jobInstanc

Delete condition: "s.JOB is null"

The source table has the alias s and the target table has the alias t

Check and switch ON versioning at start | Switch OFF versioning at the end

Statements created by the component

Create target tables and indices

```
-- create tablespace
create tablespace TBLSP_EMPLOYEE;

-- create table
create table DWH_ODS.EMPLOYEE (
    TRANS_START TIMESTAMP(12),
    DWH_BUS_VALID_FROM DATE NOT NULL,
    DWH_BUS_VALID_TO DATE NOT NULL,
    DWH_SYS_VALID_FROM TIMESTAMP(12) NOT NULL,
    DWH_SYS_VALID_TO TIMESTAMP(12) NOT NULL,
    SKEY BIGINT NOT NULL GENERATED BY DEFAULT AS IDENTITY (START WITH 1 INCREMENT BY 1),
    EMPNO CHAR(6) NOT NULL,
    FIRSTNME VARCHAR(12) NOT NULL,
    MIDINIT CHAR(1),
    LASTNAME VARCHAR(15) NOT NULL,
    WORKDEPT CHAR(3),
    PHONENO CHAR(4),
    HIREDATE DATE,
    JOB CHAR(8),
    EDLEVEL SMALLINT NOT NULL,
    SEX CHAR(1),
    BIRTHDATE DATE,
    SALARY DECIMAL(9,2),
    BONUS DECIMAL(9,2),
    COMM DECIMAL(9,2),
    JOB_INSTANCE_ID BIGINT
)
in TBLSP_EMPLOYEE;

-- setup table as temporal table
alter table DWH_ODS.EMPLOYEE alter column TRANS_START set GENERATED ALWAYS AS TRANSACTION START ID;
alter table DWH_ODS.EMPLOYEE alter column TRANS_START set IMPLICITLY HIDDEN;
alter table DWH_ODS.EMPLOYEE alter column DWH_SYS_VALID_FROM set GENERATED ALWAYS AS ROW BEGIN;
alter table DWH_ODS.EMPLOYEE alter column DWH_SYS_VALID_TO set GENERATED ALWAYS AS ROW END;
alter table DWH_ODS.EMPLOYEE add PERIOD SYSTEM_TIME (DWH_SYS_VALID_FROM,DWH_SYS_VALID_TO);
alter table DWH_ODS.EMPLOYEE add PERIOD BUSINESS_TIME (DWH_BUS_VALID_FROM,DWH_BUS_VALID_TO);
alter table DWH_ODS.EMPLOYEE add constraint EMPLOYEE_pk primary key (EMPNO, BUSINESS_TIME WITHOUT
OVERLAPS);

-- create index for source key
create index DWH_ODS.EMPLOYEE_src_pk on DWH_ODS.EMPLOYEE(EMPNO);
-- create index for business time period
create index DWH_ODS.EMPLOYEE_bustp on DWH_ODS.EMPLOYEE(DWH_BUS_VALID_FROM,DWH_BUS_VALID_TO);
-- create index for system time period
create index DWH_ODS.EMPLOYEE_systp on DWH_ODS.EMPLOYEE(DWH_SYS_VALID_FROM,DWH_SYS_VALID_TO);
-- create history table
create table DWH_ODS.EMPLOYEE_HIST like DWH_ODS.EMPLOYEE in TBLSP_EMPLOYEE;
alter table DWH_ODS.EMPLOYEE_HIST append on;

-- switch on versioning
alter table DWH_ODS.EMPLOYEE add versioning use history table DWH_ODS.EMPLOYEE_HIST;
```

Statement to switch on the versioning if it was switch off

```
-- switch on versioning
ALTER TABLE DWH_ODS.EMPLOYEE ADD VERSIONING USE HISTORY TABLE DWH_ODS.EMPLOYEE_HIST
```

After the completing the processing (after the merge statement) and if the option to switch off the versioning at the end is on:

```
-- switch off versioning
ALTER TABLE DWH_ODS.EMPLOYEE DROP VERSIONING
```

Merge statement

```
-- merge
merge into DWH_ODS.EMPLOYEE t
using (
    select * from DB2INST2.EMPLOYEE where SEX='M'
) s
on (t.EMPNO = s.EMPNO and
    t.DWH_BUS_VALID_TO > ?/*#1 business_time_start */
)
when not matched and not (s.JOB is null) then
insert (
  DWH_BUS_VALID_FROM,
  DWH_BUS_VALID_TO,
  EMPNO,
  FIRSTNME,
  MIDINIT,
  LASTNAME,
  WORKDEPT,
  PHONENO,
  HIREDATE,
  JOB,
  EDLEVEL,
  SEX,
  BIRTHDATE,
  SALARY,
  BONUS,
  COMM,
  JOB_INSTANCE_ID)
values (
  ?/*#2 business_time_start */,
  '9999-12-30',
  s.EMPNO,
  s.FIRSTNME,
  s.MIDINIT,
  s.LASTNAME,
  s.WORKDEPT,
  s.PHONENO,
  s.HIREDATE,
  s.JOB,
  s.EDLEVEL,
  s.SEX,
  s.BIRTHDATE,
  s.SALARY,
  s.BONUS,
  s.COMM,
  ?/*#3 JOB_INSTANCE_ID*/)
when matched and (
    (t.FIRSTNME <> s.FIRSTNME)
    or (t.MIDINIT <> s.MIDINIT)
    or (t.LASTNAME <> s.LASTNAME)
    or (t.WORKDEPT <> s.WORKDEPT)
    or (t.PHONENO <> s.PHONENO)
    or (t.HIREDATE <> s.HIREDATE)
    or (t.BIRTHDATE <> s.BIRTHDATE)
    or (t.SALARY <> s.SALARY)
    or (t.BONUS <> s.BONUS)
    or (t.COMM <> s.COMM)
) and not (s.JOB is null) then
update for portion of business_time
      from ?/*#4 business_time_start */
      to '9999-12-30'
set t.FIRSTNME = s.FIRSTNME,
    t.MIDINIT = s.MIDINIT,
    t.LASTNAME = s.LASTNAME,
    t.WORKDEPT = s.WORKDEPT,
    t.PHONENO = s.PHONENO,
    t.HIREDATE = s.HIREDATE,
    t.JOB = s.JOB,
    t.EDLEVEL = s.EDLEVEL,
    t.SEX = s.SEX,
    t.BIRTHDATE = s.BIRTHDATE,
    t.SALARY = s.SALARY,
    t.BONUS = s.BONUS,
    t.COMM = s.COMM,
    t.JOB_INSTANCE_ID = ?/*#5 JOB_INSTANCE_ID*/
when matched and (s.JOB is null) then
delete for portion of business_time
      from ?/*#6 business_time_start */
      to '9999-12-30'
```