Time slices

Time slices in archive

Starting with D2000 version 8.0, the support for time slices for all historical values has been implemented in the process D2000 Archiv.

Starting with D2000 version V10.1.39 (patches from June 22th, 2016 and newer) the support for time slices for structured historical values only has been implemented in the process D2000 Archiv (i.e. simple historical values work in the original way).
Note: this mode was implemented due to the properties of Oracle and PostgreSQL databases. A large amount of small tables causes significant memory usage for Oracle 11g (and possibly ORA-600 errors) and a large amount of small files on disk for PostgreSQL.

If time slices are not activated the archive works as before:

- data of every Historical value object (simple historical value, one-column structured historical value or structured historical value) are stored in one table (e.g. for simple historical value with HOBJ=155 it is table DT0000155, for one-column historical value table DT0000155R, for structured historical value table DT0000155RC),
- data from the archive table older than history depth are periodically deleted and optionally automatic reorganization of the table is performed.

The performance problem appeared in structured historical values. Their database tables could grow beyond expectations (up to several GB on production systems) and their reorganization took several dozen minutes and significantly burdened both the processor and the disks.
The solution to these problems is "slicing" the tables to time slices (for now with constant size of 30 days). I.e. every 30 days a new database table is created for every Historical value object (e.g. for simple historical value with HOBJ=155 tables DT0000155_1, DT0000155_2, DT0000155_3 etc will be created). The original database table (e.g. DT0000155) will be used for storing data older than the time of conversion of archive database and we call it a **slice zero**.
The implementation of time slices has enabled:

- Elimination of periodic deleting (implemented by SQL command DELETE) and instead it, empty the entire time slice, when it is beyond the configured history depth (by SQL command TRUNCATE TABLE, which is substantially faster).
  Note: Periodic deleting can be still optionally turned on by the archive parameter DeleteInSlices.
- Reduction of table reorganizations - reorganizations, caused by periodic deleting of data beyond the configured history depth, will be eliminated. There will only remain reorganizations caused by data deleting during recalcs of calculated or statistic Historical value objects (automatically after the primary historical values had been changed or as a result of tell command RECALC).
- Reduce time needed for reorganization - as time slices are reorganized (containing 30 days of data) instead of tables with whole history depth worth of data.

Enabling time slices mode and archive database conversion

The transition to time slices mode will be caused by the change of configuration parameter DataTableSlices to value 1 (time slices for all historical value) resp. to value 2 (time slices for structured historical values only).
During archive database conversion, the archive performs these steps:

- Value 'DATA_TABLE_SLICES' (if DataTableSlices=1) or 'DATA_TS_STRUCT' (if DataTableSlices=2) with current timestamp will be written to the table *LOG_DATA.*
- A new table *ARC_SLICE* will be created. It will contain information about time slices. The table contains columns:
  - **ID** - HOBJ of the Historical value object
  - **TAB_TYPE** - type of object (simple historical value, one-column historical value or structured historical value)
  - **SLICE_ID** - number of time slice
  - **FROM_TIME** - start time of time slice
  - **TO_TIME** - end time of time slice
  - **REORG_TIME** - time of the last reorganization of time slice (used for reorganization)
  - **DELETED_ROWS** - number of deleted rows (used for reorganization)
- For every Historical value object (resp. for every structured historical value object if DataTableSlices=2) the time slice number 1 will be created, e. g. DT0000155_1 (simple historical value) or DT0000155R_1 (one-column historical value) or DT0000155RC_1 (structured historical value). Time slice will be created for the nearest future time (multiple of 30 days starting with 1/1/1972).
- Information about the time slices 0 (slice zero) and 1 will be stored in database table *ARC_SLICE*. For slice zero the time of the last reorganization and number of deleted rows are queried from table *DELETED_ROWS*, and inserted to table *ARC_SLICE*, so that these values are not lost during the conversion.
- If the Historical value object is an archive filled from script (Value storage), all data from slice zero newer than start time of the 1st slice will be moved to the 1st slice. If the data are newer than the end time of the 1st slice, the 2nd slice (and possibly others) will be dynamically created and information about them will be stored in *ARC_SLICE* table.
- After the conversion of all tables, the value 'SLICES_CONVERTED' with current timestamp will be written to the table *LOG_DATA*.

Note: Conversion of archive database is non-reversible!
Therefore we recommend you to backup the archive database before changing parameter DataTableSlices. Database backup + data from converted database can be used (with the help of arcsynchro) to return to non-sliced archive.

Archive behaviour in time slices mode

After the time slices have been enabled, the archive keeps writing the data to *slice zero* (for up to 30 days), as the time slice 1 was created with the start time in future. Afterwards the data with the timestamps inside time slice 1 start coming and they will be stored in slice 1. In time equal to the middle of slice 1 (i.e. start time of slice + 15 days) requests for creation of next slices will be generated. These low priority requests are performed by archive. The time slice for a Historical value object will only be created if at least one value has been archived during the archive's work.
If the value with timestamp, for which there is no time slice, arrives in the archive, the slice will be created immediately and the value will be inserted into it.

During the periodic deleting, there is possible to control deleting in the original table - slice zero (parameter DeleteInSlice0) and in all other slices (parameter DeleteInSlices). We recommend to keep the default values , i.e. DeleteInSlice0=1 (data in slice zero will be deleted) and DeleteInSlices=0 (data in all other slices won't be deleted, but the slice will be emptied via TRUNCATE TABLE when it is older than history depth).
The emptied time slice will be marked as free (start time will be set to "1.1.1972") and be recycled by archive as soon as a new time slice is needed for the Historical value object to which the free slice belongs.
Note: The archive supports only one free time slice for every Historical value object. If more free time slices have been created (e.g. after reducing the history depth), all free slices except the first one will be deleted (from the table *ARC_SLICE* as well as their respective database tables), so that the archive database will not contain unusable free time slices.
Slice zero is an exception to previously mentioned rule and it will be preserved.

When reading historical values from the database, the data are obtained from one or more time slices (in a similar way as when reading from one or more d epository databases).

The list and details of time slices can be queried by the tell command SHOW_DYN_INFO.
Detailed information about the time slices (creation, deleting, recycling) will be displayed by archive when the debug category DBG.ARCHIV.SLICES is enabled either by the start parameter /E+DBG.ARCHIV.SLICES or via process D2000 System Console in the window Debug info.

Tell command REORGANIZE is extended to support time slices and it enables to reorganize the selected time slice, all time slices or the time slice for current time.

The utility arcsynchro is extended to support reading/writing from/to time slices and creating time slices during synchronization of archive databases. It can read from non-sliced/sliced database and write to non-sliced/sliced database.