Temporal Data in SQL Server

Dejan Sarka

Thanks to our sponsors:
Agenda

- Temporal databases
- SQL Server temporal tables
  - Creating
  - Querying
  - Altering and dropping
- Temporal tables problems

Temporal Databases

- Many relational databases show current state only
  - Constraints are valid in present, past and future time
  - E.g., supplier A is under a contract
- Temporal databases hold *time-stamped* propositions
  - *Since* predicate means ever since and not immediately before
  - E.g., supplier A is under a contract since a specific time point A
  - *During (or from ... to)* predicate means throughout and not immediately before or immediately after
  - E.g., supplier A is under a contract from time point A to time point B
- *Timeline* consists of discrete, indivisible time quanta
  - *Time quantum* is the smallest represented unit
  - Can represent time quanta with integers
Semi-Temporal Problems

- Example: suppliers, supplied parts

<table>
<thead>
<tr>
<th>S#</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>d04</td>
</tr>
<tr>
<td>S2</td>
<td>d07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S#</th>
<th>P#</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>P1</td>
<td>d04</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
<td>d05</td>
</tr>
<tr>
<td>S2</td>
<td>P1</td>
<td>d08</td>
</tr>
<tr>
<td>S2</td>
<td>P2</td>
<td>d09</td>
</tr>
</tbody>
</table>

- No problem with PKs and Fks
- Need additional constraint: no supplier can supply a part before the supplier is under contract

Full-Temporal Problems (1)

<table>
<thead>
<tr>
<th>S#</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>d04</td>
<td>d10</td>
</tr>
<tr>
<td>S2</td>
<td>d02</td>
<td>d04</td>
</tr>
<tr>
<td>S2</td>
<td>d07</td>
<td>d10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S#</th>
<th>P#</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>P1</td>
<td>d04</td>
<td>d10</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
<td>d05</td>
<td>d09</td>
</tr>
<tr>
<td>S2</td>
<td>P1</td>
<td>d08</td>
<td>d10</td>
</tr>
<tr>
<td>S2</td>
<td>P2</td>
<td>d09</td>
<td>d10</td>
</tr>
<tr>
<td>S2</td>
<td>P2</td>
<td>d03</td>
<td>d03</td>
</tr>
</tbody>
</table>

- Two candidate keys in each table (BCNF)
- To should not be less than From
- Cannot end contract on one day and begin on the next day
  - No abutting
Full-Temporal Problems (2)

- No supplier can be under two distinct contracts on the same day
- No overlapping
- No supplies if not under contract
  - SP(S#, From) is not a FK!
  - Inclusion dependency:

\[ SP(S#, From, To) \subseteq S(S#, From, To) \]

Valid and Transaction Times

- Bi-temporal databases support valid and transaction times
  - Valid times are human times
  - Transaction times are database times
  - Valid times are updatable
  - Transaction times are not updatable
- In model, we should not need transaction times
  - Should be implemented in a RDBMS
  - Support for valid times would also be appreciated
SQL Server Temporal Tables

- System-versioned tables store a full history of data changes
  - A validity period is stored in two datetime2 columns – SysStartTime and SysEndTime
- All versioning is automatic, no change in the code needed
  - Can be added to existing tables
- History is stored in an associated table
  - Historical table can be named, or take system name
  - Must have the same schema as the current table
  - The history table is a rowstore table with a clustered index on the period columns

Considerations and Limitations (1)

- The SysStartTime and SysEndTime columns must use the datetime2 data type
- The table must have a primary key; the historical table cannot use constraints
- To name the history table, specify the schema and table names
- History table compression defaults to PAGE
- The history table must reside in the same database as the current table
- Cannot alter or drop the history table
- No triggers on history table
Considerations and Limitations (2)

- System-versioned tables are not compatible with FILETABLE or FILESTREAM
- Columns with a BLOB data type, such as varchar(max) or image, can result in high storage requirements
- INSERT and UPDATE statements cannot reference the SysStartTime or SysEndTime columns
- You cannot directly modify data in the history table
- You cannot truncate a system-versioned table
- Merge replication is not supported
- No CDC or CT on the history table

Querying Temporal Tables

- System-versioned tables can be queried using the new FOR SYSTEM_TIME clause in the FROM part of a query
- Use one of the following four sub-clauses:
  - AS OF <date_time>
  - FROM <start_date_time> TO <end_date_time>
  - BETWEEN <start_date_time> AND <end_date_time>
  - CONTAINED IN ( <start_date_time>, <end_date_time> )
  - ALL to return everything
Creating a Temporal Table (1)

```sql
CREATE TABLE dbo.Employee
(
    EmployeeID int NOT NULL PRIMARY KEY CLUSTERED,
    ManagerID int NULL,
    FirstName varchar(50) NOT NULL,
    LastName varchar(50) NOT NULL,
    SysStartTime datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
    SysEndTime datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime)
) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.EmployeeHistory));
```

Creating a Temporal Table (2)

- Can create your own history table
- Can define your own indexes
- Including columnstore indexes
- With the HIDDEN clause, you can hide the new PERIOD columns
- Avoid impacting on existing applications that are not designed to handle new columns (SELECT *)
Converting Existing Tables

- You can also alter an existing table and add attributes to it to convert it to a system-versioned temporal table
  - Add period columns
  - Define the SYSTEM_TIME period on them
  - Set the temporal attribute
  - You need to provide default constraints for both columns, since they must be non-nullable
  - You can use a date value from the past for the default constraint for the first period column; however, you cannot set values in the future

Migrating a Custom Solution

- You might already maintain history in a custom table
- Migrate to system-versioned temporal tables
  - Align the columns in both tables
  - Add two period columns and define the period
  - There must be no gaps between historical entries in the history table
  - Use the DATA_CONSISTENCY_CHECK = ON option to check that all rows in the history table are valid
Altering Temporal Tables

- When you use ALTER TABLE to add a new nullable column, change data type or remove an existing column, system will automatically perform the action against both current and history tables.
- When you add a non-nullable column, you need to provide a default constraint as well.
  - You should take in account that this operation is an offline operation in all editions of SQL Server except in the Enterprise Edition.
  - Adding LOB or BLOB columns will cause a mass update in both the current and the history table in all SQL Server editions!
- You can also use ALTER TABLE statement to add or remove the attribute HIDDEN to period columns or to remove it.

Altering Temporal Tables Limitations

- Adding an IDENTITY or computed column
- Adding a ROWGUIDCOL column or changing an existing column to it
- Adding a SPARSE column or changing an existing column to it, when history table is compressed.
- If you need to perform schema changes to a temporal table not supported in the ALTER statement, you have to set its SYSTEM_VERSIONING attribute to false, to convert the tables to non-temporal tables, perform the changes and then convert back to temporal table.
Dropping Temporal Tables

- You cannot drop a system-versioned temporal table
  - Both current and history tables are protected until the attribute SYSTEM_VERSIONING of the current table is set to ON
  - When you set it to OFF, both tables automatically become non-temporal tables and are fully independent of each other
  - Therefore, you can perform all operations against them that are allowed to you according to your permissions
  - You can also drop period if you want to convert a temporal table to non-temporal one

Miscellaneous (1)

- SQL Server 2017 adds retention policy
  - Enable on database level
  - Configure on table level
- System-versioned temporal tables are also supported for memory-optimized tables
  - You can assign or let SQL Server create a history table for your memory-optimized table
  - The history table must be a disk table
  - Recent history table is an internal memory-optimized table that handles changes in the current table synchronously and enables DMLs to be executed from natively compiled code
Miscellaneous (2)

- Multiple tables scenario
  - Manual queries – write them correctly!
- Can use a view that joins the tables and filter the view for specific system time
  - Works well for a specific time
  - Works ...khh... For system time all
  - All possible combinations, even incorrect ones!
- For system time all also filters rows with same valid from and valid to datetimes

Temporal Tables as Dimensions

- System-versioned tables can be out of the box solution for Type 2 SCD problem
  - Can simplify the ETL process
- However, SSAS does not support temporal tables yet
  - You are limited to query the data warehouse directly, with T-SQL queries
- System versioning works on row level
  - Might need column-level versioning
Q & A

· Questions?