## **What is a system-versioned temporal table?**

A system-versioned temporal table is a type of user table designed to keep a full history of data changes, allowing easy point-in-time analysis. This type of temporal table is referred to as a system-versioned temporal table because the period of validity for each row is managed by the system (that is, the database engine).

Every temporal table has two explicitly defined columns, each with a **datetime2** data type. These columns are referred to as period columns. These period columns are used exclusively by the system to record the period of validity for each row, whenever a row is modified. The main table that stores current data is referred to as the current table, or simply as the temporal table.

In addition to these period columns, a temporal table also contains a reference to another table with a mirrored schema, called the history table. The system uses the history table to automatically store the previous version of the row each time a row in the temporal table gets updated or deleted. During temporal table creation users can specify an existing history table (which must be schema compliant) or let the system create a default history table.

## **How does temporal work?**

System-versioning for a table is implemented as a pair of tables, a current table and a history table. Within each of these tables, two additional **datetime2** columns are used to define the period of validity for each row:

* Period start column: The system records the start time for the row in this column, typically denoted as the ValidFrom column.
* Period end column: The system records the end time for the row in this column, typically denoted as the ValidTo column.

The current table contains the current value for each row. The history table contains each previous value (the old version) for each row, if any, and the start time and end time for the period for which it was valid.

* **Table Creation**

here are some considerations and limitations to be aware of when working with temporal tables, due to the nature of system-versioning:

* A temporal table must have a primary key defined in order to correlate records between the current table and the history table, and the history table can't have a primary key defined.
* The SYSTEM\_TIME period columns used to record the ValidFrom and ValidTo values must be defined with a datatype of **datetime2**.

**Example 1: (With primary key, two datetime fields (validfrom. validto) and Period fields**

CREATE TABLE Employee

(

[EmployeeID] int NOT NULL PRIMARY KEY CLUSTERED

, [Name] nvarchar(100) NOT NULL

, [Position] varchar(100) NOT NULL

, [Department] varchar(100) NOT NULL

, [Address] nvarchar(1024) NOT NULL

, [AnnualSalary] decimal (10,2) NOT NULL

, [ValidFrom] datetime2 (2) GENERATED ALWAYS AS ROW START

, [ValidTo] datetime2 (2) GENERATED ALWAYS AS ROW END

, PERIOD FOR SYSTEM\_TIME (ValidFrom, ValidTo)

) WITH (SYSTEM\_VERSIONING = ON (HISTORY\_TABLE = dbo.EmployeeHistory));

**Example 2: (Invalid table structure, without period field)**

CREATE TABLE Employee

(

[EmployeeID] int NOT NULL PRIMARY KEY CLUSTERED

, [Name] nvarchar(100) NOT NULL

, [Position] varchar(100) NOT NULL

, [Department] varchar(100) NOT NULL

, [Address] nvarchar(1024) NOT NULL

, [AnnualSalary] decimal (10,2) NOT NULL

, [ValidFrom] datetime2 (2) GENERATED ALWAYS AS ROW START

, [ValidTo] datetime2 (2) GENERATED ALWAYS AS ROW END

)

WITH (SYSTEM\_VERSIONING = ON (HISTORY\_TABLE = dbo.EmployeeHistory));

\*\*\* This will generate exception for missing the Period field. Cannot create generated always column when SYSTEM\_TIME period is not defined.

**Example 3: (Without two date fields)**

CREATE TABLE Employee

(

[EmployeeID] int NOT NULL PRIMARY KEY CLUSTERED

, [Name] nvarchar(100) NOT NULL

, [Position] varchar(100) NOT NULL

, [Department] varchar(100) NOT NULL

, [Address] nvarchar(1024) NOT NULL

, [AnnualSalary] decimal (10,2) NOT NULL

, PERIOD FOR SYSTEM\_TIME (GETDATE, GETDATE)

) WITH (SYSTEM\_VERSIONING = ON (HISTORY\_TABLE = dbo.EmployeeHistory));

\*\*\* This will generate exception for missing the Period field. Temporal 'GENERATED ALWAYS AS ROW START' column definition missing.

**Example 4. (Without primary key field)**

CREATE TABLE Employee

(

[EmployeeID] int NOT NULL

, [Name] nvarchar(100) NOT NULL

, [Position] varchar(100) NOT NULL

, [Department] varchar(100) NOT NULL

, [Address] nvarchar(1024) NOT NULL

, [AnnualSalary] decimal (10,2) NOT NULL

, [ValidFrom] datetime2 (2) GENERATED ALWAYS AS ROW START

, [ValidTo] datetime2 (2) GENERATED ALWAYS AS ROW END

, PERIOD FOR SYSTEM\_TIME (ValidFrom, ValidTo)

)

WITH (SYSTEM\_VERSIONING = ON (HISTORY\_TABLE = dbo.EmployeeHistory));

\*\*\* This will generate exception for missing the Primary key. System versioned temporal table 'TestDB.dbo.Employee' must have primary key defined.

**Example 5. Enable Historical data (Temporal Table) on existing table.**

/\*

Turn ON system versioning in Employee table in two steps

(1) add new period columns (HIDDEN)

(2) create default history table

\*/

ALTER TABLE Employee

ADD

ValidFrom datetime2 (2) GENERATED ALWAYS AS ROW START HIDDEN

constraint DF\_ValidFrom DEFAULT DATEADD(second, -1, SYSUTCDATETIME())

, ValidTo datetime2 (2) GENERATED ALWAYS AS ROW END HIDDEN

constraint DF\_ValidTo DEFAULT '9999.12.31 23:59:59.99'

, PERIOD FOR SYSTEM\_TIME (ValidFrom, ValidTo);

ALTER TABLE Employee

SET (SYSTEM\_VERSIONING = ON (HISTORY\_TABLE = dbo.Employee\_History));

[**Creating a Memory-Optimized Table and a Natively Compiled Stored Procedure**](https://learn.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/creating-a-memory-optimized-table-and-a-natively-compiled-stored-procedure?view=sql-server-ver16)**.**

BEGIN

--If table is system-versioned, SYSTEM\_VERSIONING must be set to OFF first

IF ((SELECT temporal\_type FROM SYS.TABLES WHERE object\_id = OBJECT\_ID('dbo.ProductInventory', 'U')) = 2)

BEGIN

ALTER TABLE [dbo].[ProductInventory] SET (SYSTEM\_VERSIONING = OFF)

END

DROP TABLE IF EXISTS [dbo].[ProductInventory];

DROP TABLE IF EXISTS [dbo].[ProductInventoryHistory];

END

GO

CREATE TABLE [dbo].[ProductInventory]

(

ProductId int NOT NULL,

LocationID INT NOT NULL,

Quantity int NOT NULL CHECK (Quantity >=0),

ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL ,

ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL ,

PERIOD FOR SYSTEM\_TIME(ValidFrom, ValidTo),

--Primary key definition

CONSTRAINT PK\_ProductInventory PRIMARY KEY NONCLUSTERED (ProductId, LocationId)

)

WITH

(

MEMORY\_OPTIMIZED=ON,

SYSTEM\_VERSIONING = ON

(

HISTORY\_TABLE = [dbo].[ProductInventoryHistory],

DATA\_CONSISTENCY\_CHECK = ON

)

)

CREATE CLUSTERED COLUMNSTORE INDEX IX\_ProductInventoryHistory ON [ProductInventoryHistory]

WITH (DROP\_EXISTING = ON);

1. If the name of a history table is specified during history table creation, you must specify the schema and table name.
2. By default, the history table is PAGE compressed.
3. If current table is partitioned, the history table is created on default file group because partitioning configuration isn't replicated automatically from the current table to the history table.
4. Temporal and history tables can't be FileTable and can contain columns of any supported data type other than FILESTREAM, since FileTable and FILESTREAM allow data manipulation outside of SQL Server and thus system versioning can't be guaranteed.
5. A node or edge table can't be created as or altered to a temporal table.
6. While temporal tables support blob data types, such as **(n)varchar(max)**, **varbinary(max)**, **(n)text**, and **image**, they'll incur significant storage costs and have performance implications due to their size. As such, when designing your system, care should be taken when using these data types.
7. History table must be created in the same database as the current table. Temporal querying over linked servers isn't supported.
8. History table can't have constraints (primary key, foreign key, table or column constraints).
9. Indexed views aren't supported on top of temporal queries (queries that use FOR SYSTEM\_TIME clause).
10. Online option (WITH (ONLINE = ON) has no effect on ALTER TABLE ALTER COLUMN in a system-versioned temporal table. ALTER column isn't performed as an online operation, regardless of which value was specified for the ONLINE option.
11. INSERT and UPDATE statements can't reference the SYSTEM\_TIME period columns. Attempts to insert values directly into these columns will be blocked.
12. TRUNCATE TABLE isn't supported while SYSTEM\_VERSIONING is ON.
13. Direct modification of the data in a history table isn't permitted.
14. INSTEAD OF triggers aren't permitted on either the current or the history table to avoid invalidating the DML logic. AFTER triggers are permitted only on the current table. They're blocked on the history table to avoid invalidating the DML logic.
15. Usage of replication technologies is limited:
16. **Availability groups:** Fully supported
17. **Change data capture and change tracking:** Supported only on the current table
18. **Snapshot and transactional replication**: Only supported for a single publisher without temporal being enabled, and *one* subscriber with temporal enabled. Use of multiple subscribers isn't supported as this may lead to inconsistent temporal data due to dependency on the local system clock. In this case, the publisher is used for an OLTP workload while subscriber serves for offloading reporting (including AS OF querying). When the distribution agent starts, it opens a transaction that is held open until distribution agent stops. ValidFrom and ValidTo are populated to the begin time of the first transaction that distribution agent starts. It may be preferable to run the distribution agent on a schedule rather than the default behavior of running it continuously, if having ValidFrom and ValidTo populated with a time that is close to the current system time is important to your application or organization. For more information, see [Temporal table usage scenarios](https://learn.microsoft.com/en-us/sql/relational-databases/tables/temporal-table-usage-scenarios?view=sql-server-ver16).
19. **Merge replication:** Not supported for temporal tables
20. Regular queries only affect data in the current table. To query data in the history table, you must use temporal queries. For more information, see [Querying data in a system-versioned temporal table](https://learn.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-table?view=sql-server-ver16).
21. An optimal indexing strategy will include a clustered columns store index and / or a B-tree rowstore index on the current table and a clustered columnstore index on the history table for optimal storage size and performance. If you create / use your own history table, we strongly recommend that you create this type of index consisting of period columns starting with the end of period column, to speed up temporal querying and speed up the queries that are part of the data consistency check. The default history table has a clustered rowstore index created for you based on the period columns (end, start). At a minimum, a nonclustered rowstore index is recommended.
22. The following objects/properties aren't replicated from the current to the history table when the history table is created:
    * Period definition
    * Identity definition
    * Indexes
    * Statistics
    * Check constraints
    * Triggers
    * Partitioning configuration
    * Permissions
    * Row-level security predicates
23. A history table can't be configured as current table in a chain of history tables.

* **Querying Table temporal Table**

Temporal syntax works on tables or views that are *stored locally* in the database. With remote objects such as tables on a linked server, or external tables, you can't use the FOR clause or period predicates directly in the query.

**Example1. For specific data range (Will display data after particular date)**

SELECT [PersonId]

,[Name]

,[Email]

,[Address]

,[PhoneNumber]

,[ValidFrom]

,[ValidTo]

FROM [TestDB].[dbo].[Person]

FOR System\_time AS OF '2023-06-05 09:48:50.9667043'

**Example2. For specific data range (Will display all the data)**

SELECT [PersonId]

,[Name]

,[Email]

,[Address]

,[PhoneNumber]

,[ValidFrom]

,[ValidTo]

FROM [TestDB].[dbo].[Person]

FOR System\_time All

**Example3. Querying data for analysis or complex queries.**

SELECT \* FROM Person FOR SYSTEM\_TIME

CONTAINED IN ('2021-01-01 00:00:00.0000000', '2022-01-01 00:00:00.0000000')

WHERE PersonID = 1000 ORDER BY ValidFrom;

**Example4. Creating views based on Temporal Table**

CREATE VIEW vw\_ProductInventoryDetails

AS

SELECT PrInv.ProductId ,PrInv.LocationId, P.ProductName, L.LocationName, PrInv.Quantity

, P.UnitPrice, L.NumberOfEmployees

, P.ValidFrom AS ProductStartTime, P.ValidTo AS ProductEndTime

, L.ValidFrom AS LocationStartTime, L.ValidTo AS LocationEndTime

, PrInv.ValidFrom AS InventoryStartTime, PrInv.ValidTo AS InventoryEndTime

FROM dbo.ProductInventory as PrInv

JOIN dbo.Product AS P ON PrInv.ProductId = P.ProductID

JOIN dbo.Location AS L ON PrInv.LocationId = L.LocationID;

GO

SELECT \* FROM vw\_ProductInventoryDetails

FOR SYSTEM\_TIME AS OF '2022-01-01';

* **Comparison between CT, CDC and temporal table.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Features** | **CT** | **CDC** | **Temporal** |
| Tracks Data Changes | Yes | Yes | Yes |
| Tracking mechanism | Synchronous | Asynchronous | Synchronous |
| Require enabling at Database Level | Yes | Yes | No |
| Require enabling at Table Level | Yes | Yes | Yes |
| Historical Data retention | No | Yes | Yes |
| Ability to Recover data from history | No | Yes | Yes |
| Requires change in current code | Yes | Yes | No |
| Table requires primary Key | Yes | Yes | Yes |
| Schema changes to the table allowed | Yes | No | Yes |
| Ability to analyze trends in data | No | No | Yes |
| Available in Standard Edition of SQL Server | Yes | No | Yes |
| History data is secure | No | No | Yes |
| Can you truncate history table? | Yes | Yes | No |
| History data resides in the same table | No | No | yes |
| Can you truncate main table? | Yes | No | No |
| Dependency on SQL Server Agent running | No | Yes | No |

For More reference on Temporal Table and its usage.

https://learn.microsoft.com/en-us/sql/relational-databases/tables/temporal-table-usage-scenarios?source=recommendations&view=sql-server-ver16