SQL Server 2016 Temporal摘要

> 文档：https://docs.microsoft.com/zh-cn/sql/relational-databases/tables/temporal-tables#see-also
>
> 视频介绍：https://channel9.msdn.com/Shows/Data-Exposed/Temporal-in-SQL-Server-2016

时间属性的支持

1. 只支持事务时间
2. Sys_begin_time和Sys_end_time，没有明确说明区间开闭
3. Sys_begin_time和Sys_end_time被设置为事务**开始**的时间

语法变化

CREATE TABLE

创建用户表时，即创建、绑定历史表

```
CREATE TABLE dbo.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));
```

1. 用户可自由命名系统列
2. 用户表和历史表是绑定关系
3. 用户表和历史表不用同属一个db，方便权限管理

ALTER TABLE

1. 创建用户表时，未创建历史表与之绑定，随后通过ALTER TABLE完成

```
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));
```

2. 用户表和历史表绑定，但想取消这种绑定关系

```
ALTER TABLE dbo.Department SET (SYSTEM_VERSIONING = OFF);
```

随之而来的，若想再次绑定

```
ALTER TABLE Employee
    SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.Employee_History));
```

解绑后在绑定，会进行时间对齐(align)检查

3. 彻底删除一个表上的时态内容

```
ALTER TABLE dbo.Department SET (SYSTEM_VERSIONING = OFF);
/*Optionally, DROP PERIOD if you want to revert temporal table to a non-temporal*/
ALTER TABLE dbo.Department
DROP PERIOD FOR SYSTEM_TIME;
```

解绑、删除时间段列

INSERT、UPDATE、DELETE

1. 由于SysStartTime和SysEndTime是隐藏的，因此用户无法赋值
2. 系统赋值为：
    1. INSERT，SysStartTime为INSERT事务的开始时间，SysEndTime为datetime2最大值
    2. UPDATE，原行SysEndTime设为UPDATE事务的开始时间，新行的SysStartTime设为UPDATE事务的开始时间，SysEndTime设为datetime2最大值(9999-12-31)；原行被记录到历史表，并设置为“关闭”
    3. DELETE，SysEndTime设为DELETE事务的开始时间，该行被记录到历史表，并设置为“关闭”

Query

1. 虽然有两张表`Employee` `Employee_History`，但SELECT只需要指明`Employee`，SQL Server会定向到`Employee_History`（如果涉及历史数据）
2. 语法扩展
    1. AS OF date_time，返回一个表，其行中包含过去指定时间点的实际（当前）值。在内部，临时表及其历史记录表之间将进行联合，然后筛选结果以返回在 date_time 参数指定的时间点有效的行中的值，覆盖历史和当前

如，查看某用户某天、时刻的值

```
... FROM Customer FOR SYSTEM_TIME AS OF '2015-1-1' WHERE CustomerID = 27
```

2. FROM start_date_time TO end_date_time 返回一个表，其中包含在指定的时间范围内保持活动状态的所有行版本的值；开区间

```
… FROM Customer FOR SYSTEM_TIME FROM '2015-1-1' TO '2015-1-2' WHERE CustomerID = 27
```

3. BETWEEN start_date_time AND end_date_time, 与2类似，但是左闭右开区间

   如，查看某用户某天的每个版本

```
… FROM Customer FOR SYSTEM_TIME BETWEEN '2015-1-1' AND '2015-1-2' WHERE CustomerID = 27
```

4. CONTAINED IN (start_date_time , end_date_time)，返回一个表，其中包含在 CONTAINED IN 参数的两个日期时间值定义的时间范围内打开和关闭的所有行版本的值。闭区间

   如，查看一段时间内，某用户的数据经历了什么样的操作

```
… FROM Customer FOR SYSTEM_TIME CONTAINED IN ('2015-1-1', '2015-1-2') WHERE CustomerID
= 27
```

3. Comparison

   如，和一年前相比部门人员架构如何：

```
SELECT * FROM Department_Emp
EXCEPT
SELECT * FROM Department_Emp
FOR SYSTEM_TIME AS OF '2014.01.01'
```

系统一致性检查

在将 **SYSTEM_VERSIONING** 设置为 **ON**之前，系统将对历史记录表和当前表执行一系列检查。 如果历史记录表不为空，则这些检查分为架构检查和数据检查。

架构检查

当前表和历史记录表中的列名和列数相同。

当前表与历史记录表中每一列的数据类型都一致。

将期间列设置为 NOT NULL。

当前表有主键约束，而历史记录表没有主键约束。//由此看来不支持主键

历史记录表中未定义任何 IDENTITY 列。

历史记录表中未定义任何触发器。

历史记录表中未定义任何外键。//由此看来不支持外键

历史记录表中未定义任何表或列约束。 但是，允许使用历史记录表上的默认列值。

不会将历史记录表置于只读文件组中。

历史记录表未配置更改跟踪或更改数据捕获。

数据一致性检查

将 SYSTEM_VERSIONING 设置为 ON 并作为任何 DML 操作的一部分之前，系统将执行以下检查： SysEndTime ≥SysStartTime

创建现有历史记录表的链接时，可以选择执行数据一致性检查。 此数据一致性检查可确保现有记录不重叠，并且每个单独的记录都满足临时要求。 系统默认执行数据一致性检查。 通常，每当当前表和历史记录表之间的数据可能失去同步时（如纳入已填充历史数据的现有历史记录表时），建议执行数据一致性检查。