Temporal Data

Leonard Lobel
CTO, SLEEK TECHNOLOGIES

@lennilobel

Introducing Temporal Tables

System version tables

Point-in-time data access

- Query updated and delete data, not just current data
- Seamless and transparent

Four primary use cases

- Time travel
- Slowly changing dimensions
- Auditing
- Accidental data loss recovery

Using Temporal

Create an ordinary table

- Must have primary key column
- Must have two period (start and end date) datetime2 columns

Enable the table for temporal

- Creates history table with same schema, but without constraints
- Automatically records updates and deletes to the history table

Query to point in time

- Include FOR SYSTEM_TIME AS OF in your SELECT statement

Manage schema changes

- ALTER TABLE automatically updates the history table
  - Some schema changes require turning temporal off, applying the changes to both tables, and then turning it back on

Creating a Temporal Table

```
CREATE TABLE Department
(
 DepartmentID    int NOT NULL IDENTITY(1,1) PRIMARY KEY,
 DepartmentName  varchar(50) NOT NULL,
 ManagerID       int NULL,
 ValidFrom       datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
 ValidTo         datetime2 GENERATED ALWAYS AS ROW END   NOT NULL,
 PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = DepartmentHistory))
```

Creating a Temporal Table

```
CREATE TABLE Department
(
 DepartmentID    int NOT NULL IDENTITY(1,1) PRIMARY KEY,
 DepartmentName  varchar(50) NOT NULL,
 ManagerID       int NULL,
 ValidFrom       datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
 ValidTo         datetime2 GENERATED ALWAYS AS ROW END   NOT NULL,
 PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = DepartmentHistory))
```

Creating a Temporal Table

```
CREATE TABLE Department
(
 DepartmentID    int NOT NULL IDENTITY(1,1) PRIMARY KEY,
 DepartmentName  varchar(50) NOT NULL,
 ManagerID       int NULL,
 ValidFrom       datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
 ValidTo         datetime2 GENERATED ALWAYS AS ROW END   NOT NULL,
 PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = DepartmentHistory))
```

Creating a Temporal Table

```
CREATE TABLE Department
(
 DepartmentID    int NOT NULL IDENTITY(1,1) PRIMARY KEY,
 DepartmentName  varchar(50) NOT NULL,
 ManagerID       int NULL,
 ValidFrom       datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
 ValidTo         datetime2 GENERATED ALWAYS AS ROW END   NOT NULL,
 PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.DepartmentHistory))
```

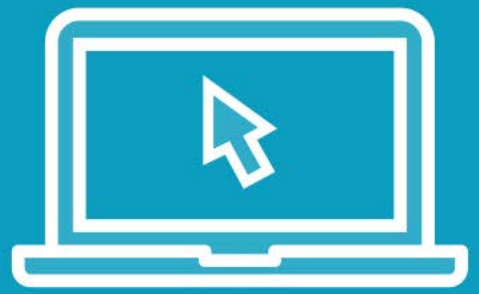
Querying a Temporal Table

```
DECLARE @ThirtyDaysAgo datetime2 = DATEADD(d, -30, SYSDATETIME())

SELECT *
 FROM Employee
 FOR SYSTEM_TIME AS OF @ThirtyDaysAgo
 ORDER BY EmployeeId
```

Querying a Temporal Table

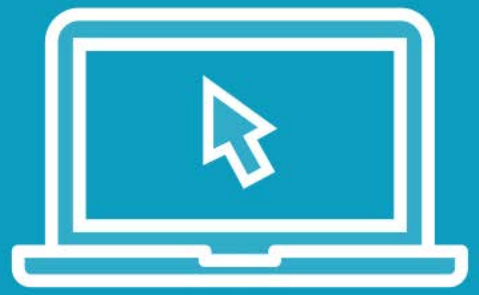
```
DECLARE @ThirtyDaysAgo datetime2 = DATEADD(d, -30, SYSDATETIME())

SELECT *
 FROM Employee
 FOR SYSTEM_TIME AS OF @ThirtyDaysAgo
 ORDER BY EmployeeId
```

Demo



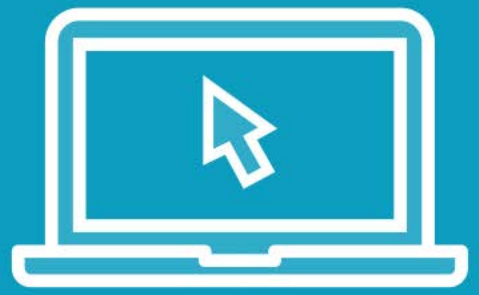
Creating a new temporal table

Demo



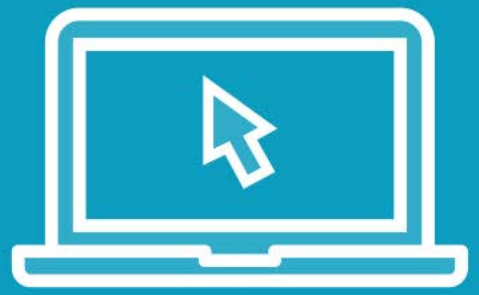
Converting an existing table to temporal

Demo



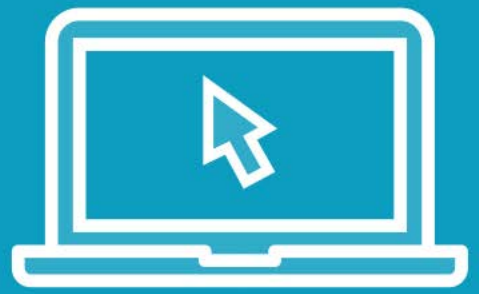
Querying temporal data

Demo



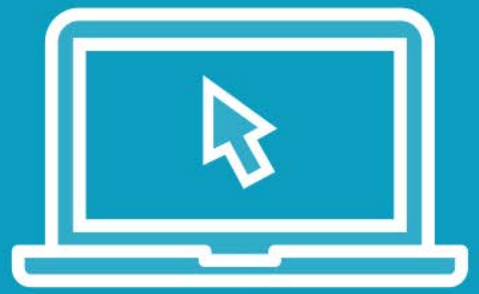
Combining temporal with stretch

Demo



Hiding period columns

Demo



Managing schema changes

Temporal Limitations and Considerations

Triggers

- INSTEAD OF triggers are unsupported
- AFTER triggers are supported on the current table only

Cascading updates and deletes are not supported

In-memory OLTP (Hekaton) is not supported

FILESTREAM/FileTable is not supported

INSERT and UPDATE statements cannot reference the period columns

Works with other new SQL Server 2016 features

- DDM, RLS, Always Encrypted, Stretch

Summary

- Creating temporal tables
- Point-in time querying
- Combining temporal with stretch
- Hiding period columns
- Managing schema changes
- Limitations and considerations