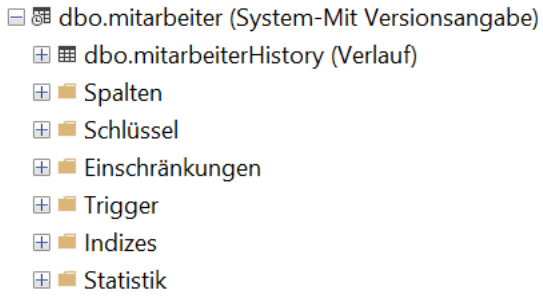
Temporale Tabellen

- Seit SQL Server 2016 besteht die Möglichkeit, temporale Tabellen zu verwenden. Solche Tabellen ermöglichen es, dass Tabellen auf der Grundlage von Zeitstempeln eine Historie führen können. Der Vorteil von temporalen Tabellen ist, dass ein Zeitpunkt angegeben werden kann. Dann kann man feststellen, welchen Wert die Daten in der Tabelle zu diesem Zeitpunkt hatten.
- Im Wesentlichen werden alle Datenänderungen nachverfolgt. Um die Änderungen zu verfolgen, wird eine zweite Tabelle genutzt, um die Historie der vorherigen Versionen der geänderten Datenzeilen zu speichern.
- Beispiel: In der Mitarbeitertabelle wird die AbteilungId eingetragen, d.h. die Zugehörigkeit zu einer Abteilung per Referenz auf die Tabelle Abteilung abgebildet (analog Funktion, Niederlassung). Um die Tabelle zu historisieren, könnte man die 1:n – Beziehungen in n:m – Beziehungen umwandeln. Die Umwidmung in eine temporale Tabelle ist jedoch effizienter

```
modelBuilder .Entity<Mitarbeiter>() .ToTable("Mitarbeiter", b => b.IsTemporal());
```

Nach der Migration

- Nach der Migration entsteht eine zweite Tabelle MitarbeiterHistory, die die Historie der Abteilungszugehörigkeit usw. zu verfolgen gestattet. Sie ist der Mitarbeitertabelle zugeordnet und kann nicht barbeitet werden:



- Die beiden Tabellen haben identische Strukturen, beide enthalten zusätzlich die Felder PeriodStart und PeriodEnd. Die bisherige Mitarbeitertabelle enthält den aktuellen Stand der Daten, in der History-Tabelle werden vorhergehende Änderungen registriert:

```
select Nachname, vorname, abteilungid, PeriodStart, PeriodEnd from mitarbeiter where mitarbeiterId=6
union
select Nachname, vorname, abteilungid, PeriodStart, PeriodEnd from mitarbeiterHistory where mitarbeiterId=6;
```

Abfragen mittels EFC-LINQ

```
var history = dbctx.MitarbeiterListe
      .TemporalAll()
      .Where(emp => emp.MitarbeiterId == id)
      .OrderByDescending(emp => EF.Property<DateTime>(emp, "PeriodStart"))
      .Select(emp => new
      {
          Mitarbeiter = emp,
          PeriodStart = EF.Property<DateTime>(emp, "PeriodStart"),
          PeriodEnd = EF.Property<DateTime>(emp, "PeriodEnd")
      });
Display(history.ToQueryString());
```

```
SELECT [m].[mitarbeiterId], [m].[abteilungId], [m].[datum_austritt], [m].[datum_eintritt],
[m].[durchwahl], [m].[email], [m].[extern], [m].[funktionId], [m].[geburtsdatum],
[m].[geschlecht], [m].[nachname], [m].[niederlassungId], [m].[PeriodEnd], [m].[PeriodStart],
[m].[prsnr], [m].[vorname]
FROM [dbo].[mitarbeiter] FOR SYSTEM_TIME ALL AS [m]
WHERE [m].[mitarbeiterId] = 6
ORDER BY [m].[PeriodStart] DESC
```

Vorteile von zeitlichen Tabellen

vgl. MS SQL: Temporal table and its use under EF Core | by Ben Witt | Medium

- Historische Daten sind verfügbar:
  - Zeitliche Tabellen ermöglichen die Speicherung historischer Daten, die eine genaue Aufzeichnung von Änderungen der Daten im Laufe der Zeit ermöglichen. Dies ist besonders nützlich für Compliance-Anforderungen, Audits und die Verfolgung von Entwicklungen.
- Einfache Analyse von Datenänderungen:
  - Historische Daten ermöglichen eine einfache Verfolgung von Datenänderungen im Laufe der Zeit. Dies erleichtert die Analyse von Trends, das Erkennen von Mustern und das Verständnis von Datenentwicklungen.
- Korrekturen und Stornierungen:
  - Zeitliche Tabellen ermöglichen es, zu einem bestimmten Zeitpunkt in der Vergangenheit zurückzugehen und den Datenzustand wiederherzustellen. Dies ist besonders nützlich, wenn Fehler korrigiert werden müssen oder eine frühere Version der Daten wiederhergestellt werden soll.
- Versionierung ohne manuellen Aufwand:
  - Die Versionierung von Daten wird automatisch durch temporale Tabellen verwaltet, wodurch der manuelle Aufwand für die Protokollierung von Änderungen oder die Implementierung separater Mechanismen zur Versionskontrolle entfällt.

Vorteile von zeitlichen Tabellen - Fortsetzung

- Einfache Verwaltung von Gültigkeitszeiträumen:
  - Zeitliche Tabellen unterstützen die Organisation von Daten in Bezug auf ihren Gültigkeitszeitraum. Diese Funktion ist besonders nützlich in Szenarien, in denen Daten für bestimmte Zeiträume gültig sind, wie z. B. Mitarbeiterverträge, Preisänderungen oder Produktverfügbarkeit.
- Compliance und Nachvollziehbarkeit:
  - Zeitliche Tabellen fördern Compliance-Anforderungen, indem sie detaillierte Informationen über alle Datenänderungen liefern. Dies ist besonders wichtig in regulierten Branchen, in denen eine genaue Rückverfolgbarkeit von Datenänderungen erforderlich ist.
- Optimierte Fehlerdiagnose:
  - Bei unerwarteten Ergebnissen ermöglichen temporale Tabellen eine schnellere Fehlerdiagnose, indem sie einen klaren Einblick in die Datenhistorie geben. So wird transparent, wann und wie sich Daten verändert haben.

Nachteile von zeitlichen Tabellen

- Erhöhte Speicheranforderungen:
  - Zeitliche Tabellen, die eine Historie von Daten speichern, können den Speicherbedarf aufgrund zusätzlicher Zeilen in der Historientabelle und erforderlicher Zeitstempel erhöhen.
- Komplexität der Datenbankstruktur:
  - Die Verwendung von temporären Tabellen mit zwei Tabellen (eine für aktuelle Daten, eine für die Historie) und zusätzlichen Zeitstempelspalten führt zu einer komplexeren Datenbankstruktur.
- Leistungskosten:
  - Die Verfolgung und Verwaltung von Verlaufsdaten kann zu Leistungseinbußen führen, insbesondere bei großen Datenmengen, und komplexe Abfragen im Zeitverlauf können zeitaufwendig sein.
- Datenbankmigration und -änderungen:
  - Die Einführung von zeitlichen Tabellen oder Änderungen an der Strategie für Verlaufsdaten in einer bestehenden Datenbank kann zu migrationsbedingten Komplexitäten und Herausforderungen führen.
- Verwaltung der Historientabelle:
  - Eine sorgfältige Verwaltung der Historientabelle ist entscheidend, um Redundanzen zu vermeiden und eine korrekte Speicherung der Daten zu gewährleisten, was zusätzliche Wartungsaufgaben erfordern kann.
- Komplexität der Entwicklung:
  - Die Arbeit mit Zeittabellen kann umfangreichere Konfigurations- und Entwicklungsarbeiten erfordern, die Entwicklern mit dieser Funktion möglicherweise weniger vertraut sind.