# Db2 Expertentipp

## Tabellenänderungen nachvollziehen mit Db2 systemtemporalen Tabellen

Oft wäre es hilfreich, nachvollziehen zu können, wann sich ein Tabelleneintrag wie geändert hat. So will man häufig bei Konfigurationstabellen sehen, seit wann welcher Wert gültig ist und was vorher galt.

**In Db2 gibt es dafür die perfekte Lösung: System Temporal Tables** (was mit „Temporale Tabellen für Systemzeitraum“ in Deutsch übersetzt wird) - nicht zu verwechseln mit Temporären Tabellen (wie DGTT oder CGTTs). Dabei übernimmt Db2 automatisch die ganze **Historisierung, ohne dass die Anwendung angepasst werden muss** - einer der gewaltigen Vorteile.

Technisch bestehen die System Temporal Tables aus zwei Tabellen - eine mit den aktuellen Daten und eine Tabelle mit den historischen Daten. Das Gute ist, dass sie nach außen, also zum SQL, wie ein Objekt auftreten und mit temporalen SQL - sogenanntem Time Travel SQL - selektiert werden können, als wäre es ein Objekt. Diese SQL-Erweiterung ist - was viele nicht vermuten - Teil des SQL-Standards (SQL:2011).

**Am Beispiel lässt sich das einfacher verdeutlichen - gehen wir also in die Praxis:**

```
CREATE TABLE EMPL (
       empno          char(6) not null,
       workdept       char(3),
       edlevel        int,
       system_start   timestamp(12) generated always as row begin not null,
       system_end     timestamp(12) generated always as row end not null,
       trans_start    timestamp(12) generated always as transaction start id
                      not null,
PERIOD SYSTEM_TIME (system_start, system_end)
).
```

Drei zusätzliche Spalten sind in der Tabelle erfordert, die Zeitinformationen enthalten, um die Historisierung abzubilden. Da diese Timestamp (12) Spalten „generated always“ zu definieren sind, werden sie von Db2 automatisch befüllt.

*system_start: Beginn des Gültigkeitszeitraums*
*system_end: Ende des Gültigkeitszeitraums*
*trans_start: Startzeitpunkt der Transaction - wird nur für die Bereinigung von Sonderfällen benötigt*

Die zusätzliche PERIOD Klausel definiert, welche Spalten für die Systemhistorisierung herangezogen werden sollen, da die Spaltennamen frei gewählt werden können.

Um die historischen Sätze verwalten zu können, wird eine Historien-Tabelle benötigt, die prinzipiell dieselbe Struktur hat, wie die Ausgangstabelle und z.B. so angelegt werden kann:

```
CREATE TABLE empl_hist LIKE empl
```

Erzeugt einen strukturellen Klon der Ausgangstabelle. Diese dient dazu alle abgeschlossenen Zeitintervalle - also die Historie - aufzunehmen.

**Tipp:**
Da es sich um ein separates Objekt handelt, können hier auch spezifische Einstellungen und Berechtigungen gesetzt werden. So kann man User auf die Basistabelle für Insert, Update und Delete berechtigen, aber nur das SELECT-Recht auf die EMPL_HIST geben, um eine Manipulation der Historie zu verhindern. Der DB-Admin kann auch die Option „append on“ im CREATE mitgeben, da in diese Tabelle nur eingefügt wird, um diese Einfüge-Operation zu optimieren, denn eine Freiplatzsuche wird hier nicht benötigt.

Der letzte Schritt ist die Verbindung der beiden Objekte zu einem logischen Objekt via:

```
ALTER TABLE empl ADD VERSIONING USE HISTORY TABLE empl_hist
```

**Tipp:**
Wer prüfen will, ob solche Tabellen bereits in der Datenbank vorhanden sind, kann dies über die Spalte TEMPORALTYPE in SCSCAT.TABLES abfragen, ein „S“ steht dabei für System Temporal. Alternativ stehen weitere Details dazu in der SYSCAT.PERIODS.

Damit ist das DDL fertig und die Nutzung kann ausprobiert werden.

**Einige beispielhafte INSERTs:**

```
INSERT INTO empl (empno, workdept, edlevel) VALUES
  ('001000', 'D11', 13)
, ('001100', 'D11', 14)
, ('001200', 'E11', 10)
```

Das ergibt also drei Zeilen in der Tabelle EMPL.

EMPL

| EMPNO | WORKDEPT | EDLEVEL | SYSTEM_START | SYSTEM_END | TRANS_START |
|---|---|---|---|---|---|
| 001000 | D11 | 13 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |
| 001100 | D11 | 14 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |
| 001200 | E11 | 10 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |

Abbildung 1: Nach Insert

Interpretiert wird dies nun folgendermaßen: Drei Mitarbeiter wurden am 01.12.2022 angelegt mit ihren jeweiligen - aktuell noch gültigen - EDLEVEL. Die zugehörige EMPL_HIST ist zu diesem Zeitpunkt noch leer, denn es hat ja keine Änderung (UPDATE oder DELETE) stattgefunden. Das ändert sich, wenn wir folgendes Update (zeitlich am 23.12.2022 um 11:00 Uhr) durchführen:

```
UPDATE empl
   SET edlevel = edlevel + 1
 WHERE empno = '001000'
```

EMPL

| EMPNO | WORKDEPT | EDLEVEL | SYSTEM_START | SYSTEM_END | TRANS_START |
|---|---|---|---|---|---|
| 001000 | D11 | 14 | 2022-12-23-11.01.08.500341 | 9999-12-30-00.00.00.000000 | 2022-12-23-11.01.08.500341 |
| 001100 | D11 | 14 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |
| 001200 | E11 | 10 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |

EMPL_HIST

| EMPNO | WORKDEPT | EDLEVEL | SYSTEM_START | SYSTEM_END | TRANS_START |
|---|---|---|---|---|---|
| 001000 | D11 | 13 | 2022-12-01-08.00.29.013618 | 2022-12-23-11.01.08.500341 | 2022-12-01-08.00.29.013618 |

Abbildung 2: Nach Update

**Das ist nun folgendermaßen zu verstehen:**

Der Mitarbeiter '001000' hat aktuell den EDLEVEL = 14 – genauer gesagt seit 23.12.2022 um 11.01 Uhr und dieser ist aktuell bis ans Ende aller Zeit gültig.

Die EMPL_HIST Tabelle zeigt, dass eben dieser Mitarbeiter vom 01.12.2022 bis 11:01 Uhr am 23.12.2022 den EDLEVEL 13 hatte. Auffällig ist, dass der SYSTEM_START Zeitpunkt des aktuellen Satzes und der SYSTEM_END des vorhergehenden Zustandes den exakt selben Zeitstempel haben. Wichtig: Dies ist problemlos, denn in Db2 sind die Startzeitpunkte inklusiv zu sehen, während die Endzeitpunkte exklusiv sind. Die Eindeutigkeit ist somit gewährleistet.

Die EMPL_HIST wurde nicht im Update erfasst, sondern Db2 hat dies im Hintergrund selbstständig verarbeitet – ein weiterer großer Vorteil.

**Jetzt wollen wir noch prüfen, was bei einem regulären Delete vor sich geht:**

```
DELETE FROM empl
 WHERE empno = '001200'
```

EMPL

| EMPNO | WORKDEPT | EDLEVEL | SYSTEM_START | SYSTEM_END | TRANS_START |
|---|---|---|---|---|---|
| 001000 | D11 | 14 | 2022-12-23-11.01.08.500341 | 9999-12-30-00.00.00.000000 | 2022-12-23-11.01.08.500341 |
| 001100 | D11 | 14 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |

EMPL_HIST

| EMPNO | WORKDEPT | EDLEVEL | SYSTEM_START | SYSTEM_END | TRANS_START |
|---|---|---|---|---|---|
| 001000 | D11 | 13 | 2022-12-01-08.00.29.013618 | 2022-12-23-11.01.08.500341 | 2022-12-01-08.00.29.013618 |
| 001200 | E11 | 10 | 2022-12-01-08.00.29.013618 | 2022-12-23-11.15.43.591320 | 2022-12-01-08.00.29.013618 |

Abbildung 3: Nach Delete

Wie zu erwarten, wurde der Datensatz des Mitarbeiters 001200 aus der Tabelle EMPL gelöscht (Abbildung 3).

In der EMPL_HIST taucht der Mitarbeiter hingegen neu auf, mit einer Gültigkeit vom 01.12.2022 08:00 Uhr bis 23.12.2022 11:15 Uhr, denn in diesem Zeitraum gab es diesen Mitarbeiter mit dem EDLEVEL = 10.

Diese Daten kann man nun auch mit sogenanntem Time Travel SQL abfragen. Das Besondere hierbei ist, dass man die einzelnen Tabellen nicht separat abfragen muss, sondern nur die EMPL Tabelle angeben muss – alles andere wiederum erledigt Db2 automatisch – noch einer der Vorteile.

Als Beispiel nehmen wir die folgende Fragestellung:
Welchen EDLEVEL hatten die Mitarbeiter des Departments D11 am 15.12.2022?

```
SELECT *
  FROM EMPL FOR SYSTEM_TIME AS OF '2022-12-15-00.00.00.000000'
 WHERE WORKDEPT = 'D11'
```

| EMPNO | WORKDEPT | EDLEVEL | SYSTEM_START | SYSTEM_END | TRANS_START |
|---|---|---|---|---|---|
| 001000 | D11 | 13 | 2022-12-01-08.00.29.013618 | 2022-12-23-11.01.08.500341 | 2022-12-01-08.00.29.013618 |
| 001100 | D11 | 14 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |

Abbildung 4: Ergebnis des AS OF SELECTs

Der Zustand eines Zeitpunktes kann also ganz ohne Bedingungen auf SYSTEM_START und SYSTEM_END erreicht werden. Bei diesem Ergebnis wurde der Zustand mit EDLEVEL = 13 des Mitarbeiters 001000 automatisch aus der EMPL_HIST verwendet, wie Abbildung 4 zeigt.

Ebenso kann man auch einen Zeitraum abfragen – wenn z.B. der gesamte Dezember betrachtet werden soll:

```
SELECT *
  FROM EMPL FOR SYSTEM_TIME FROM '2022-12-01' TO '2022-12-31'
```

| EMPNO | WORKDEPT | EDLEVEL | SYSTEM_START | SYSTEM_END | TRANS_START |
|---|---|---|---|---|---|
| 001000 | D11 | 13 | 2022-12-01-08.00.29.013618 | 2022-12-23-11.01.08.500341 | 2022-12-01-08.00.29.013618 |
| 001000 | D11 | 14 | 2022-12-23-11.01.08.500341 | 9999-12-30-00.00.00.000000 | 2022-12-23-11.01.08.500341 |
| 001100 | D11 | 14 | 2022-12-01-08.00.29.013618 | 9999-12-30-00.00.00.000000 | 2022-12-01-08.00.29.013618 |
| 001200 | E11 | 10 | 2022-12-01-08.00.29.013618 | 2022-12-23-11.15.43.591320 | 2022-12-01-08.00.29.013618 |

Abbildung 5: Ergebnis der Zeitraumabfrage

Db2 bietet noch wesentlich mehr in diesem Bereich, so gibt es neben den System-Temporalen Tabellen auch Business-Temporal Tables und die Kombination von beiden Techniken, was dann als Bi-Temporale Tabelle bezeichnet wird. Das soll aber in separaten Blogbeiträgen behandelt werden.

**Vorerst viel Spaß beim Testen** 😊

**Mein Fazit:**
Systemtemporale Tabellen sind ein sehr nützliches Hilfsmittel, um Veränderungen in Tabellen zeitlich korrekt zu erfassen und das, OHNE Anpassungen in der Applikation vornehmen zu müssen.
Es handelt sich aber nicht um Monitoring – dafür gibt es andere Möglichkeiten.

Tipp: Falls jemand per „SELECT *“ auf die obigen Tabellen zugreift, kann man im CREATE TABLE die drei extra Spalten auch per „implicitly hidden“ Klausel verstecken. So können sie zwar explizit abgefragt werden, aber werden im „SELECT *“ nicht ausgegeben.

Ich berate Sie gerne, falls auch Sie die Historisierung in Db2 Hände legen wollen. Bitte schicken Sie mir dazu einfach eine kurze Mail an kontakt@mip.de. Ich freue mich auf Ihre Anfrage.

Ihr Michael Tiefenbacher

Principal Consultant