Oberseminar: Datenbanksysteme aktuelle Trends

Temporale Datenbanken

Jonas Wagner, 62423, 13INM

Motivation

Datenbanken bilden Ausschnitte von realen Sachverhalten ab[1]. Dabei wird häufig von Fakten abstrahiert, welche für eine bestimmte Anwendung keine Rolle spielen. Auch temporale Aspekte werden teilweise vernachlässigt. Dies ist jedoch häufig ein Fehler und führt dazu, dass einige Datenbankanwendungen diese Aspekte nachträglich unter großem Aufwand[7] hinzufügen. Doch was genau sind temporale Daten und wie lassen sich diese einordnen?

Fakten in Datenbanken können im Allgemeinen folgende Zeitaspekte enthalten:

- Ein Fakt besitzt ein **Zeitattribut**, zum Beispiel das Eingangsdatum einer Rechnung oder das Einwurfdatum eines Briefs.
- Ein Fakt **ändert sich im Laufe der Zeit**, zum Beispiel ein Preis, welcher im Sommerschlussverkauf gesenkt wird oder Gesetze, welche hin und wieder geändert werden.

Einen Sonderfall stellen Datenreihen dar. Diese lassen sich beiden Kategorien zuordnen, zeichnen sich jedoch zusätzlich durch häufige Messzeitpunke und/oder kurze Abstände zwischen diesen aus.

Bei temporalen Daten spricht man häufig von **Gültigkeits**- und **Transaktionszeit**. Die Gültigkeitszeit stellt einen Zeitraum dar, in dem ein Fakt Gültigkeit besitzt. Zum Beispiel der Preis eines Artikels während des Sommerschlussverkaufs. Die Transaktionszeit

bschreibt den Zeitpunkt, an dem ein Fakt erstellt oder geändert wurde. Beispielsweise der Moment in dem ein Preis in die Datenbank eingetragen wurde. **Bitemporale** Fakten sind solche, bei denen sowohl Gültigkeits- als auch Transaktionszeit von Bedeutung sind.

Temporale Daten können in Datenbanken auf verschiedene Weise abgebildet werden. So ist es beispielsweise möglich den Gültigkeitszeitraum eines Fakts mithilfe von zwei Zeitstempeln abzubilden, welche von allen gängigen Datenbanksystemen unterstützt werden. Die Abfrage von gültigen Daten gestaltet sich dabei jedoch ungemein schwer. Bei der Verwendung von SQL3 wäre es beispielsweise notwendig bei jeder Abfrage zu testen ob die angegebene Abfragezeit vor der Endzeit des Gültigkeitszeitraums und nach dessen Anfangszeit liegt. Auch das Erfassen der Transaktionszeit bringt einige Probleme mit sich, da verhindert werden muss, dass diese im Nachhinein geändert wird. Dies ist beispielsweise mit triggern nötig. Desweiteren ist es notwendig zu verhindern, dass eine beliebige Transaktionszeit eingegeben werden kann. Auch dies lässt sich mit triggern realisieren. Auch die Versionierung von Attributen eines Tupels oder des gesamten Tupels kann von Interesse sein um den Änderungsverlauf von Fakten nachvollziehen zu können, zusätzlich zum Zeitattribut spielt hierbei häufig auch der Ersteller des neuen Fakts eine Rolle (Audit-Log). Um dies in SQL3 zu ermöglich müsste wieder ein Trigger ingeführt werden, welcher ein `UPDATE` verhindert und durch ein `INSERT` mit den entsprechenden Parametern (aktuelle Zeit, angemeldeter Nutzer) ersetzt. Leider müssen durch dieses Vorgehen automatisch die Primärschlüssel erweitert werden um weiterhin eindeutige Schlüssel verwenden zu können. Doch auch dies stellt den Datenbanknutzer vor ein Problem: Wie kann man bewerkstelligen, dass beispielsweise zwei Gültigkeitszeiträume sich nicht überschneiden?

Aus diesen Ausführungen wird deutlich, dass die Verwendung eines herkömmlichen Datenbanksystem ohne spezielle Zeitdatentypen und -perationen, sowie explizite Unterstützung von temporalen Daten erheblichen Aufwand seitens des Datenbanknutzers

mit sich bringt. Eine Datenbank sollte folgende Anforderungen erfüllen um den Aufwand für den Nutzer, also den Anwendungsprogrammierer möglichst gering zu halten:

- Zeiträume müssen abgebildet werden können. Zusätzlich sind Prädikate zur Feststellung von Zeiträumen zueinander wünschenswert.
- Die Transaktionszeit muss implizit, also ohne Aufwand durch den Anwendungsentwickler aufgezeichnet werden können.
- Der Zustand von Daten zu einem bestimmten Zeitraum kann abgefragt werden. Dabei sollten auch Integritätsbedingungen für Daten mit Gültigkeitszeit beachtet werden.
- Die Versionierung von Tupeln sollte ebenfalls implizit möglich sein.

Spezifikation

Bereits im Jahr 1994 wurde eine Erweiterung des SQL-Standards um die Unterstützung von temporalen Daten diskutiert[3]. ANSI (American National Standards Institute) und ISO (International Organization for Standardization) waren an der Ausarbeitung beteiligt. Der erste Vorschlag wurde auf Grundlage der Arbeiten von Richard Snodgrass (Universität von Arizona) erarbeitet, welcher die TSQL2-Sprache veröffentlich hatte. Diese war eine Erweiterung des SQL-92-Standards um die Unterstützung temporaler Daten.

TSQL2 definiert den neuen Datentyp **Period** um Zeiträume abzubilden[2]. Eine Period P besteht dabei aus einem Startzeitpunkt (P^-) und einem Endzeitpunkt (P^+) und repräsentiert alle Zeitpunkte zwischen P^- und P^+ inklusive Start- und exklusive Endzeitpunkt. Dadurch gilt: die Perioden Period(x,y) und Period(y,z) enthalten keinen gemeinsamen Zeitpunkt, wenn x vor y und y vor z liegt. TSQL2 definiert weiterhin Prädikate zur Arbeit mit Perioden:

- $P_a = P_b$ $\Leftrightarrow P_a^- = P_b^- \wedge P_a^+ = P_b^+$
- P_a `CONTAINS` P_b $\Leftrightarrow P_a^- \leq P_b^- \wedge P_a^+ \geq P_b^+$
- P_a `MEETS` P_b $\Leftrightarrow P_a^+ = P_b^-$
- P_a `OVERLAPS` P_b $\Leftrightarrow P_a^+ > P_b^-$

- P_a PRECEDES P_b $\Leftrightarrow P_a^+ < P_b^-$

Zusätzlich zum neuen Datentyp Period wird spezifiziert, wie Transaktions- und Gültigkeitszeiten abgebildet werden können. Diese werden implizit in Tabellenspalten verwaltet, welche durch das `TRANSACTION` oder `VALID STATE` Schlüsselwort bei der `CREATE TABLE` Anweisung erzeugt werden.

Die TSQL2-Spezifikation wurde jedoch nicht vom ISO-Gremium angenommen, da sie einige Probleme enthielt. Die Arbeiten an den temporalen Konzepten in SQL wurde bis 2008 mehr oder weniger eingestellt. Erst dann konnte eine Einigung erzielt werden und ISO und ANSI einigten sich darauf, dass temporale Daten in den Teil SQL/Foundations des SQL-Standards aufgenommen werden sollten, welcher die Fundamente von SQL beschreibt. Mit SQL2011 wurden dann *Systemversionierte Tabellen* und sogenannte *application-time period tables* in den Standard aufgenommen.

Application-time period tables bilden die Gültigkeitsdauer von Daten ab[4]. Sie werden mithilfe der neuen `PERIOD FOR` Klausel bei der Erstellung von Tabellen definiert. Dabei wird angegeben welche zwei Tabellenspalten die Gültigkeitsperiode darstellen sollen. So wird die implizite Erstellung von Tabellenspalten umgangen. Zusätzlich kann bei der `PRIMARY KEY` Definition die Periode hinzugefügt werden und spezifiziert werden, dass dieser keine Überlappungen enthalten darf (`WITHOUT OVERLAPS`).

Die im TSQL2-Standard definierten Prädikate zur Arbeit mit Perioden wurden in SQL2011 übernommen und erweitert. Die Prädikate können bei der Selektion verwendet werden.
Beispiel: In welcher Abteilung haben Mitarbeiter in Mai und Juni gearbeitet?

```
SELECT name, abteilung FROM mitarbeiter
WHERE abt_period CONTAINS
     PERIOD(DATE ´2014.05.01´, DATE ´2014.07.01´ );
```

Systemversionierte Tabellen werden erzeugt, indem `SYSTEM_TIME` Spalten bei der Tabellendefinition angegeben werden und das Schlüsselwort `WITH SYSTEM VERSIONING` verwendet wird. Selbst beim ändern und löschen aus solchen Tabellen bleiben die alten Tupel erhalten. Die Transaktionsperiode wird jedoch bei DML-Anweisungen geändert, sodass das System feststellen kann, welche Tupel in der aktuellen Sicht angezeigt werden müssen. Eine einfache `SELECT`-Anweisung würde also die Tupel mit den letzten Änderungen anzeigen. Um einen älteren Zustand der Datenbank einsehen zu können kann beim Abfragen der Daten das Schlüsselwort `FOR SYSTEM_TIME` verwendet werden. Nach diesem kann ein Datum (`AS OF`) oder eine Periode (`BETWEEN [...] AND [...]` oder `FROM [...] TO [...]`) definiert werden.

Zusammenfassung und Ausblick

Die Arbeit mit temporalen Daten ist schon vor SQL2011 möglich gewesen. Einige Datenbanksysteme (DBS) definieren eigene Datentypen oder Konstrukte um die Arbeit mit solchen zu vereinfachen. Bei der Verwendung von solchen DBS, bei denen keine explizite Unterstützung für temporale Daten vorhanden ist muss ein erheblicher Aufwand betrieben werden um temporale Daten zu verwalten. Mit dem SQL2011-Standard wurde SQL endlich um temporale Daten erweitert. Eine Erweiterung, welche viel früher hätte geschehen müssen, da viele Anwendungen temporale Konzepte auf nachahmen mussten. Ein Fakt, der der auch dadurch belebt wird, dass bereits 1994 mit TSQL2 eine Erweiterung für temporale Konzepte veröffentlicht wurde[3], also bereits damals Interesse an einer solchen Erweiterung bestand.

Trotz dem Einzug von temporalen Daten in SQL2011 ist es bis heute (07/2014) nur einem DBS-Anbieter gelungen diesen Standard umzusetzen (IBM mit DB2 v10). Dadurch ist ein produktiver Einsatz der temporalen Konzepte noch lange nicht für jeden Entwickler möglich. Es bleibt nur zu hoffen, dass die Implementierung des Standards nicht mehr allzu

lange auf sich warten lässt und auch die anderen DBS-Anbieter den großen Nutzen der temporalen Daten sehen. Dass der Nutzen enorm ist hat IBM bereits 2012 untersucht[7] und ermittelt, dass eine Implementierung mit herkömmlichen Mitteln (Trigger, Prozeduren) im Gegensatz zur expliziten Unterstützung von temporalen Daten um den Faktor 16 aufwendiger ist.

Quellen

[1] PETKOVIĆ, Dušan. Was lange währt, wird endlich gut: Temporale Daten im SQL-Standard. Datenbank-Spektrum, 2013, 13. Jg., Nr. 2, S. 131-138.

[2] SNODGRASS, Richard T. (Hg.). *The TSQL2 temporal query language*. Springer, 1995.

[3] SNODGRASS, Richard T., et al. Transitioning temporal support in TSQL2 to SQL3. In: *Temporal Databases: Research and Practice*. Springer Berlin Heidelberg, 1998. S. 150-194.

[4] KULKARNI, Krishna; MICHELS, Jan-Eike. Temporal features in SQL: 2011. *ACM SIGMOD Record*, 2012, 41. Jg., Nr. 3, S. 34-43.

[5] BÖHLEN, Michael H., et al. Querying TSQL2 databases with temporal logic. In: *Advances in Database Technology—EDBT'96*. Springer Berlin Heidelberg, 1996. S. 325-341.

[6] MYRACH, Thomas. TSQL2: Der Konsens über eine temporale Datenbanksprache. *Informatik-Spektrum*, 1997, 20. Jg., Nr. 3, S. 143-150.

[7] SARACCO, Cynthia M.; NICOLA, Matthias; GANDHI, Lenisha. A Matter of Time: Temporal Data Management in DB2 for z/OS. 2012.