Oberseminar: Datenbanken - Aktuelle Trends SS19
Leitung: Prof Kudraß

Fakultät für Informatik und Medien
HTWK Leipzig

Temporale Datenbanken

Luka Mutschler

3. Juli 2019

Inhaltsverzeichnis

Abstract

Die Zeit ist etwas so elementares, das uns ständig umgibt und im alltäglichen Leben, sowie in fast allen Vorgängen, ob technischer, wirtschaftlicher oder sozialer Natur eine Rolle spielt. Zugleich ist die Zeit etwas, das gar nicht so leicht zu fassen und Gegenstand zahlreicher philosophischer Diskussionen ist. Für die Abbildung in Temporalen Datenbanken wird die Zeit duch verschiedene Temporale Datentypen als Zeitpunkt, Dauer oder Intervall repräsentiert. Klassische Relationale Datenbanken sind ein vereinfachtes Abbild der Realität. Temporale Datenbanken erweitern dieses, indem sie die zeitliche Dimension ebenfalls in ihrem Modell abbilden.
Die Konzepte der Temporalen Datenbanken unterscheiden sich anhand der Datentypen, die sie für die Zeitstempelung verwenden und in der Frage ob ein gesamtes Tupel oder jedes einzelne Attribut gestempelt werden soll. Die Zeitstempelung kann entweder angeben in welchem Zeitraum der Wert eines zeitabhängigen Attributs gültig ist oder die Änderung von Einträgen nachvollziehbar machen. Auch eine Kombination beider Möglichkeiten ist denkbar. Mit IXSQL, TSQL2 und ATSQL2 existieren mehrere theoretische Konzepte zu Temporalen Datenbanken. Der Standard SQL:2011 unterstützt erstmals die Verwendung von Temporalen Datenbanken. Eine vollständige Implementierung des Standards findet sich nur bei wenigen Datenbankmanagementsystemen. Die Anwendungsbereiche von Temporalen Datenbanken sind hingegen breit gefächert und werden in Zukunft noch zunehmen.

1 Basiskonzepte bei Temporalen Datenbanken

Herkömmliche Relationale Datenbanken stellen ein Abbild der Realität dar. Beispielsweise kann ein Fahrradladen eine Übersicht über seine vorhandenen Artikel, inklusive Preis, Menge und Lagerort in einer Datenbank abspeichern. Dieses Abbild der Realität ist jedoch nur zu einem bestimmten Zeitpunkt gültig. Sobald sich beispielsweise der Preis eines Artikels ändert muss der entsprechende Eintrag überschrieben werden. Es ist weder möglich Änderungen nachzuvollziehen noch einen Verlauf von Werten über die Zeit abzubilden. Die zeitliche Dimension wird schlichtweg nicht berücksichtigt. Temporale Datenbanken verfolgen den Ansatz diese auch mitabzubilden. [9]

1.1 Berücksichtigung des Faktors Zeit

Temporale Datenbanken stellen eine Erweiterung Relationaler Datenbanken um die zeitliche Dimension dar. Man kann sie sich aus mehreren Relationalen Datenbanken, zu verschiedenen Zeitpunkten bestehend, vorstellen. Das bedeutet, dass sich Temporale Datenbanken, betrachtet man sie zu einem bestimmten Zeitpunkt, sich wieder auch normale Relationale Datenbanken zurückführen lassen. Dieser Vorgang entspricht einer Abbildung von der Zeit T auf die Eigenschaft eines oder mehrerer Objekte P. [6]
$sn : T \rightarrow P$

1.2 Kriterien Temporaler Datenbanken

Die Konzepte zu Temporalen Datenbanken unterscheiden sich in eingigen wesentlichen Punkten.

1.2.1 Datentypen zur Zeitstempelung

Zur Zeitstempelung bieten sich explizit Temporale Datentypen an, da sie von sich aus eine zeitliche Bedeutung besitzen und somit vor Fehlinterpretationen geschützt sind.

- Zeitpunkte (TIME, DATE, TIMESTAMP)
- Menge von Zeitpunkten
- Zeitdauer (INTERVAL)
- Intervall (PERIOD)

Zeitpunkte beziehen sich auf einen bestimmten Punkt auf dem Zeitstrahl. Die drei Datentypen TIME, DATE und TIMESTAMP unterscheiden sich in ihrer Auflösung, der kleinsten darstellbaren Einheit und in dem Bereich, in dem sie liegen können. [2]

1.2.2 Zeitstempelung

Zur Zeitstempelung der Einträge in Temporalen Datenbanken gibt es zwei Möglichkeiten:

- Tupelstempelung
- Attributstempelung

Bei der Tupelstempelung ist die Relationale Tabelle um zwei Spalten erweitert. Eine Spalte steht für die Startzeit, die zweite für die Endzeit. Von den oben vorgestellten Datentypen eignen sich für die Repräsentation die Zeitpunkte. Die Zeiten beziehen sich auf die gesamte Zeile, also auf die gesamte Entität und auf alle Werte des Tupels.
Ebenso könnte man für jedes einzelne Attribut zwei zusätzliche Spalten für die Zeitstempelung einführen. Dies ist dann Sinnvoll wenn die Änderungsfrequenz einzelner Objekte sehr groß ist, führt aber zu vielen Spalten, weshalb die Attributstempelung nicht verfolgt wird. [8] (S. 389 - 393)

1.2.3 Zeitdimension

Für die Bedeutung der Zeitstempel gibt es verschiedene Interpretationen.

- Gültigkeitszeitraum
- Systemzeitraum
- Gültigkeitszeitraum & Systemzeitraum

Der Gültigkeitszeitraum gibt an in welchem Zeitraum die Attributwerte gültig sind. Der Gültigkeitszeitraum eignet sich um Objekte abzubilden, dessen Attribute von der Zeit abhängen. Zum Beispiel kann der Preis eines Produktes zu verschiedenen Zeitpunkten unterschiedlich hoch sein. Die Höhe des Preises ist dabei vom Menschen festgelegt, genau wie die Länge der Gültigkeitszeiträume selbst. Es lässt sich sowohl die Vergangenheit als auch die Zukunft abbilden.

Der Systemzeitraum unterscheidet zwischen historischen Einträgen und solchen, die nach der Systemzeit aktuell sind. Er ermöglicht es Änderungen am Datenbestand nachvollziehen zu können. Bei Änderungen werden die aktuellen Einträge stets zu Historischen.

Somit ist es nur möglich die Gegenwart und die Vergangenheit abzubilden, nicht jedoch die Zukunft.

Bitemporale Datenbanken kombinieren den Gültigkeits- und den Systemzeitraum. Jedes Tupel besitzt vier zusätzliche Spalten, je zwei für Gültigkeits- und Systemzeitstempelung. Die Bedeutungen der Spalten sind für sich genommen die gleichen wie in ihrer einzelnen Verwendung. In der Kombination ist damit möglich Objekte abzubilden, die von der Zeit abhängen (Gültigkeitszeitraum) und zusätzlich Änderungen an der Abbildung nachzuvollziehen (Systemzeitraum). [8] (S. 70-80)

1.2.4 Automatische Normalisierung (Coalescing)

Das sogenannte Coalsescing bezieht sich auf Datenbanken, in denen die Gültigkeitszeit betrachtet wird und bezeichnet das Zusammenfassen von Zeilen mit gleichen Attributwerten und aneinandergrenzenden Gültigkeitszeiträumen bei Abfragen.
In einer Temporalen Tabelle kann es vorkommen, dass sich ein Attribut (zum Beispiel der Preis eines Artikels) über verschiedene Gültigkeitszeiträume verändert, andere Attribute desselben Objekts (wie zum Beispiel der Lagerort) aber konstant bleiben. Bei einer Abfrage nach dem konstanten Attribut fällt die Information über das zeitveränderliche Attribut weg. Trotzdem bleibt die Unterteilung in die verschiedenen Gültigkeitszeiträume zunächst bestehen und alle Zeilen, die über den erfragten Primärschlüssel verfügen werden zurückgegeben. Sie besitzen nun alle den gleichen Wert, aber unterschiedliche, teils aneinandergrenzende Gültigkeitszeiträume. Die Einträge mit gleichen Attributwerten und aneinandergrenzenden Gültigkeitszeiträumen zusammenzufassen nennt man nun Coalescing. [8] (S. 63)

1.3 Bestehende Konzepte

Historisch gibt es mehrere theoretische Konzepte wie man die zeitliche Dimension in Datenbanken berücksichtigen kann. Sie unterscheiden sich anhand der in 1.2 vorgestellten Kriterien. Auch ihre Philosophie und ihre gedachte Verwendung ist unterschiedlich. Tabelle 1.3.3 gibt einen Überblick über die Konzepte.

1.3.1 IXSQL

IXSQL stellt eine Erweiterung von SQL dar. Das Konzept zielt generell auf die Benutzung von Intervalldaten, nicht nur zeitlicher Natur ab. Mit speziellen Operatoren kann man von Punktwerten zu Intervalldaten gelangen. Auch die Umwandlung in die andere Richtung ist möglich. Allerdings läuft die Normalisierung nicht automatisch ab. [7]

1.3.2 TSQL2

TSQL2 ist ein Konzept speziell für zeitliche Daten. Es benutzt bitemporale Elemente (Eine Menge von einem oder mehreren Intervallen) zur Zeitstempelung. Somit ist auch hier automatisches Coalescing möglich. [7]

1.3.3 ATSQL2

ATSQL2 stellt streng genommen keinen eigenständiges Konzept dar. Vielmehr ist es ein Frontend für herkömmliche Relationale Datenbanken. Es zerlegt jede Anweisungen in

mehrere einzelne SQL-Anweisungen. [7]

	IXSQL	**TSQL2**	**ATSQL2**
Zeitdimension	Gültigkeitszeit	Bitemporal	Bitemporal
Datentyp zur Zeitstempelung	Intervall	Temporales Element	Intervall
Zeitstempelung	explizit	implizit	implizit
Zeitstempelung	Tupel	Tupel	Tupel
Automatisches Coalescing	Nein	Ja	Nein

2 Standardisierung in SQL:2011

Bei der Standardisierung des Umgangs mit Temporalen Datenbanken waren zwei Standardisierungsgremien mit ihren Untergruppen beteiligt: Auf der einen Seite das Internationale Standardisierungsgremium (ISO) mit der Untergruppe W3G, auf der anderen Seite des Atlantiks das Amerikanische Standardisierungsgremium (ANSI) mit der Untergruppe DM 32.2 in der auch Richard Smodgrass Mitglied war. Bereits 1994 unterbreitete Richard Smodgrass für die DM 32.2 der ANSI den Vorschlag das Konzept TSQL2 als Erweiterung des SQL-Standards mitaufzunehmen. Die W3G auf Seiten der ISO sah darin signifikante Probleme, wies den Vorschlag zurück und unterbreitete ihrerseits einen Gegenvorschlag. Die ANSI zeigte sich weder mit der Kritik an ihrem Vorschlag noch mit dem Gegenvorschlag der ISO einverstanden. So entstand eine Patt-Situation und das Thema um die Temporale Erweiterung kam vorerst nicht mehr auf den Tisch.
Erst 2008 wurde mit den Systemversionierten Tabellen erneut ein Vorschlag gemacht, der den Systemzeitraum berücksichtigt. 2010 folgte ein Vorschlag zu den Applcation-time-period Tables, die den Gültigkeitszeitraum mit abbilden. Beide Vorschläge basieren auf den ursprünglechen Vorschlägen von ISO und ANSI, allerdings wurd die Syntax teilweise stark geändert. 2011 nahmen die beiden Gremien die Vorschläge in den neuen Standard SQL:2011 auf. [9]

2.1 application-time-period tables (Gültigkeitszeitraum

CREATE legt bei den application-time-period tables zwei zusätzlich Spalten für den Gültigkeitszeitraum an. Die PERIOD FOR -Klausel gibt an welche Spalten den Gültigkeitszeitraum definieren und implementiert dabei auch die spätere automatsiche Überprüfung ob die Startzeit vor der Endzeit liegt. Optional kann mit WITHOUT OVERLAPS sichergestellt werden, dass sich die Zeiträume später nicht überlappen. Zusätzlich wird der Gültigkeitszeitraum dem Primärschlüssel hinzugefügt.

Bei INSERT gibt der Nutzer den Gültigkeitszeitraum explizit an. Lücken zwischen den Zeiträumen sind erlaubt.

Bei UPDATE speizifiziert die FOR PORTION -Angabe für welchen Zeitraum die Anweisung angewendet wird. Es kann ein Intervall angegeben werden, das deckungsgleich mit dem bestehenden Gültigkeitszeitraum ist. Dann wird der bestehende Eintrag aktualisiert. Liegt das Intervall zwischen dem bestehenden Gültigkeitszeitraum, dann zerteilt der Befehl den Eintrag in mehrere neue Einträge.

Der DELETE-Befehl funktioniert simultan zum UPDATE. Auch hier ist die FOR PORTION -Angabe möglich. Dabei kann es vorkommen, dass der Operator einen Bereich in der Mitte der Gültigkeitszeit herauslöscht. Übrig bleiben in diesem Fall zwei Einträge mit den Randzeiten.

Für SELECT existieren mehrere temporale Intervalloperatoren. Sie spezifizieren den Gültigkeitszeitraum, für den die Abfrage gestellt wird. [9]

2.2 Systemversionierte Tabellen (Gültigkeitszeitraum)

Das CREATE fügt bei systemversionierten Tabellen ebenfalls zwei zusätzlich Spalten hinzu. Hier repräsentieren sie den Systemzeitraum. Auch hier spezifiziert die PERIOD FOR -Klausel welche Spalten den Zeitraum definieren. Die Anfangssystemzeit kommt mit in den Primärschlüssel.

Der INSERT -Befehl unterscheidet sich in seiner Handhabung nicht gegenüber den klassischen Relationalen Datenbanken. Start- und Endzeit des Systemzeitraums werden automatisch vom System eingefügt. Daher auch der Begriff der impliziten Zeitstempelung.

Bei einem UPDATE wird stets die aktuelle Entität zu einer Historischen. Eine neue aktuelle Entität besitzt die geupdateten Attributwerte.

Das DELETE funktioniert wieder ähnlich wie die UPDATE -Operation. Die aktuelle Entität wird zu einer historischen. Ein neuer Eintrag kommt nicht hinzu. Somit besteht kein aktueller Eintrag mehr, was ja Sinn des Löschens ist.

Die FOR SYSTEM_TIME -Klausel spezifiert den Systemzeitraum bei der SELECT -Abfrage. Dabei gibt es mehrere Möglichkeiten den Systemzeitraum anzugeben oder einzugrenzen. Beispielsweise kann ein minimales Intervall angegeben werden, das der Systemzeitraum abdecken sollte (FROM ... TO ...) oder ein Intervall, das den Systemzeitraum enthält (BETWWEN ... AND ...). [9]

2.3 Biteporale Datenbanken

Bitemporale Datenbanken berücksichten sowohl den System- als auch den Gültigkeitszeitraum. Die Erweiterungen der Befehle werden entsprechend kombiniert.

Mit CREATE müssen vier zusätzliche Spalten, zwei für den System- und zwei für den Gültigkeitszeitraum angelegt werden.Die PERIOD FOR -Klausel findet zwei mal Verwendung, um jeweils die Spalten für den System- und den Gültigkeitszeitraum zu definieren. Die Klausel aktiviert auch hier die Prüfung ob die Startzeit vor der Endzeit liegt. Ebenso kommt die Anfangsgültigkeit in den Primärschlüssel.

Die Gültigkeitszeit muss beim INSERT wieder händisch angegeben werden; die Systemzeit kommt vom System automatisch.

Das UPDATE beeinflusst nur Entitäten mit aktueller Systemzeit. Wie bei den Systemversionierten Tabellen wird die Aktuelle dabei zu einer historischen Entität. Bei der geu-

pdateten neuen aktuellen Entität kann es aufgrund der FOR PORTION -Angabe für die Gültigkeitszeit ebenfalls zu Spaltungen kommen. Gleiches gilt für DELETE.

Der SELECT -Operator ist hier zweifach erweitert. Einmal um die temporalen Intervalloperatoren, die sich auf die Gültigkeitszeit beziehen und einmal um die FOR SYTEM_TIME -Klausel für die Systemzeit. [9]

3 Unterstüzung in aktuellen DBMS-Produkten

Momentan gibt es kein Datenbankmanagementsystem, das den temporalen Standard komplett unterstützt. Allerdings sind eine Reihe von Systemen am Markt, die über temporale Features verfügen und einen Teil des Standards umsetzen:

- Oracle (ab Version 12): Temporale Funktonen von SQL:2011
- IBM DB2 (ab V10): Temporale Daten, basierend auf SQL:2011
- PostgreeSQL (ab Version 9.2): Temporale Features
- Teradata (ab Version 14): TSQL2 Features
- Micosoft SQL Server (ab Version 16): Temporale Tabellen
- MariaDB (ab Version 10.3): Temporale Funktionen

[9] [1] [5] [10] [3]

3.1 IBM DB2

IBM DB2 war eines der ersten Datenbankmanagementsysteme, das die Verwendung von Systemversionierten Tabellen, application-time-period-tables sowie auch Bitemporale Datenbanken unterstützt. Deshalb soll auf dieses System näher eingegangen werden.
IBM DB2 existiert für Windows, UNIX, Linux (V11.1) und z/OS (V11). Zudem gibt es eine eingeschränkte Version für Windows und Linux. In dem Datenbankmanagementsystem können neben den Standard-Datentypen auch binäre Daten (Texte, Bilder, Video, Audio und XML) repräsentiert werden. Die Abragesprache ist SQL, mit entsprechenden Einbettungen in Programmiersprachen. Teilweise unterscheidet sich die Syntax jedoch vom SQL-Standard, zum Beispiel bei der Erweiterung FOR SYSTEM_TIME OF. [9] [4]

4 Anwendungsgebiete Tempraler Datenbanken

Es gibt zahlreiche Anwendungen, bei denen die Zeit mit im Datenbanksystem abgebildet werden soll.

Beim Data Warehousing werden Daten aus anderen Quellen gesammelt und in einem zentralen System gespeichert. Das Einlesen aus den externen Quellen geschieht zyklisch. Daher macht es Sinn diese Daten beim Einlesen mit einem Zeitstempel zu versehen. Damit hat man die Vorteile aus beiden Welten: Im Data Warehouse ist die Zeitinformation mitabgebildet und man kann zuordnen welcher Wert zu welcher Zeit gültig ist/war, während die externen Produktivsysteme ohne Zeitstempelung und damit weniger umfangreich

bleiben können.

Bei Prognosen und Analysen in der Wirtschaft interessieren zeitabhängige Daten aus der Vergangenheit beziehungsweise in der Zukunft. Daher ist der Einsatz von application-time-period tables sinnvoll.
In der Buchhaltung (Abrechnungen, Gehaltsabrechnungen, Rechnungsberichte) muss stets klar ersichtlich sein auf welcher Basis die Abrechnung erstellt wurde. Daher ist es sinnvoll die Systemzeit mit abzubilden. Zusätzlich gibt es Attribute, deren Wert zeitabhängig ist, wie zum Beispiel die Gehaltsklasse eines Angestellten, was die Repräsentation im Gültigkeitszeitraum erfordert. Somit bietet sich der Einsatz von Bitemporalen Datenbanken für alle Anwedungen rund um Buchhaltung an.

Bei Transaktionen ändert ein Gut häufig den Ort und den Besitzer. Um diese Änderungen darzustellen und später nachvollziehen zu können sind Systemversionierte Tabellen sinnvoll.

Bei der Aufzeichnung von Verkehrs- und Wetterdaten spielt die zeitliche Dimension ebenfalls eine Rolle. Werden die Daten direkt im System gespeichert, dann ist die Verwendung von Systemversionierten Tabellen möglich. Bei einer Latenz, besonders wenn diese unregelmäßig ist oder wenn ein nachträgliches Hinzufügen von Daten gewünscht ist, muss auf application-time-period tables zurückgegriffen werden.

Versicherungen und Banken unterliegen gesetzlichen Vorgaben zum Speichern von Daten. Versicherungen sind dazu verpflichtet gewisse Daten zur Nachverfolgung zu speichern. Kreditinstitute müssen dokumentieren auf welcher Basis Entscheidungen zum Beispiel über die Vergabe eines Kredits getroffen wurden. Beides kann in Systemversionierten Tabellen realiesiert werden, sofern keine weiteren zeitabhängigen Parameter vorkommen.

5 Fazit

Temporale Datebnanken berücksichtigen mit der zeitlichen Dimension etwas sehr Grundlegendes und stellen damit eine sinnvolle Erweiterung klassicher Relationaler Datenbanken dar.
Eine Standardisierung der Behandlung Temporaler Datenbanken war überfällig und ist mit SQL:2011 endlich realisiert. Die Standardisierung ermöglicht eine Portierung zwischen verschiedenen Systemen.
Kritik bezeihungsweise Wünsche, werden in Bezug auf den PERIOD-Datentyp und das Coalescing geäußert. Das Coalescing, das automatische Zusammfassen von wertgleichen Attributen mit benachbarten Zeitstempeln, ist derzeit nirgends implementiert, wäre aber ein Plus an Übersichtlichkeit. Auch der PERIOD-Datentyp ist im Standard SQL:11 nicht enthalten. Er ist als Zeitintervall mit einer Menge von aufeinander folgenden Zeitpunkten definiert und kann auf natürliche Weise verwendet werden um Intervalle darzustellen und sie zu vereinigen. Die Implementierung dieses Datentyps würde ein einfaches Coalescing ermöglichen. [9]
Allerdings muss erwähnt werden, dass momentan kein Datenbankmanagementsystem den Standard vollständig und komplett umsetzt. Obwohl erste Konzepte seit mehr als 25 Jahren existieren sind die Temporalen Datenbanken wohl noch nicht am Ende ihrer Entwicklung angekommen. Insgesamt wird ihre Bedeutung und Anwendung in den nächsten

Jahren durch die fortschreitende Digitalisierung und Automatisierung steigen.

Literatur

[1] Multi-temporal Features in Oracle 12c - Philipp Salvisberg's Blog, Jan 2014. [Online; accessed 2. Jul. 2019].

[2] *Temporale Datentypen*, Oct 2014. [Online; accessed 22. Jun. 2019].

[3] MariaDB to Update its Server with Temporal Data Processing, Feb 2018. [Online; accessed 2. Jul. 2019].

[4] IBM Db2 Systemeigenschaften, Jul 2019. [Online; accessed 2. Jul. 2019].

[5] SQL2011Temporal - PostgreSQL wiki, Apr 2019. [Online; accessed 2. Jul. 2019].

[6] Christiane Dollinger. Temporal datbases - concepts. Technical report, Alpe-Adria Universität, Pörtschach, 2006.

[7] Alexander Kaiser. Neuere entwicklungen auf dem gebiet temporaler datenbanken - eine kritische analyse. Technical report, Universität Wien, Wien, 1998.

[8] Thomas Myrach. *Temporale Datenbanken in betrieblichen Informationssystemen : Prinzipien, Konzepte, Umsetzung*. Vieweg+Teubner Verlag, Wiesbaden, 2005.

[9] D. Petkovic. Was lange währt, wird endlich gut: Temporale daten im sql-standard. *Datenbank Spektrum*, 13(2), 2013.

[10] Koen Verbeeck. Introduction to SQL Server Temporal Tables. *MSSQLTips*, Sep 2018.