HOCHSCHULE FÜR TECHNIK, WIRTSCHAFT UND KULTUR
LEIPZIG

FAKULTÄT - INFORMATIK, MATHEMATIK UND
NATURWISSENSCHAFTEN

MASTER MEDIENINFORMATIK

ABSTRACT

Temporale Datenbanken

Name: Maxim Osipov

Dozent: Prof. Dr.-Ing. Thomas Kudrass
Veranstaltung: Oberseminar Datenbanksysteme

Halle (Saale), 5. Juli 2017

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Forschungen in dem Gebiet der temporalen Datenbanken werden erst seit etwa 25 Jahren durchgeführt. Am Ende der 80er und in den 90er Jahren wurden viele Veröffentlichungen und Konzepte zu diesem Thema gemacht. Es wurden z.B. die Sprachvorschläge IXSQL, TSQL2, SQL/TP entwickelt. Die zwei Vorschläge: TSQL2 und SQL/TP sind bis heute noch wichtig. Sie wurden als SQL-Erweiterungen integriert. In der SQL-92 wurde versucht, die ersten temporalen Datentypen und Vergleichsoperatoren einzuführen. Ab der Version SQL-2011 wurde die Zeitunterstützung standardisiert. Am Anfang der 2010er wurde, mit dem Beginn von großem Interesse von Unternehmen zu dem Themengebiet von "Data Warehouse" und "Business Intelligence", bei vielen Datenbankmanagementsystem-Anbietern die temporalen Features zur Verfügung gestellt, z.B. IBM DB2 Version 2 hat die s.g. "Time Travel Queries" (s. [IBM17]). Das ist das ersten Datenbankmanagementsystem, welches konforme Implementierung "temporale Futures" aus SQL-2011 hat.

Bei vielen Anwendungsfällen ist es erforderlich, nicht nur Informationen über Objekte, sondern auch die zeitlichen Änderungen dieser Informationen, über ihren Gültigkeitszeitraum hinaus, festzuhalten. Beim Löschen oder Ändern von Daten müssen Überspeicherungen verhindert werden. Konventionelle Datenbanksysteme sind nur beschränkt geeignet, diese Datenhaltung zu organisieren. Sie bieten nur eine Momentaufnahme der Daten. Außerdem wird die zeitliche Entwicklung nicht unterstützt. Existierende Sprachkonstrukte sind bei Recherchen über Zeitverläufe nicht ausreichend. Damit ist die Forschung etwa über neue Datenmodelle, Datenbanksprachen oder anzulegende Speicherstrukturen in diesem Themengebiet notwendig (vgl. [Ste01], S. 1).

1.2 Definition

Die temporalen Datenbanken bieten einen einheitlichen und systematischen Umgang mit historischen bzw. chronologischen Daten, d.h. den Daten, welche auf vergangene und möglicherweise zukünftige Zeiträume bezogen sind.

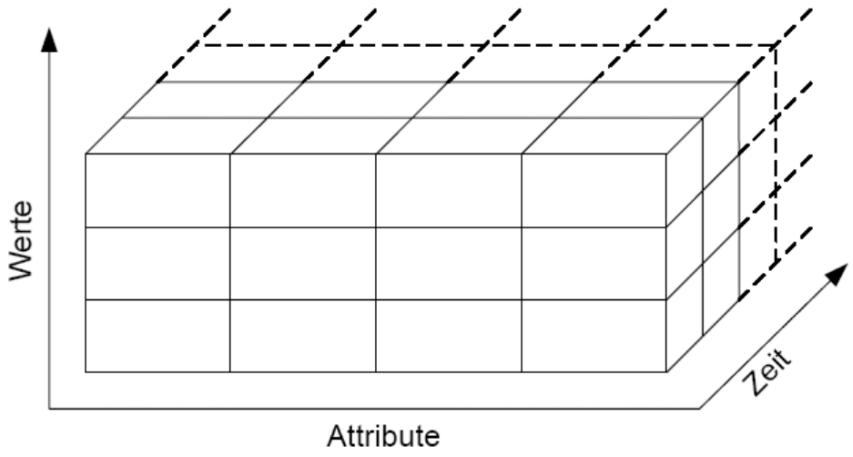
1.3 Anwendung

Die Anwendungsmöglichkeiten von temporalen Datenbanken in der Praxis sind vielfältig. Diese Datenbanken sind besonders notwendig in den Wirtschaftswissenschaften, z.B. für Prognosen, Analysen, Strategieplanung, Rechnungsberichte, Anwendungen mit zeitlichen Aspekten, Buchhaltungssysteme, Transaktionsüberwachung, sowie in den Gebieten von "Data Warehouse" und "Business Intelligence". Außerdem sind sie wichtig für die Untersuchung von historischen und medizinischen, sowie von Verkehrsdaten.

2 Temporale Datenbankmodellen und deren Eigenschaften

2.1 Grundlagen

Das allgemeine Konzept ist die Erweiterung der relationalen Datenbank um eine Zeitdimension. Es kann Temporale Relation als ein dreidimensionales Objekt dargestellt werden, welches aus einer Wert-, Attribut- und Zeitachse besteht:



[Tao04]

Es gibt folgende Zeitbegriffe bei den temporalen Datenbanken:

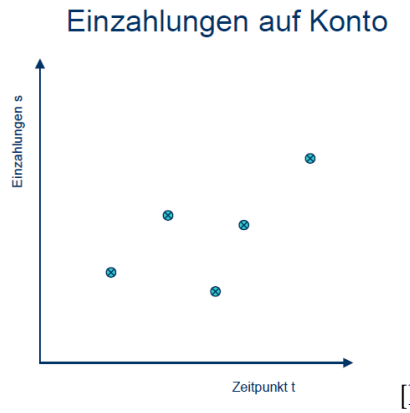
- **valid time** (Gültigkeitszeit): die Zeit, wenn der Datenwert wirksam wird - gültig (Gültigkeit aus Sicht der Anwendung);
- **transaction time** (Transaktionszeit/Aufzeichnungszeit): die Zeit, wenn der Datenwert in die Datenbank eingetragen wird. Der Zeitpunkt der Eintragung eines Tupels in die Datenbank "bekannt" aus Sicht des Systems;
- **bitemporale Zeit:** Beides wird gleichzeitig repräsentiert;
- **user defined time** (benutzerdefinierte Zeit): Die Zeit, welche vom Benutzer definiert und interpretiert wird.
- **andere Zeiten:** z.B. Beobachtungszeit

Es existieren folgende am häufigsten verwendeten Datentypen temporaler Attribute:

- Zeitpunkt (TIME,DATE,TIMESTAMP),
  z.B. 28.06.2017, 22:23:59, (28.10.2015, Klausur)
- Mengen von Zeitpunkten oder Zeitmarken,
  z.B. (Sep, Okt, Nov, Dez)
- Zeitintervalle (PERIOD-TSQL2.0),
  z.B. [28.10.1990, 28.06.2017]
- Zeitdauer (INTERVAL), z.B. 7 Tage
- temporale Elemente (eine Zusammenfassung beliebig vieler Intervalle), z.B. (28.10., 28.11., 28.12., Projektvorstellung)

Die verschiedenen Informationsarten können folgende Zeitfolgen haben:

Punktereignisse oder Fakten

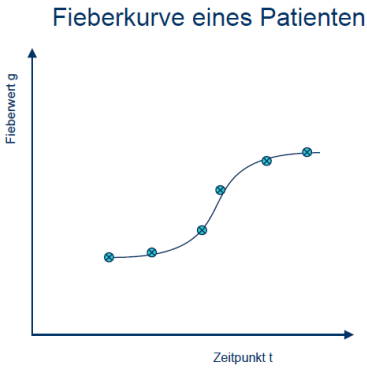


[Han05]

Dauerereignisse oder Fakten

[Han05]

Kontinuierliche Ereignisse oder Fakten



[Han05]

2.2 Temporale Datentypen

Es existieren verschiedene temporale Datentypen, mit deren Hilfe sich einfache temporale Gegebenheiten in SQL-92 darstellen lassen können.

Snapshot-Modell

Dieses Modell enthält keine explizite Zeitinformation. Es betrachtet die Daten zu einem fixierten Zeitpunkt für die Gültigkeits- und ggf. auch die Transaktionszeit. Das Snapshot-Modell bezieht sich meist auf die aktuelle Zeit und wird mit der klassischen Datenbank realisiert. Bei diesem Modell ist es möglich, zu einem bestimmten Zeitpunkt, zu bestimmen, welche Eigenschaft ein Objekt/Objekte gerade hat/haben. Diesen Prozess nennt man **Snapshot** und dieser kann mit folgender Funktion **sn** : $T \rightarrow$ P abgebildet werden, wo T eine Zeiteingabe und P die Eigenschaft eines Objektes ist.

Gültigkeitszeitmodell

Das Modell von Jones ist im Rahmen des LEGOL2-Projektes im Jahre 1979 entstanden. Bei diesem Modell wird die Gültigkeitszeit verwendet (valid time). Die Modellierung erfolgt durch die zwei Zeitattribute "Start-" und "Endzeitpunkt", welche ein Intervall definieren in dem der Fakt seine Gültigkeit besitzt. Die Angabe der Zeitwerte ist explizit, d.h. sie wird durch den Benutzer eingegeben (vgl. [Ste01], S. 8). In der folgenden Abbildung ist das Beispiel dafür dargestellt:

Name	Primärschlüssel	Gehalt	Gültigkeitsstartzeit	Gültigkeitsendzeit
Müller	123456789	1000,00	01.01.2004	31.12.2004
Müller	123456789	1200,00	01.01.2005	20.03.2005
Müller	123456789	1500,00	21.03.2005	Now
Friedrich	777778888	3000,00	01.01.2003	Now

[Han05]

Aufzeichnungszeitrelation/Transaktionszeitmodell

Es wird die exakte Zeit der Operation vermerkt, wann ein Fakt in der Datenbank gespeichert, geändert oder gelöscht wurde. Dieser Zeitstempel ist implizit, direkt von der Systemuhr bestimmt und vom Benutzer nicht beeinflussbar. In der folgenden Abbildung ist das Beispiel dazu entworfen:

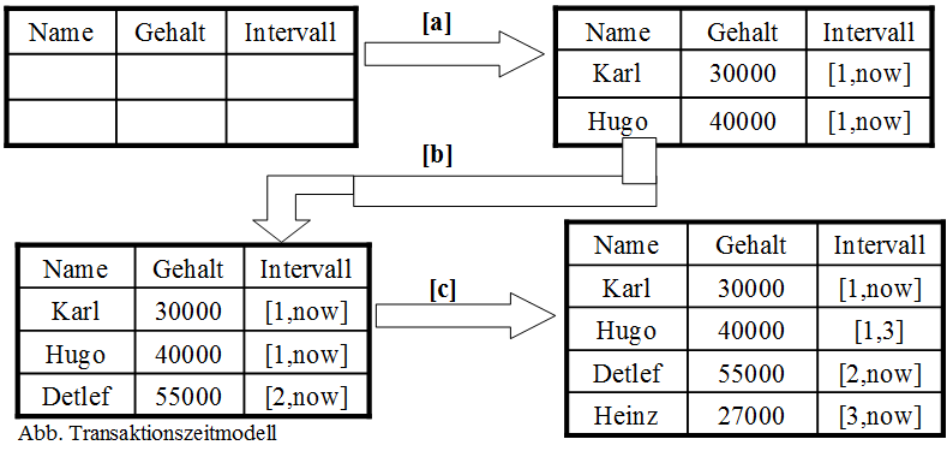


Abb. Transaktionszeitmodell

[Ste01]

[*a*] : Zu Beginn war die Datenbank leer, dann wurden zum Zeitpunkt 1 Hugo und

Karl hinzugefügt. Die Transaktionszeit ist auf die Gegenwart beschränkt. Die Variable ”now” drückt aus, dass der Fakt bis zum jetzigen Zeitpunkt aktuell ist.

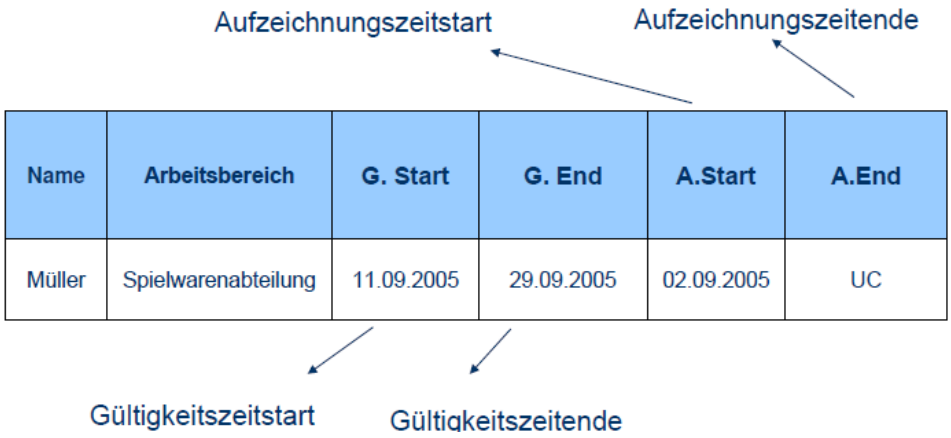
[b] : Zum Zeitpunkt 2 ist Detlef hinzugefügt worden, was eindeutig an seinem Zeitstempel [2, now] abgelesen werden kann.

[c] : Am Ende wird zum Zeitpunkt 3 Heinz hinzugefügt und Hugo gelöscht.

In der Praxis: Alle veränderten/gelöschten Informationen bleiben auch weiterhin in der DB verzeichnet, daher ist dieses Modell sehr für Write-Once-Datenträger geeignet (vgl. [Ste01], S. 8-9).

Bitemporales Modell

Dieses Modell besitzt sowohl Gültigkeitszeit als auch Transaktionszeit/Aufzeichnungszeit. In der folgenden Abbildung ist das Beispiel dazu dargestellt:



[Han05]

Mithilfe von diesem Modell können die Zeitverhältnisse bestimmt werden. Es ist möglich die Geschichte des Objektes bzw. Informationen über einen bestimmten Zeitraum nachzuvollziehen (vgl. [Ste01], S. 10).

3 Temporale Datenbankmodellen und dazugehörige Datenbanken

Allgemein unterscheiden sich die temporalen Datenbanken in den unterstützten Zeitdimensionen und in der Art der Repräsentation temporaler Daten. Sie besitzen noch weitere Eigenschaften. Die Zeitstempel können entweder explizit sein, d.h. von außen sichtbar oder implizit mit den Aussagen verbunden. Die Zeitstempelung kann für jedes Attribut einzeln oder für eine gesamte Datenzeile (Tupel) erfolgen. Manche Datenbankmanagementsysteme führen nach entsprechenden Aktualisierungsoperationen automatisch eine Zusammenfassung von Zeiträumen mit übereinstimmenden Attribut-Werten durch. Diese Zusammenfassung von Zeiträumen heißt "Coalescing"(automatische temporale Normalisierung) (vgl. [Tao04], S.21).

3.1 Sprachvorschläge und Konzepte

BCDM (bitemporal conceptual data model) ist das bekannteste bitemporale Modell. Dies ist ein erweitertes relationales Datenmodell. Das Modell ist Teil und Grundlage des Sprachvorschlages TSQL2, welcher 1997 von R. Snodgrass, J. Clifford und C. Jensen konzipiert wurde. BCDM hat folgende Eigenschaften:

Zeitdimension:	bitemporal
Datentyp zur Zeitstempelung:	Temporales Element
Zeitstempelung explizit/implizit:	implizit
Tupel- / Attributzeitstempelung:	Tupel
Automatisches Coalescing:	Ja

Im Juni 1993 formte sich nach einem "Workshop on an Infrastructure for Temporal Databases" in Arlinton, Texas ein TSQL2 Komitee mit dem Ziel, eine Sprachspezifikation einer temporalen Anfragesprache zu verfassen. Im Januar 1993 wurde eine erste Version veröffentlicht und die endgültige Version folgte im September 1994 (vgl. [Tao04], S. 26). Bei TSQL2 wurde ein Zeitstempelattribut hinzugefügt, das Mengen von bitemporalen Chronons enthält, und zwar die Zeitpunkte, wann ein Fakt gültig war und wann er eingetragen wurde. Dieser Sprachvorschlag unterstützt die Evolution der relevanten Objekte und damit der ihnen zugeordneten Daten im Zeitablauf. Der Sprachvorschlag "SQL/Temporal" orientiert sich an den Forschungspapieren von TSQL2. Die Konzepte von TSQL2 wurden nicht

vollständig übernommen, weil z.B. die Diskussion, die Unterstützung von Gültigkeitszeit und Transaktionszeit in SQL/Temporal zu integrieren, entstand (vgl. [Tao04], S. 27).

3.2 SQL-92 und SQL-2011

SQL-92

Seit SQL-92 wurden schon folgende temporale Datentypen und Klauseln angeboten:

- DATE
- TIME
- TIMESTAMP als Kombination der beiden erstgenannten
- INTERVAL ist für die Zeitdauer (keine Zeitintervalle, die durch zwei Zeitpunkte begrenzt sind !)
- PERIOD für den Zeitintervall-Datentyp;
- INTERVAL(PERIOD) : Unwandlung einer Periodenangabe in eine Zeitdauer;
- INTERSECT(PERIOD1, PERIOD2) : Ermittlung der Schnittmenge zweier Perioden;
- BEGIN(PERIOD) für den Anfangszeitpunkt einer Periode;
- END(PERIOD) für den Endzeitpunkt einer Periode;
- ALTER TABLE: um Zeitdimensionen nachträglich einzufügen, oder zu entfernen.

Es wurden auch spezielle Werte zur Definition der Periodenbegrenzung definiert:

- **'beginning'** steht für die unendlich weit zurückliegende Vergangenheit;
- **'forever'** ... unendlich weit entfernte Zukunft;
- **'now'** ... für die aktuelle Zeit;
- **PERIOD '[beginning, -forever]'** stellt den maximal in einem System abbildbaren Zeitraum dar.

Zeitangaben werden durch die beiden Operatoren CAST und SCALE in andere Granularitäten umgewandelt. CAST liefert immer ein konkretes Ergebnis. SCALE liefert unbestimmte Ausdrücke, wenn eine Umwandlung in ein feineres Granulat vorgenommen werden soll. Die allgemeine Form wurde wie folgt definiert:
$[CAST|SCALE](< operand > AS < granularity >)$

Außerdem wurden temporale Vergleichsoperatoren eingeführt:

- **PRECEDES:** die Vorzeitigkeit einer Periode gegenüber einer anderen Periode ausdrücken;
  Bsp.: $X\ PRECEDES\ Y$, dann ist $END(X) < BEGIN(Y)$;
- **MEETS:** Spezialfall von $PRECEDES$, bei dem $END(X)$ und $BEGIN(Y)$ genau ein Chronon auseinander liegt.
- **OVERLAPS:** da wird ermittelt, ob die Schnittmenge zweier Perioden nicht leer ist.
- **CONTAINS:** ist ein Spezialfall, bei dem eine Periode vollst. in einer anderen Periode enthalten sein muss.

Bei den temporalen Tabellen erfolgt die Datendefinition mithilfe von CREATE-, die Datenmodifikation mithilfe von INSERT-, UPDATE-, DELETE- und die Datenabfragen mithilfe von einer SELECT-Klausel.

SQL-2011

Bei der SQL-2011 wurden die sogenannten temporalen Features eingeführt. Die Zeitunterstützung wurde standardisiert. Folgendes wurde eingeführt:

- **application-time period table:** anwendungsversionierte Tabellen;
- **system-versioned tables:** systemversionierte Tabellen für Transaktionszeit;
- **bitemporal tables:** anwendungs- und systemversionierte Tabellen;

Anwendungsversionierte Tabellen

Die Anwendungszeit/Gültigkeitszeit wird bei diesen Tabellen durch Nutzer bestimmt. Es werden Zeitintervalle (PERIOD) mit Start- und Endzeitpunkt vom Typ DATE oder TIMESTAMP verwendet. Dabei werden halboffene Intervalle (Startzeitpunkt ist Teil des Intervalls, Endzeitpunkt nicht) verwendet. Der Primärschlüssel erfordert die Hinzunahme des Zeitintervalls. Die Fremdschlüssel müssen für die Bezugsintervall korrekt sein (vgl. [Erh17], S. 35).

Systemversionierte Tabellen

Bei diesen Tabellen erfolgt eine automatische Erzeugung und Anpassung von Zeitintervallen bezüglich der Änderungszeit. Die Attribute für Start- und Endzeitpunkte sind obligatorisch. Es können nur aktuelle (nicht historische) Versionen von Tupeln geändert/gelöscht werden. Die Integritätsbedingungen werden nur auf aktuellen Tupelversionen überwacht. Nur ein Datenbankmanagementsystem kann Start- und Endzeitpunkte für Systemzeitperioden liefern.

Beim INSERT erfolgt folgendes:

- Systemzeitstart wird auf Transaktionszeit gesetzt;
- Systemzeitende wird auf Maximalwert gesetzt;

Bei den UPDATE- und DELETE-Klauseln werden die Operationen nur auf aktuellen Zeilen durchgeführt. Es werden automatisch historische Zeilen für jede geänderte oder gelöschte Zeile erzeugt. Bei UPDATE wird der neue Startzeitpunkt auf die Transaktionszeit gesetzt und bei DELETE wird der Endzeitpunkt auf den Transaktionszeitpunkt gesetzt (vgl. [Erh17], S. 38-39).

Bitemporale Tabellen

Die Anwendungszeit und Systemzeit ist bei diesen Tabellen kombiniert. Die Anfragen können Prädikate für Systemzeitperioden und Anwendungszeitperioden enthalten.

neue Vergleichsprädikate für PERIOD-Intervalle:

CONTAINS, OVERLAPS, EQUALS, PRECEDES, SUCCEEDS (Eine Periode beginnt später als eine andere.), IMMEDIATELY PRECEDES (Eine Periode beginnt früher als eine andere und geht unmittelbar in diese über), IMMEDIATELY SUCCEEDS (Eine Periode beginnt später als eine andere und setzt sie unmittelbar fort.)

neue Suchprädikate:

- FOR SYSTEM_TIME **AS OF** <datetime value expression>
- FOR SYSTEM_TIME **BETWEEN** <datetime1> **AND** <datetime2>

- FOR SYSTEM_TIME **FROM** <datetime1> **TO** <datetime2>

4 Unterstützung der temporalen Datenbanken in Datenbankmanagementsystemen

- **IBM DB2 Version 2** hat die s.g. "Time Travel Queries". Das ist die erste DB, welche konforme Implementierung "temporale Features" aus SQL-2011 hat. Obwohl sie manchmal eine andere Syntax verwendet, z.B. FOR SYSTEM_TIME AS OF
- **Oracle 12c** unterstützt die temporale Funktionalität in Übereinstimmung mit SQL-2011. Dieses System hat auch manchmal eine andere Syntax, z.B. AS OF TIMESTAMP.
- **Microsoft SQL Server:** hat temporale Tabellen mit SYSTEM_VERSIONING implementiert.
  Derzeit existiert kein kommerzielles DBMS, das die Anforderungen der temporalen Datenhaltung vollständig abbildet.
- **PostgreSQL** bietet ab Version 9.2 viele der temporalen Features an;
- **Teradata** bietet ab Version 13.10 temporale Features und ab Version 14 TSQL2 Features an;

4.1 IBM DB2 Version 2 - Demonstration

Time Travel Queries (TTQ) hat folgende Tabellen zur Verfügung:

- Temporale Tabellen für den Systemzeitraum: *History tables, SYSTEM_TIME period*
- Temporale Tabellen für den Anwendungszeitraum: *BUSINESS_TIME period*
- Bitemporale Tabellen

Temporale Tabellen für Systemzeitraum:

– eine Tabelle mit System Time definieren:

```
CREATE TABLE policy (
id              INT primary key not null,
vin             VARCHAR(10),
annual_mileage  INT,
```

```
rental_car      CHAR(1),
coverage_amt    INT,
sys_start       TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL,
sys_end         TIMESTAMP(12) GENERATED ALWAYS AS ROW END NOT NULL,
trans_start     TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID
IMPLICITLY HIDDEN,
PERIOD SYSTEM_TIME (sys_start, sys_end)
);
```

– History-Tabelle definieren:

```
CREATE TABLE policy_history LIKE policy;
```

– Versionierung für die Basis-Tabelle aktivieren:

```
ALTER TABLE policy ADD VERSIONING USE HISTORY TABLE policy_history;
```

– Dateneingabe in eine Tabelle mit System Time

```
INSERT INTO policy(id, vin, annual_mileage, rental_car, coverage_amt)
VALUES(1111, 'A1111', 10000, 'Y', 500000);

INSERT INTO policy(id, vin, annual_mileage, rental_car, coverage_amt)
VALUES(1414, 'B7777', 14000, 'N', 750000);
```

– Aktualisieren von Daten in einer Tabelle mit System Time

```
UPDATE policy
SET coverage_amt = 750000
WHERE id = 1111;
```

– Löschen von Daten aus einer Tabelle mit System Time

```
DELETE FROM policy WHERE id = 1414;
```

–Abfragen zu den History-Tabelle:

```
SELECT coverage_amt
```

```
FROM policy FOR SYSTEM_TIME AS OF '2010-12-01'
WHERE id = '1111';

SELECT count(*)
FROM policy FOR SYSTEM_TIME FROM '2011-11-30'
TO '9999-12-30'
WHERE vin = 'A1111';
```

- *FOR SYSTEM_TIME AS OF:* gibt Daten ab einem bestimmten Zeitpunkt zurück;
- *FOR SYSTEM_TIME FROM .. TO ...:* gibt Daten für einen bestimmten Zeitraum zurück. Der angegebene Startzeitpunkt liegt innerhalb des Zeitraums, der Endzeitpunkt nicht;
- *FOR SYSTEM_TIME BETWEEN ... AND ...:* gibt Daten zurück, die zwischen bestimmten Start- und Endzeiten liegen. Der angegebene Startzeitpunkt und der Endzeitpunkt liegen innerhalb des Zeitraums.

Temporale Tabellen für Anwendungszeitraum:

–eine Tabelle mit Business Time definieren:

```
CREATE TABLE policy (
id              INT NOT NULL,
vin             VARCHAR(10),
annual_mileage  INT,
rental_car      CHAR(1),
coverage_amt    INT,
bus_start       DATE NOT NULL,
bus_end         DATE NOT NULL,
PERIOD BUSINESS_TIME(bus_start, bus_end),
PRIMARY KEY(id, BUSINESS_TIME WITHOUT OVERLAPS) );
```

– Einfügen von Daten in eine Tabelle mit Business Time:

```
INSERT INTO policy
VALUES(1111,'A1111',10000,'Y',500000,'2010-01-01','2011-01-01');
INSERT INTO policy
```

```
VALUES(1111,'A1111',10000,'Y',750000,'2011-01-01','9999-12-30');
INSERT INTO policy
VALUES(1414,'B7777',14000,'N',750000,'2008-05-01','2010-03-01');
INSERT INTO policy
VALUES(1414,'B7777',12000,'N',600000,'2010-03-01','2011-01-01');
```

–Aktualisieren von Daten in eine Tabelle mit Business Time:

```
UPDATE policy
FOR PORTION OF BUSINESS_TIME FROM '2010-06-01' TO '2011-09-01'
SET coverage_amt = 900000
WHERE id = 1111;
```

–Löschen von Daten aus einer Tabelle mit Business Time:

```
DELETE FROM policy
FOR PORTION OF BUSINESS_TIME FROM '2010-06-01' TO '2011-01-01'
WHERE id = 1414;
```

5 Literatur- und Linkverzeichnis

[Erh17] ERHARD RAHM: *3. Objekt-relationale DBS*. 2017. – https://dbs.uni-leipzig.de/file/dbs2-ss17-kap3.pdf; abgerufen im Juni 2017.

[Han05] HANNO EICHELBERGER AND ALEXANDER RUDER AND KHALID ADDAKIRI: *Temporale Datenbanken*. 2005. – www.chrisix.net/fhhn/ss05/4tempdb.pdf; abgerufen im Juni 2017.

[IBM17] IBM KNOWLEDGE CENTER: *Time Travel Query using temporal tables*. 2017. – https://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0058476.html; abgerufen im Juni 2017.

[Ste01] STEFAN SCHEIDEWIG AND DANIEL LEMPE: *Temporale Erweiterungen des relationalen Datenmodells*. 2001. – www.informatik.uni-jena.de/dbis/lehre/ws2000/proseminar/.../scheidewigLempe.doc; abgerufen im Juni 2017.

[Tao04] TAOU HASSANI: *Konzepte temporaler Datenbanken*. 2004. – homepages.thm.de/hg10013/Lehre/MMS/SS04/hasani/TDBReferat.pps; abgerufen im Juni 2017.