In: Zimmermann, Harald H.; Schramm, Volker (Hg.): Knowledge Management und Kommunikationssysteme, Workflow Management, Multimedia, Knowledge Transfer. Proceedings des 6. Internationalen Symposiums für Informationswissenschaft (ISI 1998), Prag, 3. – 7. November 1998. Konstanz: UVK Verlagsgesellschaft mbH, 1998. S. 329 – 343

Neuere Entwicklungen auf dem Gebiet temporaler Datenbanken - eine kritische Analyse

Dr. Alexander Kaiser

Abteilung Angewandte Informatik
Institut für Informationsverarbeitung- und wirtschaft
Wirtschaftsuniversität Wien
Augasse 26, 1090 Wien
EMail: alexander.kaiser@wu-wien.ac.at

Inhalt:

Zusammenfassung:

Der vorliegende Beitrag skizziert die wichtigsten temporalen Datenmodelle und temporalen Datenbanksprachen, vergleicht sie miteinander und beurteilt sie in Bezug auf Vollständigkeit, Aufwärtskompatibilität zum relationalen Datenmodell und Verfügbarkeit in der Praxis. Diesen Modellen wird ein Ansatz gegenübergestellt, der ohne das relationale Modell zu erweitern eine effiziente Verwaltung zeitbezogener Daten vorstellt. Insbesondere wird in diesem Ansatz auf den Entwurf von Datenbanken mit zeitbezogenen Daten im Detail eingegangen.

1. Einführung und Problemstellung

Temporale Datenbanken sind in den letzten Jahren zu einem bedeutenden Forschungsbereich auf dem Gebiet der Datenbankforschung geworden. Das zunehmende Interesse an temporalen Datenbanken hat mehrere Ursachen. Zum einen ist ein verstärktes Interesse am Thema der Zeit an sich zu beobachten, was unter anderem auch mit dem nahenden Jahrtausendwechsel in Zusammenhang stehen dürfte. Zum anderen ist die Zeit ein wesentlicher Aspekt der realen Welt. Datenbanksysteme sollen letztendlich eine möglichst genaue Abbildung der modellierten Realität darstellen. Spielt die Zeit in der Realität eine bedeutende Rolle, so muß die Zeitdimension daher auch in Datenbanksystemen angemessen berücksichtigt werden. Darüber hinaus hat in den letzten Jahren auch das Konzept des Datawarehouse stark an Bedeutung gewonnen. In Datawarehousesystemen spielt die Zeitdimension eine sehr wichtige Rolle, so daß auch von diesem Bereich her das Forschungsgebiet der temporalen Datenbanken wichtige Impulse erhalten hat.
Ziel des vorliegenden Beitrags ist es, die wichtigsten zeitbezogenen Datenmodelle und Datenbanksprachen kritisch zu beleuchten und sie einem vom Autor der Arbeit entwickelten Ansatz gegenüber zu stellen, der das weit verbreitete relationale Modell als Datenmodell verwendet.

2. Temporale Datenmodelle und temporale Datenbanksprachen

In temporalen Datenbanken wird zwischen der Gültigkeitszeit, das ist die Zeit, die festlegt, in welchem Zeitraum ein Objekt in der modellierten Realität den beschriebenen Zustand aufweist, und der Transaktionszeit, das ist der Zeitpunkt, zu dem ein Datensatz in die Datenbank aufgenommen bzw. geändert oder gelöscht wird, unterschieden. In weiterer Folge der Arbeit beschränken wir uns schwerpunktmäßig auf die Betrachtung der Gültigkeitszeit. Eine weitere wichtige Unterscheidung besteht zwischen der Attributzeitstempelung und der Tupelzeitstempelung. Bei der Attributzeitstempelung wird einem einzelnen Attribut ein Zeitstempel zugeordnet. Dadurch sind die Werte eines Attributs nicht mehr atomar und entsprechen nicht mehr der 1.Normalform. Bei der Tupelzeitstempelung wird einem gesamten Tupel ein Zeitstempel zugeordnet, so daß alle Attributsausprägungen dieses Tupels in diesem Zeitraum in der modellierten Realität gültig sein müssen.
Im folgenden Abschnitt der Arbeit werden einige Modelle kurz vorgestellt. Ziel ist dabei aber keine detaillierte Beschreibung der Modelle, sondern eine Gegenüberstellung der wesentlichen Charakteristika der Modelle und eine Beurteilung der Modelle in Bezug auf Vollständigkeit, kommerzielle Verfügbarkeit und Praktikabilität.

3. Ausgewählte temporale Datenmodelle

Im folgenden werden für jedes der ausgewählten Datenmodelle die jeweils typischen Charakteristika näher beschrieben, also vor allem diejenigen Aspekte, in denen sich das jeweilige Datenmodell von den übrigen Datenmodellen unterscheidet. Für eine vollständige Beschreibung der Datenmodelle und Datenbanksprachen sei auf die jeweils angegebene Literatur verwiesen. Die Auswahl der Datenmodelle erfolgte nach den Aspekten Aktualität (ATSQL2,

SQL/Temporal), Repräsentanten der Tupelzeitstempelung (TSQL2, TSQL, IXSQL, ATSQL2, SQL/Temporal) und Repräsentanten der Attributzeitstempelung (SQL^T).

3.1 Das BCDM und TSQL2

Das BCDM (Bitemporal Conceptual Data Model) und die darauf basierende Sprache TSQL2 wurde von einer größeren Gruppe von Forschern entwickelt, die sich mit temporalen Daten beschäftigen (Snodgrass et al. 95). Das BCDM ist ein bitemporales Datenmodell mit Tupelzeitstempelung, d.h. es unterstützt sowohl die Gültigkeitszeit als auch die Transaktionszeit. Das BCDM sieht ein implizites Zeitstempelattribut vor und steht damit im Gegensatz zu den meisten anderen tupelorientierten Modellen, bei denen die Zeitstempelattribute explizite Attribute sind. Durch die implizite Tupelzeitstempelung wird auch angenommen, daß alle Attribute eines Tupels zeitabhängig sind und sich (im Idealfall) auch gleichzeitig ändern sollten. TSQL2 ist aufwärtskompatibel zu SQL92.
Neben den in SQL92 vorhandenen Datentypen sieht TSQL2 auch den Datentyp `PERIOD` vor, mit dem eine Periode von einem Beginnzeitpunkt bis zu einem Endzeitpunkt abgebildet werden kann. Der Datentyp `PERIOD` entspricht im wesentlichen dem Datentyp `INTERVAL OF DATE` in IXSQL. Der Datentyp `PERIOD` findet sich auch in SQL/Temporal des SQL3 Standardentwurfs wieder. Für den neuen Datentyp `PERIOD` werden spezielle Vergleichsoperatoren eingeführt, die eine Untermenge der 13 Zeitoperatoren von Allen (Allen 83) bilden (Myrach 97). Durch den Zusatz `STATE` bzw. `EVENT` bei der `CREATE TABLE` Anweisung in TSQL2 kann man explizit zwischen Zustands- bzw. Ereignis-Tabellen wählen. In Kombination mit der Angabe, ob es sich um eine Schnappschußtabelle (also eine konventionelle Tabelle ohne zeitbezogene Daten), eine bitemporale Tabelle, eine Gültigkeitszeit-Tabelle oder eine Transaktionszeit-Tabelle handelt, sieht TSQL2 sechs verschiedene Typen von Tabellen vor.
Ein wesentliches Charakteristikum des BCDM und TSQL2 ist das automatische Zusammenfassen mehrerer aufeinander folgender Versionen mit identischen Attributwerten, was sowohl in der Literatur als auch in der Praxis nicht unumstritten ist. Dieser Vorgang wird als coalescing bezeichnet. Beim coalescing ist die Gefahr eines Informationsverlustes gegeben, da bei der Zusammenfassung zweier aufeinander folgender Versionen zu einer Version, in die Historie eingegriffen wird. Auf der anderen Seite ist die Bearbeitung von Relationen, in denen Versionen existieren, die zusammengefaßt werden können, bei Datenbankabfragen wesentlich schwieriger und aufwendiger. Das coalescing sollte daher bestenfalls explizit unterstützt werden.
Auf Integritätsbedingungen, die gerade bei zeitbezogenen Daten ein Problem darstellen können, wird in TSQL2 überhaupt nicht eingegangen, so daß die Ausführung der Datenmodifikationsoperationen `INSERT`, `UPDATE` und `DELETE` in TSQL2 auch zu logisch widersinnigen bzw. falschen Ergebnissen führen könnten.
Eine gute, komprimierte und auch kritische Beschreibung von TSQL2 findet sich in (Myrach 97).

3.2 Das TRM und TSQL

Bereits etwas älter ist das TRM (Temporal Relational Model) und die darauf basierende Sprache TSQL, entwickelt von Navathe und Ahmed (Navathe and Ahmed 89), (Navathe and Ahmed 93). Das TRM ist eine Weiterentwicklung des relationalen Modells, unterstützt nur die Gültigkeitszeit, und bildet die Zeitdimension mittels

Tupelzeitstempelung ab. Insbesondere wird im TRM das Konzept der temporalen Abhängigkeit zwischen zwei zeitabhängigen Attributen vorgestellt. Eine temporale Abhängigkeit ist dann gegeben, wenn sich in einer Relation zwei oder mehrere zeitabhängige Attribute befinden, deren Werte sich nicht zum selben Zeitpunkt ändern. Beim Entwurf temporaler Datenbanken mit dem TRM muß darauf geachtet werden, daß temporale Abhängigkeiten vermieden werden, da es ansonsten zu Update- und Retrievalanomalien kommen kann. Ändern sich zeitabhängige Attribute immer zum selben Zeitpunkt, spricht man von synchronen Attributen. In einer zeitbezogenen Relation können im TRM nur synchrone Attribute enthalten sein. In der folgenden Relation existiert eine temporale Abhängigkeit zwischen den Attributen Funktion und Gehalt, da sich zum Zeitpunkt 1.September 1996 zwar der Gehalt des Mitarbeiters 1 ändert, nicht aber seine Funktion.

P#	Funktion	Gehalt	Beginn	Ende
1	Assistent	20.000	1995-SEP-01	1996-AUG-31
1	Assistent	22.000	1996-SEP-01	1998-DEC-31

Die Relation müßte auf zwei Relationen mit den Attributen P#, Funktion, Beginn, Ende bzw. P#, Gehalt, Beginn, Ende aufgeteilt werden. Die temporale Abhängigkeit zwischen den Attributen Funktion und Gehalt ist deshalb problematisch, da dadurch die Gültigkeitsspanne (life span) des Attributs Funktion auf zwei oder mehrere Tupel aufgeteilt wird und somit einfache Abfragen wie z.B. "Wie lange war ein Mitarbeiter Assistent?" nur mit relativ aufwendigen Abfragen beantwortet werden können. Auch bei der Erfassung neuer Versionen kann es bei Relationen mit temporal abhängigen Attributen zu Problemen kommen. Soll eine neue Version eingefügt werden, da sich z.B. der Gehalt von 22.000 auf 23.000 ändert, muß vor der INSERT Anweisung eine SELECT Anweisung durchgeführt werden, um den aktuellen Wert des Attributs Funktion zu erhalten.

Navathe und Ahmed definieren eine eigene time normal form (TNF), indem sie festlegen, daß eine zeitbezogene Relation dann in TNF ist, wenn sie in BCNF ist und keine temporalen Abhängigkeiten zwischen ihren Attributen existiert.
Die Sprache TSQL ist eine Erweiterung von Standard-SQL und erweitert SQL um einige zeitbezogene Konstrukte. Obwohl TSQL ein Gültigkeitsintervall mit zwei Zeitstempelattributen abbildet, können in TSQL mit dem Konstrukt Relationenname.INTERVAL das Gültigkeitsintervall direkt angesprochen und die Zeitoperatoren von Allen (Allen 83) darauf angewendet werden. Mit der BREAK Klausel in einer SELECT Anweisung können in TSQL auch Lücken in der Historie eines Objekts angesprochen werden. Eine Lücke ist dann gegeben, wenn die Gültigkeitsintervalle der Versionen eines Objekts nicht direkt aneinander anschließen, sondern dazwischen Zeitpunkte existieren, zu denen das Objekt keinen definierten Zustand aufweist. Auch das Gruppieren von zeitbezogenen Daten ist mittels der MOVING WINDOW Klausel möglich.

3.3 Das Temporal Relational Data Model und SQL^T

Die Sprache SQL^T (Tansel 93), (Tansel 97a), (Tansel 97b) basiert auf einem Datenmodell mit Attributzeitstempelung. Das heißt insbesondere, daß Relationen verwendet werden, die nicht in 1.Normalform sind (NFNF), da die Werte ihrer zeitabhängigen Attribute nicht atomar sind. Das Datenmodell ist ein Gül-

tigkeitszeitmodell. Das der Sprache SQL^T zugrundeliegende temporale relationale Datenmodell unterstützt "nested relations", das sind Relationen, deren Attribute wieder aus Relationen oder Mengen bestehen können. Das hier behandelte Datenmodell sieht allerdings nur eine Schachtelungstiefe von 1 vor, d.h. daß Attribute entweder atomar sein können, oder aber Mengen sein können, jedoch keine Mengen von Mengen. Jeder Attributwert zeitabhängiger Attribute wird als <*Zeit, Wert*>-Paar aufgezeichnet, wobei *Zeit* ein Zeitpunkt sein kann, zu dem der Attributwert belegt wurde, oder ein Intervall während dessen der Attributwert in der modellierten Realität gültig ist.
Die Sprache SQL^T wird nur in (Tansel 93) beschrieben. In seinen späteren Arbeiten zu diesem Thema ((Tansel 97a), (Tansel 97b)) geht der Autor auf diese Sprache nicht mehr ein, obwohl im Schlußteil von (Tansel 93) einige Erweiterungs- und Verbesserungsvorschläge für SQL^T gemacht werden. In (Tansel 93) wird lediglich der Abfrageaspekt der Sprache genau beschrieben. Durch das Voranstellen des Zeichens # bei einem Attribut in der SELECT Klausel einer SELECT Anweisung wird die gesamte Historie dieses Attributs ausgegeben. Wenn nicht die gesamte Historie eines zeitbezogenen Attributs ausgegeben werden soll, sondern nur ein bestimmter Zeitbereich, kann in der SELECT Klausel das Attribut mit einem Zeitzusatz versehen werden. Sei *A* ein zeitabhängiges Attribut und sei *t* eine Menge von Zeitpunkten, so kann mit der Anweisung SELECT #A[t] FROM Tabelle der Teil der Historie von *A* ausgegeben werden, der sich innerhalb des angegebenen Zeitraums *t* befindet. Die COLLAPSE Klausel dient dazu um mehre <*Zeit,Wert*>-Paare mit aneinander angrenzenden oder überlappenden Zeitbereichen zu einem <*Zeit,Wert*>-Paar zusammenzufassen und entspricht damit einem coalescing auf Attributebene.
In seinen späteren Arbeiten (Tansel 97a), (Tansel 97b) definiert Tansel einen temporalen Relationenkalkül und eine temporale relationale Algebra, ohne diese jedoch in eine konkrete temporale Datenbanksprache syntaktisch umzusetzen.

3.4 IXRM und IXSQL

Das Interval extended relational model (IXRM) und die darauf basierende Sprache IXSQL (Lorentzos and Mitsopoulos 97), (Lorentzos 93) wurde von den Griechen Lorentzos und Mitsopoulos entwickelt. IXRM ist ein Gültigkeitszeitmodell, unterscheidet sich jedoch von den meisten anderen Modellen und Sprachen, da im Mittelpunkt dieses Modells die explizite Unterstützung eines Datentyps Interval steht. Basierend auf diesem Intervalldatentyp wurden neue Prädikate und Funktionen entwickelt und auch die relationale Algebra intervallmäßig erweitert. IXSQL ist eine aufwärtskompatible Sprache zu SQL92, d.h. daß mit IXSQL auch nichtzeitbezogene Daten verarbeitet werden können und alle Elemente des SQL92-Standards unterstützt werden. Die wichtigsten Erweiterungen von IXSQL gegenüber SQL92 sind neben dem Intervall-Datentyp die Funktionen FOLD bzw. UNFOLD und NORMALIZE. Mit der Funktion UNFOLD ist es möglich, eine Relation von ihrer Intervalldarstellung auf die Darstellung einzelner Zeitpunkte "herunterzubrechen". Wendet man auf folgende Relation die Funktion UNFOLD an:

P#	Funktion	Zeit
1	Assistent	[d32,d35)
1	Dozent	[d35,d38)

dann erhält man als Ergebnis die nachstehende Relation:

P#	Funktion	Zeit
1	Assistent	d32
1	Assistent	d33
1	Assistent	d34
1	Dozent	d35
1	Dozent	d36
1	Dozent	d37

UNFOLD ermöglicht somit, ein "umgekehrtes coalescing" durchzuführen. FOLD ist die inverse Funktion zu UNFOLD. Die Funktion NORMALIZE führt zuerst ein FOLD und dann auf die so erzeugte Relation ein UNFOLD durch.
IXSQL unterstützt alle Datentypen *D* aus SQL92 und zusätzlich für jeden Datentyp *D* auch den korrespondierenden Intervalltyp *I(D).* Das bedeutet, daß Intervalle auch von Zahlen und Zeichendatentypen und nicht nur von zeitbezogenen Datentypen gebildet werden können. Intervalle sind in IXSQL immer halboffene Intervalle.
Bei den Integritätsbedingungen wird in den Arbeiten zu IXSQL nur auf die Entitätsintegrität Bezug genommen. Der PRIMARY KEY einer zeitbezogenen Relation wird gegenüber dem PRIMARY KEY der korrespondierenden konventionellen Relation um das Zeitattribut erweitert. Dadurch ist eine effektive Überprüfung der Entitätsintegrität gewährleistet, da mit dem INTERVAL Datentyp auch die Überlappung von Zeitintervallen überprüft werden kann.

3.5 ATSQL2

ATSQL2 (Steiner 98) ist eine der neuesten Entwicklungen auf dem Gebiet temporaler Datenbanksprachen. ATSQL2 basiert auf dem relationalen Datenmodell und hat sich als Hauptziel eine konsequente Aufwärtskompatibilität zu SQL92 gesetzt. Im Unterschied zu den meisten anderen Sprachentwürfen existiert für ATSQL2 mit TimeDB 2.0 (TimeDB 98) auch ein bitemporales relationales DBMS, das Anbindungen an kommerziell verfügbare Datenbankmanagementsysteme wie Oracle oder Sybase vorsieht. Die Grundidee liegt dabei darin, daß zwar mit ATSQL2 eine zu SQL92 aufwärtskompatible Sprache definiert wurde, daß jedoch jede ATSQL2-Anweisung in eine gültige SQL92-Anweisung umgewandelt wird. Damit dient ATSQL2 quasi als äußere Schicht, auf der die Benutzer zeitbezogene Daten effizient bearbeiten und abfragen können. Diese Abfragen werden dann in teilweise sehr komplexe SQL92-Befehle umgewandelt und an ein konventionelles DBMS weitergegeben, ohne daß der Anwender diesen "Umwandlungsschritt" bemerkt. Zwei wesentliche Anforderungen an die Aufwärtskompatibilität von ATSQL2 sind, daß jede korrekte SQL92-Anweisung auch in ATSQL2 gültig ist und daß die Syntax von ATSQL2 so ähnlich wie möglich der Syntax von SQL92 ist. Diesen Anforderungen entsprechend werden in ATSQL2 durch das Voranstellen der Schlüsselwörter VALID oder TRANSACTION oder VALID AND TRANSACTION vor einer SELECT Anweisung aus konventionellen Abfragen zeitbezogene Abfragen. Die Sprache ATSQL2 unterstützt Gültigkeitsintervalle, wobei diese Intervalle bei der Umwandlung in SQL92 in zwei versteckte Zeitstempelattribute Begin und End umgewandelt werden. TimeDB hat eine temporale relationale Algebra implementiert, was bedeutet, daß die Standardoperationen aus der relationalen Algebra

unter Berücksichtigung der Gültigkeitszeit erweitert und anschließend wieder in SQL92-Anweisungen umgewandelt werden. Auch eine bitemporale relationale Algebra ist vorgesehen, die nach demselben Schema funktioniert. Zusätzlich zu den beiden versteckten Gültigkeitszeitstempelattributen werden dann noch zwei versteckte Transaktionszeitstempelattribute in SQL92 angelegt. Ein wichtiges Konzept von ATSQL2 sind abgeleitete Tabellen. Dabei geht es darum, daß in der FROM Klausel einer SELECT Anweisung anstelle eines Tabellennamens wieder eine SELECT Anweisung stehen kann, deren Ergebnis dann wie eine Tabelle behandelt wird. Dieses Konzept ist auch im Full Level von SQL92 vorgesehen.
Auf Integritätsbedingungen wird in den Arbeiten zu ATSQL2 nur kurz eingegangen. Durch das Voranstellen des Schlüsselwortes VALID werden aus konventionellen Integritätsbedingungen temporale Integritätsbedingungen. Details dazu sind in den Arbeiten allerdings nicht vorhanden.

3.6 SQL/Temporal, der zeitbezogene Teil des SQL3 Standardentwurfs

Seit einigen Jahren wird an der Weiterentwicklung des derzeit gültigen SQL-Standards gearbeitet. Der Entwurf für diese Weiterentwicklung wird als SQL3 (ISO/IEC 97) bezeichnet und enthält mit SQL/Temporal auch einen Teil, der sich mit zeitbezogenen Daten beschäftigt. Das zugrundeliegende Datenmodell bleibt freilich das relationale Datenmodell, so daß bei SQL/Temporal in erster Linie Erweiterungen in Bezug auf die benutzerdefinierte Zeit enthalten sind. Wesentlichste Neuerung ist die Einführung eines neuen Datentyps PERIOD. PERIOD entspricht dem Datentyp INTERVAL OF DATE in IXSQL. SQL/Temporal besitzt damit mit DATE, TIME, TIMESTAMP, INTERVAL und PERIOD Datentypen für die Abbildung von Zeitpunkten, von Perioden und von Intervalldauern. Mit CONTAINS, PRECEEDES, SUCCEEDS, OVERLAPS und MEETS werden auch alle für den Datentyp PERIOD relevanten Operationen definiert. Darüber hinaus wird in SQL/Temporal auch eine NORMALIZE Funktion definiert. Die Aufgabe dieser Funktion ist dieselbe wie in IXSQL.

4. Bewertung und Vergleich der Modelle

In diesem Abschnitt der Arbeit werden die Modelle vorerst einzeln beurteilt und daran anschließend basierend auf sieben ausgewählten Kriterien miteinander verglichen.

4.1 Bewertung der vorgestellten Modelle

BCDM und TSQL2: Beim Sprachentwurf von TSQL2 werden eine große Anzahl impliziter Annahmen getroffen, deren Ausnutzung die Formulierung kompakter Anfragen erlaubt (Myrach 97). Problematisch erscheint an diesem Ansatz, daß der Aspekt des Datenbankentwurfs völlig vernachlässigt wird. Das geht soweit, daß in TSQL2 keine Konstrukte für Integritätsbedingungen, wie PRIMARY KEY oder FOREIGN KEY, explizit vorgesehen sind. Das automatische coalescing kann ebenso wie beim TRM und TSQL zu massiven Informationsverlusten führen.

TRM und TSQL: Das Konzept der temporalen Abhängigkeit hilft beim effizienten Entwurf einer temporalen Datenbank. Problematisch ist jedoch, daß mit diesem Modell zeitabhängige Attribute, deren Werte sich über mehrere Perioden eines Beobachtungszeitraums nicht ändern, nicht explizit abgebildet werden können.

Das führt zu einem Informationsverlust, nämlich einem Verlust der Information, daß sich der Wert eines Attributs eben nicht geändert hat. Auf die referentielle Integrität wird im TRM nicht eingegangen. Bei der Beschreibung von TSQL wird ein Hauptgewicht auf die erweiterte SELECT Anweisung gelegt. Die Aspekte der Datendefinitionssprache und der Datenmanipulationsoperationen werden hingegen völlig vernachlässigt.

SQL^T: Die Attributzeitstempelung hat gegenüber der Tupelzeitstempelung unter anderem den Vorteil, daß die gesamte Historie eines zeitabhängigen Attributs direkt beim Attribut abgespeichert ist. Deshalb kann bei Modellen mit Attributzeitstempelung das "Coalescing-Problem" (vgl. dazu Abschnitt 3.1 dieser Arbeit) nie auftreten. Auf das Design zeitbezogener Relationen mit SQL^T und die Datenmanipulation mit dieser Sprache wird in den Arbeiten von Tansel überhaupt nicht eingegangen. Daher ist es auch sehr schwierig, die Behauptung des Autors zu überprüfen, daß SQL^T zu Standard-SQL aufwärtskompatibel ist. Jedenfalls bleibt die Tatsache bestehen, daß das Datenmodell, auf dem SQL^T basiert, zum relationalen Datenmodell, auf dem die meisten weit verbreiteten DBMS basieren, nicht aufwärtskompatibel ist. Dieses Manko macht eine baldige Verfügbarkeit von Sprachen mit Attributzeitstempelung in kommerziell verfügbaren DBMS schwer möglich. Ein alternativer Ansatz ist m.E. darin zu sehen, Datenmodelle mit Attributzeitstempelung im allgemeinen und das Datenmodell von SQL^T im besonderen in objektrelationalen Datenmodellen abzubilden. Mit Oracle 8.0 unterstützt ein weit verbreitetes kommerzielles DBMS ein objektrelationales Datenmodell.

IXRM und IXSQL: Der wesentliche Vorteil von IXSQL liegt in der expliziten Unterstützung von Intervallen und der Definition von intervallspezifischen Funktionen und Prädikaten. Gültigkeitsintervalle können damit mit einem Attribut abgebildet werden und brauchen nicht mehr mit zwei Zeitstempelattributen (Beginn und Ende) quasi simuliert werden. Das hat unter anderem auch den Vorteil, daß das DBMS automatisch die ordnungsgemäße Verwendung von Intervallen überprüft. Beispielsweise ist es nicht möglich, ein Intervall anzugeben, bei dem die untere Grenze größer als die obere Grenze ist, was bei der Darstellung von Intervallen mit zwei Zeitstempelattributen erst durch entsprechende Integritätsbedingungen explizit ausgeschlossen werden muß. Weiters ist es in IXSQL auch möglich, mehrere Gültigkeitszeitintervalle anzulegen. Auf der Ebene der Integritätsbedingungen wird nur auf die Entitätsintegrität näher eingegangen. Mit der expliziten Erweiterung des PRIMARY KEY um das IntervallZeitattribut ist eine effektive Überwachung der Entitätsintegrität möglich, da auch die Überlappung von Zeitintervallen überprüft wird. Auf die referentielle Integrität wird in den Arbeiten von Lorentzos und Mitsopoulos leider nicht eingegangen. Positiv zu vermerken ist noch, daß in IXSQL ein coalescing, das dort als NORMALIZE bezeichnet wird, zwar möglich ist, vom Datenbankdesigner aber explizit angegeben werden muß. D.h., es gibt kein implizites Zusammenfassen von Datensätzen wie in TSQL2.

ATSQL2: Der Vorteil dieses Ansatzes liegt vor allem darin, daß diese Sprache mit kommerziell verfügbaren DBMS kombinierbar und vollständig aufwärtskompatibel zu SQL92 ist. Damit ist ein schrittweiser Übergang von einer konventionellen Datenbank auf eine temporale Datenbank möglich, ohne daß "Altsysteme" neu programmiert werden müssen. Da die Sprache kein eigenes Datenmodell hat, son-

dern auf dem relationalen Modell basiert, werden alle ATSQL2-Anweisungen in SQL92-Anweisungen umgewandelt. Die dadurch entstehenden SQL92-Anweisungen sind oftmals sehr komplex, was negative Auswirkungen auf die Performance haben kann. Angaben dazu liegen in den aktuellen Arbeiten jedoch noch nicht vor.

SQL Temporal: In SQL/Temporal sind Konzepte und Ideen von TSQL2, ATSQL2 und IXSQL eingeflossen. SQL/Temporal kann damit als ein guter Kompromiß der wichtigsten Sprachentwürfe verstanden werden, da sich in dieser Sprache eine große Anzahl von Forschern auf dem Gebiet temporaler Datenbanken wiederfindet. Darüber hinaus sollte SQL/Temporal als Teil des neuen SQL3-Standards nach dessen Verabschiedung auch für die Hersteller allgemein verbindlich werden. Ähnlich wie bei ATSQL2 kann auch SQL/Temporal als "zeitbezogene Schicht" eines DBMS verstanden werden, da das zugrundeliegende Datenmodell mit dem relationalen Modell nach wie vor kein temporales Datenmodell ist.

4.2 Ein vergleichender Überblick der vorgestellten Modelle

In der folgenden Übersicht sollen die vorgestellten Datenmodelle und Datenbanksprachen nochmals miteinander verglichen werden. Für diesen Vergleich werden sieben Vergleichskriterien (V1-V7) herangezogen. Die Beschreibung einiger Datenmodelle bzw. Datenbanksprachen in der Literatur ist nicht vollständig, so daß nicht alle Vergleichskriterien eindeutig bestimmt werden können.

V1: Welche Zeitdimension wird unterstützt?
V2: Wird die Entitätsintegrität angemessen unterstützt?
V3: Wird die referentielle Integrität angemessen unterstützt?
V4: Wird ein coalescing automatisch durchgeführt?
V5: Ist das Datenmodell zum relationalen Modell aufwärts kompatibel?
V6: Ist die Datenbanksprache zu SQL92 (syntaktisch) aufwärts kompatibel?
V7: Gibt es einen (kommerziell) verfügbaren Prototyp der Sprache?

	TSQL2	**TSQL**	**SQL^T**	**IXSQL**	**ATSQL2**	**SQL/Temporal**
V1	bi	Gültigk.	Gültigk.	Gültigk.	bi	Gültigk.
V2	nein	ja[1]	k.A.[2]	ja	ja?[3]	k.A.[2]
V3	nein	nein	k.A.[2]	nein	ja?[3]	k.A.[2]
V4	ja	ja	nein	nein	nein	nein
V5	nein	ja	nein	ja	ja	ja
V6	ja	ja	ja	ja	ja	ja
V7	nein	nein	nein	nein	ja	nein

[1] Allerdings nur im Datenmodell und nicht in der Datenbanksprache
[2] k.A. = in der Literatur wurden dazu keine entsprechenden Angaben gefunden.
[3] In den Arbeiten zu ATSQL2 wird zwar von entsprechenden Integritätsbedingungen gesprochen. Eine genauere Beschreibung derselben erfolgt aber nicht, so daß hier keine eindeutige Antwort gegeben werden kann.

Ein Mangel ist fast allen vorgestellten Modellen und auch allen anderen in der Literatur behandelten temporalen Datenmodellen gemeinsam: die Modelle und die darauf basierenden Datenbanksprachen sind in keinen kommerziell verfügbaren DBMS implementiert. Auch mittelfristig ist nicht zu erwarten, daß ein temporales Datenbankmanagementsystem verfügbar sein wird. Aus diesem Grund erscheint es sinnvoll, sich verstärkt mit der Frage auseinanderzusetzen, wie die Zeitdimension in bereits bestehende und bewährte Modelle integriert werden kann.
In (Kaiser 96) wurden dazu bereits einige Gedanken dargestellt. Im folgenden Abschnitt wird auf neue Erkenntnisse und Weiterentwicklungen mit diesem Ansatz näher eingegangen. Insbesondere wurde in den jüngsten Arbeiten an diesem Modell versucht, auch Aspekte von Modellen mit Attributzeitstempelung zu berücksichtigen, dabei aber trotzdem im Rahmen des konventionellen relationalen Modells zu bleiben, um eine Implementierung in weit verbreiteten Datenbankmanagementsystemen zu ermöglichen.

5. Die Abbildung zeitbezogener Daten im relationalen Modell

Die Analyse und Bewertung der vorgestellten Modelle hat gezeigt, daß die Modelle vor allem in der Phase des Entwurfs temporaler Datenbanken keine angemessene Unterstützung bieten. Kein einziges Modell behandelt alle Integritätsbedingungen in temporalen Datenbanken zufriedenstellend. Auch in der aktuellsten Literatur wird auf diesen so wichtigen Aspekt kaum eingegangen (eine Ausnahme dabei bildet (Bergamaschi and Sartori 98)). Im folgenden wollen wir uns daher auf den Aspekt des Entwurfs temporaler relationaler Datenbanken konzentrieren.
Ein wichtiger Schwerpunkt bei der Abbildung zeitbezogener Daten im relationalen Modell muß in der sauberen Definition adäquater Integritätsbedingungen liegen. Dazu werden mit dem PRIMARY KEY TEMPORAL (Kaiser 98b) und dem FOREIGN KEY TEMPORAL (Kaiser 98a) zwei spezifisch zeitbezogene Integritätsbedingungen eingeführt. Diese beiden Konstrukte stellen zwar Erweiterungen zu SQL92 dar, können jedoch auch mit dem im SQL92-Standard definierten Sprachumfang formuliert werden. Das Konzept der temporalen Abhängigkeit von Attributen wird aufgenommen und erweitert. Dabei wird jedoch zwischen verschiedenen Kategorien von zeitabhängigen Attributen unterschieden. Durch diese Unterscheidung kann wesentlich besser auf den Kontext der abgebildeten modellierten Realität Rücksicht genommen werden. Obwohl das Modell eine Tupelzeitstempelung verfolgt, können durch die Unterscheidung in unterschiedliche Attributkategorien auch einige Aspekte der Attributzeitstempelung einfließen. Im folgenden werden die Ecksteine dieses Modells kurz vorgestellt.

5.1 Temporale Integritätsbedingungen

Die PRIMARY KEY TEMPORAL Klausel ist eine Erweiterung von SQL92 und stellt die temporale Entitätsintegrität sicher. Die temporale Entitätsintegrität legt fest, daß zu jedem Zeitpunkt *t* jedes Objekt nur eine gültige Version besitzen darf. Wie aus der folgenden Definition ersichtlich ist, kann die PRIMARY KEY TEMPORAL Klausel mittels einer expliziten Integritätsbedingung auch in SQL92 aufgelöst werden.

Seien die c_i die Komponenten des PRIMARY KEY TEMPORAL.
Die PRIMARY KEY TEMPORAL-Bedingung ist genau dann verletzt, wenn folgende Suchbedingung wahr ist:

```
EXISTS (SELECT * FROM B a, B b
      WHERE NOT (UNIQUE (SELECT a.c1,...,a.cn FROM a)
          AND (a.c1,...,a.cn) IS NOT NULL
          OR  ((a.c1,...,a.cn)=(b.c1,...,b.cn) AND
              (a.TB<>b.TB) AND
              (A.TB=b.TE OR a.TE=b.TB OR
              (a.TB,a.TE) OVERLAPS (b.TB,b.TE)))
```

Die FOREIGN KEY TEMPORAl Klausel ist eine Erweiterung zu SQL92 und stellt die temporale referentielle Integrität sicher. Die temporale referentielle Integrität legt fest, daß eine Version eines Objekts sich nur dann auf eine referenzierte Tabelle beziehen kann, wenn die Gültigkeitperiode des Tupels in der referenzierten Tabelle die Gültigkeitsperiode des Tupels in der referenzierenden Tabelle beinhaltet. Auch die FOREIGN KEY TEMPORAL Klausel kann mittels einer expliziten Integritätsbedingung in SQL92 aufgelöst werden.

Seien *R2* eine Tabelle und *FK* der Fremdschlüssel der Tabelle und T_B, T_E die Zeitattribute dieser Tabelle. *R2* ist die referenzierende Tabelle. Seien *R1* eine Tabelle und *PK*,T_B die Attribute des PRIMARY KEY TEMPORAL dieser Tabelle und T_B, T_E die Zeitattribute von *R1*. *R1* ist die referenzierte Tabelle.
Die FOREIGN KEY TEMPORAL-Bedingung ist erfüllt, wenn die folgende Suchbedingung wahr ist:

```
EXISTS (SELECT * FROM R2,R1
            WHERE FK=PK
            AND R2.TB >= R1.TB
            AND R2.TE <= R1.TE);
```

Mit diesen beiden Konstrukten ist eine effiziente und saubere Überwachung der Entitätsintegrität und der referentiellen Integrität von Tabellen mit zeitbezogenen Daten gewährleistet.

5.2. Attribute mit Zeitbezug

Bereits die Beschreibung und Bewertung der Datenmodelle in Abschnitt 3 und 4 dieser Arbeit hat gezeigt, daß die Attributzeitstempelung einige Vorteile gegenüber der Tupelzeitstempelung bietet. Insbesondere ist dabei die direkte Verbindung der Historie mit einem Attribut hervorzuheben. Ziel unserer laufenden Forschungsarbeiten ist es daher, basierend auf dem traditionellen relationalen Modell die Attribute einer zeitbezogenen Relation mit zusätzlichen semantischen Informationen zu versehen. Unter einer zeitbezogenen Relation wird dabei eine konventionelle Relation verstanden, die um zwei Zeitstempelattribute (Beginn und Ende) erweitert wird.

In zeitbezogenen Relationen kann zwischen fünf unterschiedlichen Attributarten unterschieden werden:

Zeitstempelattribute
zeitunabhängige Attribute
zeitabhängige Attribute im weiteren Sinn
zeitabhängige Attribute im engeren Sinn
zeitabhängige zyklische Attribute

Diese fünf unterschiedlichen Attributarten übernehmen alle Eigenschaften eines Attributs aus dem relationalen Modell, erhalten aber darüber hinaus zusätzliche semantische Informationen. In weiterer Folge werden die je eigenen Charakteristika dieser Attributarten beschrieben.

Klasse 0: Zeitstempelattribute: Die beiden Zeitstempelattribute T_B und T_E sind auf der Domäne *'Zeit'* definiert und legen den Beginn und das Ende einer Gültigkeitsperiode fest.
Klasse 1: zeitunabhängige Attribute: Die Werte von zeitunabhängigen Attributen bleiben im Zeitablauf einer zeitbezogenen Relation konstant. Ein typisches Beispiel für ein zeitunabhängiges Attribut wäre etwa das Geburtsdatum eines Mitarbeiters.
Klasse 2: zeitabhängige Attribute im weiteren Sinn: Die Werte von zeitabhängigen Attributen im weiteren Sinn können sich im Zeitablauf einer zeitbezogenen Relation ändern. Die Wertänderungen dieser Attribute sind jedoch im Kontext der modellierten Realität nicht relevant. Das bedeutet, daß die Historie solcher Attribute nicht nachvollziehbar sein muß und daß immer nur der momentan gültige und aktuelle Wert dieses Attributs relevant ist. Ein Beispiel für ein zeitabhängiges Attribut im weiteren Sinn wäre etwa der Name eines Mitarbeiters, natürlich aber nur dann, wenn frühere Namen eines Mitarbeiters in der modellierten Realität nicht relevant sind.
Klasse 3: zeitabhängige Attribute im engeren Sinn: Die Werte von zeitabhängigen Attributen im engeren Sinn können sich im Zeitablauf einer zeitbezogenen Relation ändern. Die Wertänderungen dieser Attribute sind relevant, d.h. die Historie dieser Attribute muß nachvollziehbar sein. Wird für ein Objekt eine neue Version angelegt, so muß sich der Wert eines zeitabhängigen Attributs im engeren Sinn gegenüber dem Wert in der Vorversion jedenfalls ändern. Ein Beispiel für ein zeitabhängiges Attribut im engeren Sinn wäre etwa die Funktion eines Mitarbeiters.
Klasse 4: zeitabhängige zyklische Attribute: Die Werte von zeitabhängigen zyklischen Attributen können sich im Zeitablauf einer zeitbezogenen Relation ändern. Die Wertänderungen dieser Attribute sind relevant, d.h. die Historie dieser Attribute muß nachvollziehbar sein. Wird für ein Objekt eine neue Version angelegt, so muß sich der Wert eines zeitabhängigen zyklischen Attributs gegenüber dem Wert in der Vorversion nicht unbedingt ändern. Kommt es zu keiner Wertänderung, so wird jedoch die Information, daß es keine Änderung gegeben hat (obwohl prinzipiell eine Wertänderung möglich gewesen wäre), ebenfalls abgebildet. Ein Beispiel für ein zeitabhängiges zyklisches Attribut wäre etwa der Gehalt eines Mitarbeiters. Bekommt ein Mitarbeiter keine Gehaltserhöhung, obwohl prinzipiell eine Gehaltserhöhung möglich gewesen wäre, so bleibt der Gehalt konstant, die Information über die nicht erfolgte Gehaltserhöhung wird aber ebenfalls abgebildet.

Zeitabhängige Attribute im weiteren Sinn, zeitabhängige Attribute im engeren Sinn und zeitabhängige zyklische Attribute können unter dem Überbegriff zeitabhängige Attribute zusammengefaßt werden.

Die Festlegung, welche Attribute zu welcher Kategorie gehören, liegt in den Händen des "Datenbankdesigners". Das bedeutet, daß schon in der Phase des konzeptionellen Datenbankentwurfs eine Entscheidung darüber getroffen werden muß, welche Attribute welche Bedeutung haben.

Eine Datenbank mit zeitbezogenen Daten muß nun so entworfen werden, daß in einer zeitbezogenen Relation neben den Zeitstempelattributen nur Attribute enthalten sind, die derselben Attributsklasse (Klasse 1 bis Klasse 4) angehören und nicht voneinander temporal abhängig sind. Die Definition der temporalen Abhängigkeit von Attributen wird dabei vom TRM ((Navathe and Ahmed 89) und (Navathe and Ahmed 93)) übernommen.

Im Vergleich mit den im Abschnitt 3 dieser Arbeit beschriebenen Modellen hat dieser Ansatz den Vorteil, daß auf den Entwurf von Datenbanken mit zeitbezogenen Daten und damit auf die Integritätsbedingungen explizit eingegangen wird. Das Coalescing-Problem wird mit der Einführung der unterschiedlichen Attributarten wesentlich entschärft. Durch die Abbildung im relationalen Modell können so zeitbezogene Daten in verfügbaren Datenbankmanagementsystemen abgebildet werden.

6. Zusammenfassung und Schlußfolgerungen

Im vorliegenden Beitrag wurde versucht, die Entwicklungen der letzten Jahre auf dem Gebiet der temporalen Datenbanken zu skizzieren. Es hat sich dabei gezeigt, daß es trotz einer großen Anzahl von temporalen Datenmodellen und Entwürfen für temporale Datenbanksprachen noch immer kein verfügbares temporales Datenbankmanagementsystem gibt. Ein solches ist auch mittelfristig nicht zu erwarten. Ansätze, die versuchen, die Zeitdimension in dem in vielen DBMS verfügbaren relationalen Datenmodell sauber zu integrieren, erhalten daher eine immer größere Bedeutung. Die Eckpunkte eines solchen Ansatzes wurden in dieser Arbeit vorgestellt, wobei insbesondere auf den sonst in der Literatur weitgehend vernachlässigten Aspekt des Datenbankentwurfs Rücksicht genommen wurde. Die weitere Umsetzung, Erweiterung und Implementierung dieses Ansatzes wird Inhalt zukünftiger Forschungsarbeiten und Projekte sein. Dabei wird in einiger Zeit auch auf den zeitbezogenen Teil des SQL3-Standards zurückgegriffen werden können.

Literatur

[Allen 83]
J.F.Allen. *Maintaining knowledge about temporal intervals*. Communications of the ACM, 1983, S. 832-843
[Bergamaschi and Sartori 98]
S. Bergamaschi and C.Sartori. *Chrono: a conceptual design framework for temporal entities.* Proceedings of the 17th International Conference on Conceptual Modeling (ER'98), 1998

[Böhlen and Jensen 96]
M. Böhlen and C.S. Jensen. *Seamless integration of time into SQL*. Technical report, R962049, Aalborg University, Department of Computer Science,1996.
[ISO/IEC 97]
ANSI TC X3H2, ISO/IEC JTC 1/SC 21/WG 3. *ISO Working Draft Temporal (SQL/Temporal)*, October 1997

[Kaiser 96]
A. Kaiser. *Zeitbezogene Datenbanksysteme: eine Bestandsaufnahme und ausgewählte Problemstellungen*. Proceedings 5. Internationales Symposium für Informationswissenschaft ISI'96, 1996, S. 143-155
[Kaiser 98a]
A. Kaiser. *Referential integrity constraints in temporal relational databases*. Working paper, submitted for publication, 1998.
[Kaiser 98b]
A. Kaiser. *Überlegungen zur Entitätsintegrität in temporalen relationalen Datenbanksystemen*. Proceedings des 4.ZOBIS-Workshops Hamburg. 1998.
[Lorentzos 93]
N.A.Lorentzos. *The Interval Extended Relational Model and its Application to Valid Time Databases.* J.Clifford and R.Snodgrass(eds.), Temporal Databases: Theory, Design and Implementation, 1993, S. 67-91
[Lorentzos and Mitsopoulos 97]
N.A. Lorentzos and Y.G. Mitsopoulos. *SQL Extension for Interval Data*. IEEE Transactions on Knowledge and Data Engineering, 1997, S.480-499
[Myrach 97]
T. Myrach. *TSQL2: Der Konsens über eine temporale Datenbanksprache.* Informatik-Spektrum, 1997, S.143-150
[Navathe and Ahmed 89]
S. Navathe and R. Ahmed. *A Temporal Relational Model and a Query Language.* Information Sciences, 1989, S.147-175
[Navathe and Ahmed 93]
S. Navathe and R. Ahmed. *Temporal extensions to the relational model and SQL.* In Tansel,A.: Temporal databases, Benjamin Cummings, 1993, S.92-109
[Snodgrass et al. 95]
R. Snodgrass et al. *The TSQL2 Temporal Language*. Kluwer Academic Publishers, 1995.
[Steiner 98]
A. Steiner. *A Generalisation Approach to Temporal Data Models and their Implementations.* Dissertation ETH Zürich, 1998
[Tansel 93]
A. Tansel. *SQL^T: A Temporal Extension to SQL.* In Snodgrass,R.(Ed.), Proceedings of the International Workshop of an Infrastructure for Temporal Databases, 1993, S. II1-II14
[Tansel 97a]
A. Tansel. *The Expressive Power of Temporal Relational Query Languages.* IEEE Transactions on Knowledge and Data Engineering, 1997, S.120-134
[Tansel 97b]

A. Tansel. *Temporal relational data model*. IEEE Transactions on Knowledge and Data Engineering, 1997, S. 464-479
[TimeDB 98]
A TimeConsult Product. *TimeDB 2.0 Demo Version*, http://www.TimeConsult.com, 1998