Temporale Algebren - ein vergleichender Überblick

Studienarbeit
Universität Rostock, Fachbereich Informatik

vorgelegt von: Thomas Courvoisier
geboren am 08.04.1972 in Stralsund

Betreuer: Dipl.-Math. Bernd Griefahn

Abgabedatum: 06.05.1997
letzte Änderung: 06.05.1997

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation

Ende der Siebziger Jahre wurde von verschiedenen Wissenschaftlern damit begonnen, nach Datenbanksystemen zu suchen, die zeitvariante Daten und Objekte in vernünftiger Weise handhaben können. Dafür gab und gibt es gute Gründe.
Die Zeit ist ein wichtiger Aspekt fast aller Phänomene der realen Welt. Ereignisse finden zu einem bestimmten Zeitpunkt statt, und auch Objekte und die Beziehungen zwischen ihnen bestehen nur über einen gewissen Zeitraum. So gibt es auch in der Praxis viele Anwendungen, in denen die Notwendigkeit besteht, den Zeitfaktor zu berücksichtigen. Anwendungen, in denen nicht nur der einfache Wert eines Objektes interessiert, sondern mindestens in gleichem Maße seine Änderung über einen gewissen Zeitraum. Man bezeichnet das als die *Geschichte* des Objektes. Solche Anwendungen können zum Beispiel

- Flugzeugreservierungen,
- Aufzeichnung wissenschaftlicher Testreihen,
- eine Datenbank der Börsenkurse,
- oder andere Anwendungen aus Wirtschaft und Bankwesen

sein. Die traditionellen[1] Datenbanksysteme konnten diesen Anforderungen nicht gerecht werden. Sie lieferten (und liefern noch) immer nur ein Abbild der realen Welt zu einem bestimmten Zeitpunkt (gewöhnlich „jetzt“). Man nennt dieses Abbild auch einen *Schnappschuß* der realen Welt. Es mußten also neue Datenbankmodelle entwickelt werden, die den Ansprüchen von zeitvarianten Daten genügten.
Die ersten Arbeiten auf diesem Gebiet konzentrierten sich auf die Erweiterung des Relationenmodelles von Codd [5]. Das Relationenmodell war Ende der Siebziger Jahre der letzte Stand der Forschung. Es zeichnet sich durch Klarheit und Einfachheit bei gleichzeitiger hoher Modellierungskraft aus. Deshalb sind noch heute die relationalen Datenbanksysteme am weitesten verbreitet.
Bis heute widmen sich die meisten Arbeiten der temporalen Erweiterung des Relationenmodelles. Aber auch auf dem Gebiet der objektorientierten Datenbanken wurden in den letzten Jahren temporale Modelle entwickelt. Zum Beispiel von Rose und Segev [10], die das objektbasierte Entity-Relationship-Modell in ein temporales objektorientiertes Modell erweiterten.
Zu den meisten der neuen Datenmodelle wurde auch eine Algebra angegeben, die temporale Anfragen an die Datenbank unterstützt. Diesen Algebren soll das Hauptaugenmerk dieser Arbeit gelten.

1.2 Was ist eine Algebra?

Die Algebra ist ein kritischer Teil eines Datenbankmanagementsystemes. Eine Algebra ist eine Menge von Objekten und eine Anzahl von Operationen über diesen Objekten. Das Datenbankmanagementsystem übersetzt die Anfragen in einen algebraischen Ausdruck, der dann optimiert und ausgeführt wird. Implementationsfragen, wie Optimierung und Speicherstrategien, können am besten auf Grund der Algebra entschieden werden.

[1] *„Traditionell“* bedeutet in diesem Sinne ebenso heute und jetzt, da es noch kein einziges kommerzielles temporales Datenbanksystem gibt.

Der Erfolg des relationalen Modells erklärt sich nicht zuletzt auch durch seine Algebra [11].
Mathematisch gesehen ist eine Algebra eine Menge von Trägermengen und dazugehörige Operationen. Um das genauer zu formulieren, definiert man zunächst den Begriff der Signatur:

Eine Signatur Σ *ist eine Familie von Mengen* $\Sigma_{w,s}$ *mit* $s \in S^*$ *(Hülle von* S*) und* $s \in S$*, wobei* S *eine Menge von Sorten ist.* $\Sigma_{w,s}$ *ist eine Menge von Operationssymbolen der Funktionalität* $s_1 \times s_2 \times \ldots \times s_n \rightarrow s$ *mit* $w = s_1, s_2, \ldots, s_n$.

Eine Sorte kann dabei zum Beispiel die Menge der Integer Zahlen sein. Aufbauend auf dieser Definition kann der Begriff der Σ-Algebra definiert werden:

Eine Σ*-Algebra* A *ist ein Paar* (A, OP) *von Mengenfamilien* $A = \{A_s\}_{s \in S}$ *und* $OP = \bigcup OP_{w,s}$*, wobei* $w \in S^*$ *und* $s \in S$*.* $OP_{w,s}$ *enthält zu jedem Operationssymol* $\sigma \in \Sigma_{w,s}$ *genau eine Operation* $\sigma_A : A_{s_1} \times A_{s_2} \times \ldots \times A_{s_n} \rightarrow A_s$*.* A_s *heißt Trägermenge der Sorte* S*.*

Die Menge der ganzen Zahlen ist zum Beispiel die Trägermenge der Sorte Integer.
Diese Definition einer Algebra beschreibt allerdings noch nicht die Algebren, die in Datenbanksystemen verwendet werden. In einem Datenbanksystem gibt es mehrere Sorten und damit auch Trägermengen, auf denen man nicht nur einzelne Operationen ausführen will, sondern ganze Operationsterme anwenden will. Eine Algebra, wie die bekannte Relationenalgebra des Relationenmedells, ist eine *Termalgebra*. Mathematisch ist eine Termalgebra wie folgt definiert:

Eine Termalgebra T_Σ *ist eine Paar, bestehend aus der Familie der Trägermengen* $\{T_{\Sigma,s}\}_{s \in S}$ *und der Familie der Mengen* OP_Σ *der Operationen zu den Operationssymbolen aus* Σ*.* $\{T_{\Sigma,s}\}_{s \in S}$ *ist die kleinste Familie von Mengen von Zeichenketten, bestehend aus den Operationssymbolen von* Σ*: „(“, „)“ und „,“. Dabei gilt:*

1. $\Sigma_s \subseteq T_{\Sigma,s}$

(Alle Konstanten der Sorte S sind in $T_{\Sigma,s}$ enthalten)

2. *Es sei* $\sigma \in \Sigma_{w,s}$, $w = s_1, s_2, \ldots, s_n \in S^*$, $s \in S$, $t_i \in T_{\Sigma,s} \Longrightarrow$ $\sigma(t_1, t_2, \ldots, t_n) \in T_{\Sigma,s}$
   Die Elemente von $T_{\Sigma,s}$ *heißen Terme.*

Die den Operationssymbolen zugeordneten Operationen (zur Termkonstruktion) sind folgendermaßen definiert:

- $\sigma \in \Sigma_s \Rightarrow \sigma_{T_\Sigma} = \sigma \in T_{\Sigma,s}$
- $\sigma \in \Sigma_{w,s} \Rightarrow \sigma_{T_\Sigma}(t_1, t_2, \ldots, t_n) \in T_{\Sigma,s}$, *mit* $t_i \in T_{\Sigma,s_i}$

Auf diese theoretische Definition einer Algebra wird im weiteren Verlauf der Arbeit kein Bezug mehr genommen. Es sei nur erwähnt, daß die Trägermengen der Algebra durch das jeweils verwendete Datenmodell bestimmt werden - weshalb die Beschreibung der temporalen Datenmodelle auch einen großen Platz in dieser Arbeit einnimmt - und die entsprechenden Operationen darauf definiert werden.
Interessant für die Betrachtungen dieser Arbeit sind noch zwei Eigenschaften von Algebren. Die *Abgeschlossenheit* einer Algebra ist dadurch gegeben, daß jede Operation wieder ein Gültiges Objekt produziert (das einer Trägermenge angehört). Mit anderen Worten: auf das Ergebnis einer Operation müssen wieder Operationen der selben Algebra angewendet werden können. Diese Eigenschaft ist für die Handhabung einer Algebra in einem Datenbanksystem sehr wichtig. Einmal ist das Produzieren von ungültigen Objekten eine potentielle Fehlerquelle, und zum anderen würde die Arbeit durch ungültige Objekte nur erschwert werden.
Eine andere Eigenschaft ist die *Vollständigkeit* einer Algebra. Bei relationalen Modellen gilt eine Algebra als vollständig, wenn sie mindestens so ausdrucksstark wie die Relationenalgebra von Codd ist. Für temporale Modelle reicht diese Definition allerdings nicht aus, obwohl Snodgrass in [38] diese Begründung liefert, um die Vollständigkeit seiner temporalen Algebra zu beweisen. Obwohl für eine temporale relationale Algebra

aus praktischen Gründen vorgeschlagen wird, daß sie sich auf die Relationenalgebra reduziert, wenn der Zeitaspekt außer Acht gelassen wird, muß sie logischer Weise auch einiges mehr leisten. Clifford, Crooker und Tansel schlagen in [24] eine Bewertung der Vollständigkeit von Algebren vor, die auf dem Vergleich der Ausdruckskraft der Algebra mit der Ausdruckskraft einer abstrakten Sprache. Die Sprache TQuel [38] ist ebenso mächtig, wie die von Clifford vorgeschlagene. Somit beweist man heute die Vollständigkeit von temporalen Algebren dadurch, daß man zeigt, daß jedes TQuel-Statement auch mit der Algebra ausgedrückt werden kann [7].
Andere für temporale Algebren wichtige Eigenschaften sind in Abschnitt 2.2.6 beschrieben.

Kapitel 2

Erweiterungen der Relationenalgebra

2.1 Datenmodelle

Um eine Algebra betrachten zu können, muß immer auch das zugrundeliegende Datenmodell einbezogen werden. Deshalb sollen in diesem Abschnitt die grundlegenden Konzepte von Modellen temporaler Datenbanken betrachtet werden.
Bei dieser Betrachtung interessieren intuitiv vor allem zwei Dinge:

- Wie stellt man die Zeit selbst in einer temporalen Datenbank dar?
- Wie integriere ich den Zeitaspekt der Objekte der realen Welt in die Datenbank?

Die erste Frage ergibt sich dabei aus der zweiten. Wenn man den Zeitaspekt eines Objektes in einer Datenbank festhalten will, muß man wissen, was genau *Zeit* in diesem Sinne eigentlich ist und wie sie - mathematisch gesehen - aussieht. Heute unterscheidet man drei orthogonale Arten von Zeit.
Die sogenannte *Gültigkeitszeit* (valid time) bezeichnet die tatsächliche Zeit, zu der zum Beispiel ein Ereignis in der realen Welt passierte. Unabhängig davon, wann es in die Datenbank aufgenommen wird. Oder sie

bezeichnet die Gültigkeit eines Wertes in der Realität. Die Gültigkeitszeit ist also identisch mit der Zeit in der realen Welt. Viele Forscher auf dem Gebiet der temporalen Datenbanken sind der Meinung, daß zur Lösung des in der Einführung beschriebenen Problems, den Zeitfaktor realer Objekte in einer Datenbank zu modellieren, die Betrachtung ausschließlich der Gültigkeitszeit, völlig ausreicht. Theoretisch muß man dem sicherlich zustimmen. Eine Datenbank, die nur Gültigkeitszeit unterstützt, wird *historisch* genannt [40].
Trotzdem führte Ben Zvi [4] den Begriff der *Transaktionszeit* (transaction time) ein. Die Transaktionszeit bezieht sich auf die Speicherung von Informationen in der Datenbank. Wenn zum Beispiel ein Ereignis in eine Datenbank aufgenommen wurde, so bezeichnet die Transaktionszeit, wann das geschah und nicht, wann das Ereignis wirklich stattfand. Oder sie gibt die Zeit an, zu der ein nicht mehr gültiger Wert aus der Datenbank entfernt wurde. Die Speicherung der Transaktionszeit ermöglicht es, den Zustand der Datenbank selbst (nicht der realen Welt) zu einem bestimmten Zeitpunkt wieder herzustellen. Deshalb nennt man Datenbanken, die diese Art von Zeit unterstützen auch *Rollback*-Datenbanken [40].
Snodgrass führte zusätzlich den Terminus der *benutzerdefinierten Zeit* (user defined time) ein [40]. Damit werden alle Darstellungen von Zeit bezeichnet, deren Semantik nur durch ein Anwendungsprogramm festgelegt wird. Die Datenbank selbst unterstützt nur verschiedene Operationen, wie zum Beispiel den Vergleich von solchen Zeiten. Mit anderen Worten ist die benutzerdefinierte Zeit nur eine zusätzliche Domäne, ähnlich Integer oder String. Diese Art von Zeit wird schon durch daß Relationenmodell von Codd [5] unterstützt.
Zusätzlich ist noch anzumerken, das man eine Datenbank *temporal* nennt, wenn sie sowohl Gültigkeitszeit als auch Transaktionszeit unterstützt [40]. Wenn eine Datenbank diese beiden Zeitarten unterstützt, unterscheidet man zwischen *proaktiven* und *retroaktiven* Datenänderungen. Im ersteren Fall werden Daten in der Datenbank geändert, noch bevor das in der Realität geschieht, das heißt, daß die Transaktionszeit kleiner als die Gültigkeitszeit ist. Im letzteren Fall ist dies genau umgekehrt.

Nach den semantischen Aspekten der Zeitdarstellung sollen nun auch die konzeptionellen Modelle betrachtet werden. Im Groben gibt es zwei Zeitmodelle. Das kontinuierliche Modell, in dem Zeit als isomorph zu den reellen Zahlen betrachtet wird, und das diskrete Modell, in dem Zeit als isomorph zu den natürlichen beziehungsweise zu einer diskreten Untermenge der reellen Zahlen gesehen wird. Im kontinuierlichen Modell entspricht jede reelle Zahl einem Zeitpunkt. Im diskreten Modell hingegen entspricht eine Zahl einer atomaren Zeiteinheit von bestimmter Dauer. Aber beide Modelle haben gemeinsam, daß die Zeit linear geordnet ist. Also gilt für zwei Zeiten t_1 und t_2, daß entweder t_1 vor t_2 oder t_2 vor t_1 ist, wenn beide Zeiten nicht gleich sind.
Eine atomare Zeiteinheit im diskreten Modell soll hier *Chronon* genannt werden [22]. Obwohl die Dauer zweier verschiedener Chronons nicht notwendigerweise gleich sein muß, wird sie für gewöhnlich durch die Granularität der Zeitmessung festgelegt.
Die Zeit selbst ist ganz allgemein gesehen eine kontinuierliche Größe. Trotzdem gibt es kaum Modelle temporale Datenbanken, die zur Zeitdarstellung das kontinuierliche Modell benutzen. Dafür gibt es verschiedene praktische Gründe. Zunächst ist die Zeitmessung ansich ungenau. Zeitmeßgeräte geben das Auftreten eines Ereignisses im Sinne von Chronons an. Außerdem beziehen wir uns selbst, in unserer natürlichen Sprache fast immer auf Chronons. Wenn zum Beispiel jemand sagt, daß er 1972 geboren sei, implizieren wir damit, daß er irgendwann im Zeitraum zwischen dem 1.1.1972 und dem 31.12.1972 geboren ist. Aus diesen beiden Gründen ergibt sich, das jede Implementierung eines temporalen Datenmodelles so oder so irgendeine diskrete Codierung der Zeit beinhalten muß.
Wie integriert man nun die Zeit in ein temporales Datenbankmodell? Die Gültigkeits- beziehungsweise Transaktionszeit wird den Objekten einer Datenbank in Form von *Zeitstempeln* (time stamps) zugeordnet. Diese Zeitstempel können je nach Datenmodell aus Intervallen oder Zeitpunkten beziehungsweise einzelnen Chronons bestehen und sowohl Ereignisse als auch Gültigkeitszeiten bezeichnen.

Zunächst soll nur die Erweiterung des Relationenmodells betrachtet werden. Nach Snodgrass und McKenzie [27] hat man beim Design eines Datenmodells für temporale relationale Datenbanken vier grundlegende Entscheidungen zu treffen, die im Folgenden näher erläutert werden sollen.

1. Wird die Gültigkeitszeit Tupeln oder Attributwerten zugeordnet?
Die temporalen Erweiterungen des Relationenmodelles kann man in zwei Hauptrichtungen unterteilen, und zwar danach, ob die Gültigkeitszeit den Tupeln einer Relation oder einzelnen Attributwerten zugeordnet wird. Beide Darstellungen haben Vor- und Nachteile.
Der Ansatz, Attributwerte mit Zeitstempeln zu versehen, kommt der Wirklichkeit näher als das sogenannte Tupeltimestamping. Schließlich sind es einzelne Attributwerte, die zeitabhängig sind, und nicht das ganze Tupel. Ein Beispiel soll das veranschaulichen. Abbildungen 2.1 und 2.2 zeigen eine Relation, die das Gehalt und den Manager von Angestellten zu verschiedenen Zeiten beinhaltet. In Abbildung 2.1 zeigt eine Variante, in der die Gültigkeitszeit den Attributwerten in Form von Intervallen zugeordnet ist. Es ist zu sehen, daß die Relation bei dieser Art der Darstellung nicht in der ersten Normalform vorliegt. Das hat bei der Implementierung von solchen Datenmodellen einen erhöhten Aufwand zur Folge. Auch eine Algebra für ein solches Modell ist komplexer als bei Modellen, die in erster Normalform bleiben. Mit diesem Preis bezahlt man die höhere Modellierungskraft eines solchen Modells.

Abbildung 2.1: Relation EMP mit Tupeltimestamping

ENAME	GEHALT	MANAGER
TOM	$\{<[15,25),14K>,$ $<[25,now],18K>\}$	$\{<[15,19),RON>,$ $<[19,22),GARY>,$ $<[22,now],AL>\}$
ANN	$\{<[10,now),12K\}$	$\{<[10,now),LIZ>\}$

Die Abbildung 2.2 zeigt die gleiche Relation in einer Darstellung, in der

Abbildung 2.2: Die Relation EMP mit Tupeltimestamping

ENAME	GEHALT	MANAGER	Valid Time
TOM	$14K$	RON	$[15, 19)$
TOM	$14K$	$GARY$	$[19, 22)$
TOM	$14K$	AL	$[22, 25)$
TOM	$18K$	AL	$[25, now)$
ANN	$12K$	LIZ	$[10, now)$

die Gültigkeitszeit für jedes Tupel in Intervalldarstellung angegeben ist. Diese Relation ist in erster Normalform, verstößt aber in dieser Form gegen die dritte Normalform. Die dadurch entstehende Redundanz kann reduziert werden, wenn die Relation in zwei neue Relationen aufgespalten wird. Der Nachteil eines Modells mit Tupeltimestamping ist also die Fragmentierung der Daten. Aber die Idee des Tupeltimestampings kommt dem Konzept des ursprünglichen Relationenmodelles am nächsten. Die Implementierung eines solchen Datenmodelles ist um ein Vielfaches einfacher als bei einem Modell mit Attributtimestamping. Snodgrass und McKenzie fordern, das Tupel und nicht Attribute mit Zeitstempeln versehen werden [27].

Lediglich ein Modell verwendet Timestamping sowohl auf der Attribut- als auch auf der Tupelebene - das HRDM von Clifford und Crooker [22], das noch ausführlicher vorgestellt wird.

2. Wie wird die Gültigkeitszeit repräsentiert?

Zeitstempel, die die Gültigkeitszeit repräsentieren können, entweder Chronons, Intervalle oder Mengen von Intervallen sein. In den meisten neueren Modellen hat sich die Darstellung der Gültigkeitszeit als Intervall durchgesetzt. Abbildung 2.3 zeigt einige Möglichkeiten der Darstellung. Das letzte Beispiel in der Abbildung definiert Intervalle implizit und fordert, daß der Wert eines Attributs über die gesamte Lebensdauer der Datenbank definiert ist. Eine solche Darstellung wird zum Beispiel von

Abbildung 2.3: Beispiele für die Darstellung der Gültigkeitszeit

ENAME	GEHALT	MANAGER	Valid Time
TOM	$14K$	*RON*	$[15, 19)$
TOM	$14K$	*GARY*	$[19, 22)$

ENAME	GEHALT	MANAGER	START	STOP
TOM	$14K$	*RON*	15	19
TOM	$14K$	*GARY*	19	22

ENAME	GEHALT	MANAGER	GÜLTIG SEIT
TOM	$14K$	*RON*	15
TOM	$14K$	*GARY*	19
DEAD	DEAD	DEAD	22

Käfer gewählt.

3. Sind Attributwerte atomar oder mengenwertig?

Wie schon oben aufgezeigt, ist es möglich, daß Attributwerte, die in einer normalen Relation atomar sind, ein anderes Verhalten zeigen, wenn sie über einen Zeitraum aufgezeichnet werden.

4. Wie wird die Transaktionszeit zugeordnet?

Genau wie die Gültigkeitszeit kann auch die Transaktionszeit sowohl Tupeln als auch Attributen zugeordnet werden. Meistens wird die Transaktionszeit in Form eines zusätzlichen Zeitstempels zugeordnet. Es besteht aber auch die Möglichkeit die Transaktionszeit einer Menge von Tupeln zu zu ordnen. Ben Zvi führte den Begriff der Tupelversionsmengen ein [4]. Er bezeichnet damit alle Tupel einer Relation, die den gleichen Schlüsselwert haben. Das Modell von Ben Zvi wird später noch ausführlicher vorgestellt. Es ist also möglich, die Transaktionszeit nur über diesen Mengen von Tupeln zu betrachten.

2.2 Temporale Algebren

Der erste, der eine Zeitorientierte Algebra definierte, war Jones [42]. LEGOL 2.0 ist eine Sprache, die für den Gebrauch in Datenbankapplikationen, wie High-Level Systemspezifikationen, bei denen die temporale Ordnung und die Gültigkeitszeiten der Objekte eine Rolle spielen, geschaffen. Viele Eigenschaften finden sich in späteren Algebras wieder. Die Objekte in LEGOL 2.0 sind normale Relationen, die jedoch zwei zusätzliche Attribute *start* und *stop* besitzen. Diese Attribute repräsentieren das Intervall der Existenz des realen Objektes. Die Mengenoperationen bleiben in dieser Algebra unverändert. Die neuen zeitorientierten Operationen sind nicht formal definiert, sondern durch Beispiele beschrieben. Die neuen Operationen sind *Zeitdurchschnitt, einseitiger Zeitdurchschnitt, Zeitvereinigung, Zeitdifferenz* und *Zeitmengenmitgliedschaft.* Der Zeitdurchschnitt fungiert als temporaler Join, wobei die Gültigkeitszeit jedes Ergebnistupels mit Durchschnittssemantik berechnet wird. Die Semantik der anderen Operatoren bleibt weitestgehend unklar. Es wird je-

doch auch eine Art temporaler Selektion und ein Join mit Vereinigungssemantik unterstützt.
Ben Zvi war der erste, der mit seiner Algebra Transaktionszeit unterstützte [4]. Auch die Definition von Ben Zvi's Operationen ist etwas vage und schöpft die Möglichkeiten des Datenmodelles nicht aus. Die Ergebnisse der Operationen sind stets normale Schnappschußrelationen und keine historischen Relationen mehr. Ein weiteres Konzept in Ben Zvi's Algebra ist das Konzept der Tupelversionsmengen. Dabei wird die Geschichte eines Objektes durch eine Menge von nach Zeit geordneten Tupeln beschrieben. Käfer nennt diese Mengen Zeitsequenzen [48]. In einer solchen Zeitsequenz haben alle Tupel den selben zeitinvarianten Schlüssel. Dieser Ansatz, der von Käfer und Ben Zvi genutzt wird, erfordert eine Neudefinition aller relationalen Operatoren, so daß sie auf einer Menge von Zeitsequenzen anstatt auf einer Menge von Tupeln arbeiten. Relationen, die Mengen von Zeitsequenzen sind, erfüllen die erste temporale Normalform [3]. Das heißt, daß die Gültigkeitsintervalle der Tupel in einer Zeitsequenz verschieden, aber adjazent sein müssen. Außerdem müssen alle paarweise verschiedenen Tupel mindestens einen Unterschied bei zeitvarianten Attributen aufweisen. Die Neudefinition der relationalen Operatoren versucht Ben Zvi zu umgehen, indem er die zeitvarianten Relationen vor der Anwendung von herkömmlichen relationalen Operationen einfach in eine normale Schnappschußrelation konvertiert. Das Ergebnis dieser Operationen ist natürlich wieder eine Schnappschußrelation. Diese Algebra ist also nicht geschlossen und in ihrem Funktionsumfang unzureichend.
Orgun und Müller gehen einen ähnlichen Weg. Ein zeitvariante Relation ist definiert als eine Sammlung von endlich vielen Relationen, die durch einen Zeitmoment indiziert sind. Auch dieser Ansatz erfordert eine Neudefinition aller relationalen Operatoren. Zusätzlich enthält die Algebra die Operationen *first, next* und*prev*, um durch eine zeitvariante Relation zu navigieren. Weiterhin wird ein temporaler Join eingeführt, der es erlaubt, zeitvariante Daten von verschiedenen Zeitpunkten zu verbinden. Diese Join-Operation hat nichts mit der bekannten relationalen Join-Operation zu tun. Der temporale Join $[x, y]E$ findet alle Tupel in E, die über das

ganze Intervall $[x, y]$ gültig sind.

2.2.1 Temporale Selektion

In einer temporalen Algebra hat die Selektion die Aufgabe, genau wie in der relationalen Algebra, eine Relation auf die Tupel zu begrenzen, die ein bestimmtes Kriterium erfüllen, das durch eine boolsche Formel gegeben ist. Diese boolsche Formel muß in einen temporalen Kontext gesetzt werden. Die Formel wird nur mit Daten berechnet, die im gegebenen temporalen Kontext gültig sind. Nimmt man ein Zeitintervall als temporalen Kontext, gibt es zwei verschiedene Interpretationen. Erstens, die boolsche Formel ist wahr während des ganzen Intervalls. Zweitens, die Formel gilt mindestens an einem Punkt im Intervall. Clifford bringt diese beiden Interpretationen zum Beispiel in einer einzigen Formel unter, indem als Parameter ein Quantor (entweder $\forall$ oder $\exists$) übergeben wird [22, 21]. Der Allquantor gibt an, daß das Selektionskriterium über das ganze Intervall gültig sein muß, und der Existensquantor bezeichnet dementsprechend, daß das Kriterium nur an einem Punkt im Intervall erfüllt werden muß. Auch Gadia [44] verbindet die Selektionsbedingung mit einem temporalen Kontext und führt so gleichzeitig eine Selektion als auch eine Projektion auf der Zeitachse durch. Auch Snodgrass und McKenzie [38, 12] definieren eine solche kombinierte Operation, die *Ableitungsoperation* (derivation operation) genannt wird.
Vielfach wird die Definition der Selektionsoperation auch aus der Relationenalgebra übernommen, wenn das Datenmodell es erlaubt. Der temporale Kontext bei der Selektion entfällt dadurch natürlich. Diesen Weg gehen zum Beispiel Dey, Barron und Storey [7]. Dafür wird eine temporale Projektion eingeführt, die eine Relation auf einen bestimmten temporalen Kontext beschränkt. Die Nacheinanderausführung dieser beiden Operationen bringt also auch das gewünschte Ergebnis. Clifford definiert beides, sowohl eine Selektionsoperation mit temporalem Kontext als auch eine Operation, um eine Relation entlang der Zeitachse zu reduzieren [22, 21]. In einer temporalen Datenbank ist es aber auch interessant zu erfragen, zu

welcher Zeit eine bestimmte Bedingung erfüllt ist. Clifford führt für diese Art der Selektion den Operator *Select-When* ein. Das Ergebnis dieser Operation ist die Menge der Zeiten, zu denen die Selektionsbedingung erfüllt ist. Sadeghi definiert eine Operation *When*, die eine historische Relation auf die Zeitintervalle abbildet, die Zeitstempel der Tupel der Relation sind. In Kombination mit der normalen Selektion läßt sich der gleiche Effekt wie mit Cliffords Select-When erreichen.

2.2.2 Temporale Projektion

In jeder temporalen Algebra wird, genau wie in der Relationenalgebra, eine Operation benötigt, mit deren Hilfe man eine Relation auf bestimmte Attribute begrenzen kann. Die Definition dieser Operation muß nur modifiziert werden, wenn das zugrundeliegende Datenmodell eine Übernahme der Definition aus der Relationenalgebra nicht erlaubt. Die Semantik der Operation bedarf keiner Veränderung. Allerdings verknüpft zum Beispiel Käfer die Operation mit einem temporalen Kontext. Dieses Vorgehen kann zu Verirrungen führen, weil auch schon die Selektion bei Käfer in einem temporalen Kontext steht [48]. Da Selektion und Projektion meistens zusammen ausgeführt werden, würde die Spezifikation einer temporalen Eingrenzung ausreichen. Das Auftreten von zwei vielleicht verschiedenen temporalen Kontexten kann hier höchstens zur Verwirrung führen.
Es ist also nützlich, den temporalen Kontext ganz aus der Selektion und Projektion herauszuhalten und eine Relation durch eine spezielle Operation entlang der Zeitachse zu begrenzen. Diese Operation wird in manchen Algebren temporale Projektion (Dey, Barron, Storey), bei Orgun temporaler Join, in den meisten jedoch Time-Slice oder Slice (Clifford, Tansel, Navathe) genannt. Wenn ein Intervall der Parameter einer solchen temporalen Projektion ist, gibt es natürlich die Möglichkeit, daß die die Zeitstempel der Ergebnisrelation mit Hilfe des Parameters und des Originalzeitstempels mittels Vereinigungs-, Durchschnitts- oder Differenzsemantik berechnet werden. Tansel definiert deshalb gleich drei Operationen, Slice, USlice und DSlice, um alle diese Möglichkeiten zur Verfügung

zu stellen. Meistens wird jedoch einfach die Durchschnittssemantik verwendet.

2.2.3 Temporales kartesisches Produkt

Für Modelle, bei denen Attributtimestamping benutzt wird, kann die Definition des kartesischen Produktes von der Relationenalgebra übernommen werden [38]. Clifford, der in seinem HRDM sowohl Attribut- als auch Tupeltimestamping benutzt, bestimmt die Lebensspanne der Ergebnistupel aus der Vereinigung der Lebensspannen der Operandentupel. Das hat zur Folge, daß das Ergebnistupel Nullwerte enthalten kann. Wenn aus zwei Relationen $R1$ und $R2$ ein kartesisches Produkt, $R1 \times R2$, gebildet wird, enthalten in jedem Ergebnistupel die Attribute aus $R1$ *Null*, zu allen Zeiten, die nicht in der Lebensspanne des Ausgangstupels aus $R1$ liegen. Analog wird mit Attributen aus $R2$ verfahren [22, 21]. Solche Nullwerte entstehen nicht, wenn bei einem Modell mit Tupeltimestamping der Zeitstempel des Ergebnistupels aus dem Durchschnitt der Zeitstempel der Operandentupel gebildet wird. Sarda behandelt in seiner Algebra für die Sprache HSQL die Zeitstempel der Tupel bei der Bildung des kartesischen Produktes nicht gesondert [36]. Das hat zur Folge, daß die Tupel der Ergebnisrelation zwei Zeitstempel haben, was bedeutet, daß das Ergebnis keine gültige historische Relation mehr ist. Dafür wird eine andere Operation, *CP* (Concurrent Product), eingeführt. Diese Operation berücksichtigt nur die Tupel der Ausgangsrelationen, deren Zeitintervalle sich überlappen. Der Zeitstempel der Ergebnistupel ist der Durchschnitt der Zeitstempel der Operandentupel. Mit dieser Operation könnte Sarda eine temporale Join-Operation definieren. Jedoch verzichtet er darauf. Es scheint so, daß die Join-Operation in Sardas Algebra wie in der traditionellen Relationenalgebra durch die Selection und das kartesische Produkt ausgedrückt werden soll. Das kann aber, wie oben bereits angedeutet, kein befriedigendes Ergebnis liefern, da das Ergebnis keine gültige historische Relation wäre. Auch bei Cliffords HRDM gibt es Unstimmigkeiten bei der Definition der Join-Operation und des kartesischen Produktes. In der Re-

lationenalgebra kann die Join-Operation auch durch $Rjn_F S = \sigma_F(R \times S)$ ausgedrückt werden. Cliffords Join-Operation kann jedoch nicht mit Hilfe seines kartesischen Produkt-Operators ausgedrückt werden, da der Join den Zeitstempel der Ergebnistupel aus dem Durchschnitt der Zeitstempel der Operandentupel bildet, wohingegen beim kartesischen Produkt die Vereinigung benutzt wird. Die Algebra Cliffords ist damit keine pure Erweiterung der Relationenalgebra.
Ein gravierender Unterschied des kartesischen Produktes in temporalen Algebren zu dem der Relationenalgebra ist, daß die Ergebnisrelation nicht $n*m$ Tupel enthalten muß, wenn n und m die Kardinalitäten der Originalrelationen sind, es sei denn, die Tupel des Ergebnisses dürfen Nullwerte enthalten.

2.2.4 Temporaler Join

Die Joinoperation gehört nicht zu den Grundoperationen der Relationenalgebra. Wie oben bereits beschrieben, läßt sich ein Join mit Hilfe der Selektion und dem kartesischen Produkt ausdrücken, $R \bowtie_F S = \sigma_F(R \times S)$. Diese Definition wird in die meisten temporalen Algebren übernommen (z.B. Lorentzos, Dey, Tansel). Dieser Ansatz ergibt sich aus dem Bestreben, eine Algebra zu definieren, die eine konsistente Erweiterung der Relationenalgebra ist [27]. Einige andere Autoren (z.B. Clifford, Navathe, Orgun) geben eine andere Definition des Joins an. Navathe definiert zum Beispiel gleich vier Joinoperationen [41]. Eine Joinoperation betrachtet nur Tupel, deren Zeitstempel sich überlappen und behält beide Zeitstempel im Ergebnis. Das ist problematisch, da eine Relation, die Tupel mit zwei Zeitstempeln enthält, kein gültiges Objekt der Algebra ist. Das gleiche gilt für die Definition einer Natural Join Operation. Navathe definiert jedoch beide Operationen noch einmal so, daß das Ergebnis nur noch einen Zeitstempel enthält, der sich aus dem Durchschnitt der Ausgangsstempel bildet. Der allgemeine Fall des Θ-Joins wird jedoch überhaupt nicht definiert. Es wird auch kein Beispiel für die Operationen angegeben, die ungültige Relationen produzieren. Clifford definiert, wie im Abschnitt

über das kartesische Produkt bereits erwähnt, eine Joinoperation mit Durchschnittssemantik und ein kartesisches Produkt mit Vereinigungssemantik. Zusätzlich definiert Clifford noch einen *Time-Join*. Diese Operation ist nur für Attribute definiert, deren Domäne die Zeit ist. Bei dieser Operation werden beide Operandenrelationen verbunden und auf die Zeit des angegebenen Joinattributes projiziert.

2.2.5 Andere Operationen

Entsprechend der vielfältigen Datenmodelle, wurden in den Verschiedenen Algebren viele andere und neue Operationen eingeführt. In vielen Vorschlägen gibt es Operationen wie *PACK/UNPACK* oder *FOLD/UNFOLD*. Durch das „Entpacken“ oder „Entfalten“ einer Relation werden in der Regel aus einem Tupel viele wertäquivalente Tupel erzeugt, zum Beispiel für jedes Chronon im Gültigkeitsintervall eins. PACK und FOLD sind dann die Umkehroperationen dazu. Diese Operationen dienen eigentlich nur dazu eine andere Sicht auf die Daten zu generieren und beschreiben nicht die expressive Stärke einer Sprache [7]. Solche Operationen werden von Tansel, Lorentzos und Sarda verwendet. Dey und Snodgrass umgehen die Verwendung solche Ausdrücke durch die Einführung von Hilfsfunktionen. Dey führt eine *COALESCENCE*-Operation ein und Snodgrass die Operation *REDUCE*. Beide Operationen sind Bestandteil jeder Definition der Algebra und verhindern das Auftreten wertäquivalenter Tupel. Das macht PACK- und FOLD-Operationen überflüssig.
Eine andere Operation, die von einigen Autoren eingeführt wird, ist *DROP-TIME*. Diese Operation überführt eine historische Relation in eine Schnappschußrelation. Tansel führt zum Beispiel eine solche Operation ein.
Clifford führt eine Operation, *WHEN*, ein. Diese Operation reduziert eine Relation auf ihre Lebensspanne, indem die Vereinigung aller Lebensspannen der Tupel berechnet wird. Clifford will die Ergebnisse dieser Operation als Parameter für andere Operationen verwenden.

2.2.6 Bewertung temporaler Algebren

Die Bewertung temporaler Algebren wurde in der Forschung lange vernachlässigt. Erst Anfang der 90er Jahre wurde die Forschung auf diesem Gebiet forciert. So schlagen zum Beispiel Clifford und Crooker in [25] Kriterien vor, um die Abgeschlossenheit von temporalen Algebren zu bewerten. Eine temporale Algebra gilt danach als vollständig, wenn sie die Ausdruckskraft der in [25] vorgeschlagenen Sprache hat. Clifford und Crooker unterscheiden dabei zwischen TG-Modellen (temporally grouped) und TU-Modellen (temporally ungrouped). TG-Modelle sind Modelle, bei denen die Relationen aus sogenannten Tupelversionsmengen oder temporalen Sets bestehen, deren einzelne Tupel alle den gleichen zeitinvarianten Schlüsselwert haben. TU-Modelle sind in der Regel solche mit Attributtimestamping. Laut [25] gibt es keine wirklich vollständige Algebra für TG-Modelle. Eine Algebra für ein solches Modell ist TG-vollständig, wenn sie der in [25] vorgestellten Sprache entspricht.
Snodgrass und McKenzie schlagen in [27] 26 Kriterien vor, um eine temporale Algebra zu bewerten. Sie erweitern damit die fünf Kriterien, die von Clifford und Tansel [26] vorgeschlagen wurden.

1. *Alle Attributwerte eines Tupels sollen für das gleiche Intervall definiert sein.* Diese Forderung wird auch Homogenität genannt, und wird von Gadia eingeführt [43]. Bedeutung bekommt dieses Kriterium, wenn Attribute mengenwertig sind oder Zeitstempel auf der Attributebene vergeben werden. Für jedes Chronon, zu dem ein beliebiges Attribut einen Wert hat, sollen auch alle anderen Attribute des Tupels einen gültigen Wert haben. Das vereinfacht die Algebra. Trivialer Weise ist dieses Kriterium erfüllt, wenn Tupeltimestamping verwendet wird.

2. *Die Algebra soll eine konsistente Erweiterung der Relationenalgebra sein.* Diese Forderung wurde schon von Clifford und Tansel [26] aufgestellt. Die Algebra sollte in der Ausdruckskraft genauso stark sein, wie die Relationenalgebra. Anfragen, die möglich sind, wenn

die Zeit nicht modelliert wurde, sollen auch erlaubt sein, wenn der Zeitaspekt hinzugefügt wird. Jeder algebraische Ausdruck der Relationenalgebra sollte einen Konterpart in der temporalen Algebra haben.

3. *Periodizität soll unterstützt werden.* Dieses Kriterium wird von Lorentzos schon in [35] behandelt. Viele Phänomene der realen Welt treten in Perioden auf. Idealer Weise unterstützt ein temporales Datenmodell solche periodischen Phänomene, ohne daß jede Zeit ihres Auftretens spezifiziert werden muß, und temporale Operationen sollten solche periodischen Daten direkt manipulieren können.

4. *Jede Sammlung von legalen Attributwerten soll ein legales Tupel sein.* Dieses Kriterium bedeutet, daß, genau wie in der Relationenalgebra, die Attribute unabhängig voneinander sind, außer Schlüsselwerte und funktionale Abhängigkeiten.

5. *Die formale Semantik soll wohl definiert sein.* Eine korrekte mathematische Definition alle Objekte und Operationen der Algebra ist notwendig, damit die Bedeutung der algebraischen Operationen klar ist.

6. *Eine temporale Algebra soll die Ausdruckskraft eines historischen oder temporalen Kalküls haben.* Dieses Kriterium bedeutet, daß es eine kalkülbasierte Sprache als Benutzerschnittstelle eines DBMS geben sollte, deren Ausdruckskraft durch die Algebra ausgedrückt wird, die dann als Bewertungsmechanismus benutzt werden kann [43].

7. *Durchschnitt,*Θ*-Join, Natural Join und Quotient sollen definiert sein.* In der Relationenalgebra sind diese Operationen mit Hilfe der Grundoperationen, Differenz, Selektion, Projektion und kartesisches Produkt definiert. In einer temporalen Algebra können diese Definitionen unter Umständen verändert werden, wenn die Konterparts der Operationen der Relationenalgebra in der temporalen

Algebra nicht deren Eigenschaften erfüllen.

8. *Die temporale Algebra soll eine wirkliche Algebra sein.* Dieses Kriterium, das von Clifford und Tansel [26] aufgestellt wird, ist fundamental. Es beinhaltet auch, daß die Algebra geschlossen sein soll.

9. *Das Modell soll keine nullwertigen Attribute erfordern.* Wenn fehlende oder unbekannte Informationen aufgezeichnet werden sollen, sind Nullwerte unverzichtbar, aber die Algebra sollte solche Werte nicht als semantisches Hilfsmittel benutzen (wie z.B. Clifford). Das vereinfacht die Semantik der Algebra.

10. *Multidimensionale Zeitstempel sollen unterstützt werden.* Die Forderung wird von Gadia und Yeung aufgestellt [45]. Dieses Kriterium ermöglicht es, mehr als einen Aspekt der Zeit zu modellieren. Ein Beispiel für eine zweite Zeitdimension ist die Transaktionszeit.

11. *Die temporale Algebra soll sich auf die Relationenalgebra reduzieren.* Snodgrass führt dieses Kriterium in [37] ein. Wenn der Zeitaspekt außer acht gelassen wird, soll die Algebra genauso arbeiten, wie die Relationenalgebra.

12. *Die Relationen sollen in der ersten Normalform sein.* Die Relationenalgebra verdankt dieser Restriktion viel von ihrer Einfachheit. Jede Erweiterung der Relationenalgebra sollte diese Eigenschaft übernehmen.

13. *Die Algebra und das Modell sollen eine dreidimensionale konzeptionelle Sicht der historischen Relationen und Operationen unterstützen [17, 18, 26].* Temporale Relationen können als dreidimensionale Quader dargestellt werden, mit der Zeit als dritter Dimension. Dieses Kriterium bedeutet auch, daß eine Operation vorhanden sein sollte, die einen zweidimensionalen Schnappschuß einer historischen Relation liefert.

14. *Die Algebra unterstützt die grundlegenden algebraischen Äquivalenzen.* Dieses Kriterium fordert, daß die Kommutativität und Distributivität von algebraischen Ausdrücken der Relationenalgebra auch für die entsprechenden Ausdrücke der temporalen Algebra gilt. Damit ist die Algebra einfacher zu implementieren, und Optimierungsmechanismen können von der Relationenalgebra übernommen werden.

15. *Die Algebra soll Relationen aller vier Klassen unterstützen.* Eine Algebra, die Gültigkeits- und Transaktionszeit unterstützt, sollte alle vier Arten von Relationen, Schnappschußrelationen, Rollbackrelationen, historische Relationen und temporale Relationen (siehe Abschnitt 2.1) unterstützen.

16. *Die Algebra soll Rollbackoperationen unterstützen [4, 39].* Die Algebra sollte Anfragen an einen früheren Status der Datenbank erlauben, besser noch an mehrere. Dieses Kriterium kann nur erfüllt werden, wenn Transaktionszeit unterstützt wird.

17. *Die Algebra soll mehrfach gespeicherte Schemata unterstützen [4, 27].* Dieses Kriterium bezieht sich auf die sogenannte Schemaevolution. Im Laufe der Zeit kann sich ein Relationenschema verändern. Attribute können hinzugefügt, gelöscht oder verändert werden. Um eine Rollbackoperation auf solch einer Relation durchzuführen, ist es notwendig, die Veränderungen des Schemas abzuspeichern.

18. *Die Algebra sollte statische Attribute unterstützen [26].* Dieses Kriterium besagt nur, daß auch zeitunabhängige Attribute unterstützt werden sollen.

19. *Die Algebra soll Gültigkeits- und Transaktionszeit orthogonal behandeln [39].* Gültigkeits- und Transaktionszeit sind orthogonale Aspekte der Zeit. Gültigkeitszeit gibt an, wann Ereignisse auftreten und Beziehungen in der realen Welt existieren. Die Transaktionszeit

gibt an, wann solche Ereignisse und Beziehungen in der Datenbank gespeichert werden. Die Gültigkeitszeit, die einem Objekt zugeordnet wird, sollte nicht durch die Transaktionszeit bestimmt oder beschränkt werden. Die Algebra sollte sowohl retroaktive als auch postaktive Änderungen der Datenbank erlauben (siehe Abschnitt 2.1).

20. *Tupel und nicht Attributwerte sollen die Zeitstempel enthalten.* Tupeltimestamping vereinfacht die Semantik der Algebra. Es müssen keine Operatoren definiert werden, die disjunkte Attributzeitstempel behandeln. Außerdem kann die Definition der Operationen näher an die Relationenalgebra angelehnt werden.

21. *Einmalige Repräsentation für jede temporale Relation.* Genau wie in der Relationenalgebra soll eine Relation in der temporalen Algebra nur eine Repräsentation haben. Andererseits wird die Semantik kompliziert und die Implementation ineffizient.

22. *Die Algebra soll unisorted sein (nicht multisorted).* Alle Operationen sollen nur einen Typ von Objekt als Parameter und als Ergebnis haben.

23. *Die Algebra soll Updatesemantik spezifizieren [37].* Ohne die mathematische Definition von Updateoperationen in der Algebra bleibt deren Auswirkung und Funktionsweise unklar.

Alle der oben beschriebenen Kriterien sind jedoch nicht kompatibel. So kann eine Algebra zum Beispiel nicht gleichzeitig ein dreidimensionales Modell unterstützen (16) und gleichzeitig Tupeltimestamping (23). Das kartesische Produkt oder der Joinoperator einer Algebra, die solch ein Modell unterstützt, müßte die Tupel unabhängig von ihrer Gültigkeitszeit aneinanderhängen und im Ergebnis die Gültigkeitszeit jedes unterliegenden Tupels repräsentieren. Die Alternative ist Datenverlust. Wenn aber das kartesische Produkt an Attribute unterschiedliche Zeitstempel vergibt, kann das Kriterium des Tupeltimestampings nicht erfüllt werden. Daraus folgt auch, daß Kriterium (1), daß alle Attributwerte eines

Tupels über das gleiche Intervall definiert sein sollen, nicht erfüllt werden kann in einem streng dreidimensionalen Modell (16).
Eine Algebra, die ein streng dreidimensionales Modell unterstützt, kann die Distributivität des kartesischen Produktes über der Differenz (17) nicht gleichzeitig unterstützen [27].
Außerdem ist das Kriterium, daß jede legale Menge von Tupeln eine legale Relation ist (5), inkompatibel mit der Forderung, daß es nur eine Repräsentation für jede Relation gibt (24). Wenn eine Algebra nur eine Repräsentation für eine Relation, zum Beispiel A_1, erlaubt, dann ist eine andere Repräsentation derselben Relation (zum Beispiel A_2) zwar eine legale Tupelmenge, aber keine legale Relation.
Schließlich sind die Kriterien, daß die erste Normalform unterstützt wird (15), daß ein dreidimensionales Modell unterstützt wird (16) und daß eine einzige Repräsentation für eine Relation erlaubt ist (24), nicht vereinbar. Eine Algebra kann immer nur zwei dieser drei Kriterien erfüllen, aber nicht alle drei.
Eine Algebra, die unter Berücksichtigung der Inkompatibilitäten alle diese Kriterien erfüllt, wird zum Beispiel von Dey [7] vorgestellt.

2.3 Ben-Zvi's Pionierarbeit

Jacov Ben-Zvi war Informatikstudent an der University of California, Los Angeles, als er in der Zeit von 1979 bis 1981 ein temporales Datenbankmodell entwickelte. Ben-Zvi's Dissertation stellte ein Modell für temporale Datenbanken, genannt *Zeitrelationenmodell*, eine temporale Anfragesprache und eine Speicherarchitektur und deckte auch die Themen temporale Indexe, Recovery, Konkurrenz und Synchronisation und Implementation ab [4].
Ben-Zvi hatte Ideen, die noch heute das Forschungsgebiet der temporalen Datenbanken bestimmen. Vielleicht die wichtigste war, daß sein Modell nicht die erste Normalform berücksichtigte. In der Zeit, in der Ben-Zvi seine Arbeit vorstellte, galt bei vielen die erste Normalform als das letzte Wort auf dem Gebiet der relationalen Datenbanken. Er führte zeitinvari-

ante Schlüsselwerte ein. Er erkannte als erster die zweidimensionale Natur der Zeit in Datenbanken. Er unterschied zwischen Fehlern und Änderungen.

2.3.1 Das Datenmodell

Zeit

Zeit besteht in Ben-Zvi's Modell aus Instanzen zusammen mit einer linearen Ordnung, die festlegt, daß eine Instanz vor einer anderen liegt. Eine solche Zeitinstanz wird durch ein Paar aus Datum und Uhrzeit repräsentiert. Er führte das Konzept der *Effektivzeit* und der *Registrationszeit* ein. Heute benutzt man die Begriffe *Gültigkeitszeit* und *Transaktionszeit*. Ben-Zvi's Modell ist also ein bitemporales Modell. Das Universum von valid time und transaction time erstreckt sich von einem festen Startpunkt ins Unendliche.

Zeitperiodenmengen

Eine *Zeitperiode* ist einfach ein Intervall von Zeit. Eine *Zeitperiodenmenge* ist eine Menge von paarweise disjunkten Zeitperioden. Die *Vereinigung*, der *Durchschnitt* und die *Differenz* von Zeitperiodenmengen ist entsprechend definiert.

Timestamping von Daten

Timestamping wird auf dem Tupellevel ausgeführt. Fünf Timestamps werden an einen Datenwert $d = (a_1, a_2, ..., a_n)$ vergeben:

- T_{es}, Start der Effektivzeit,
- T_{rs}, Start der Registrationszeit,
- T_{ee}, Ende der Effektivzeit,
- T_{re}, Ende der Registrationszeit,

- T_d, Zeit der Löschung.

Die Werte T_{es} und T_{ee} werden vom Nutzer angegeben, wogegen die Werte für T_{rs} und T_{re} von der Systemuhr abgefragt werden, und zwar wenn ein Tupel in die Datenbank eingefügt wird, oder wenn es nicht mehr registriert, oder wenn es geändert wird. T_{es} kann gleich, größer oder kleiner T_{rs} sein. In den letzten beiden Fällen spricht man von proaktiven beziehungsweise retroaktiven Datenänderungen. Wenn ein existierendes Tupel fehlerhaft ist, kann man es löschen, indem man die Zeit T_d setzt. Das gibt dem Modell die Möglichkeit, zwischen Änderungen und Fehlern zu unterscheiden.
Ein Tupel wie $(a_1, a_2, ..., an, T_{es}/T_{rs}, T_{ee}/T_{re}, T_d)$ heißt Tupelversion in Ben-Zvi's Terminologie.

Tupelversionsmengen

Ben-Zvi's Modell enthält das Konzept von Schlüsseln. Die Menge aller Tupel, die den gleichen Schlüsselwert haben, sind eine *Tupelversionsmenge*. Ein Beispiel für eine Tupelversionsmenge ist in Abbildung 2.4 angegeben.

Abbildung 2.4: Die Relation EMP mit Tupeltimestamping

ENAME	GEHALT	MANAGER	T_{es}/T_{rs}	T_{ee}/T_{re}	T_d
TOM	$14K$	*RON*	15/16	19/19	-
TOM	$14K$	*GARY*	19/19	22/23	-
TOM	$14K$	*AL*	22/21	25/25	-
TOM	$18K$	*AL*	25/25)	-/-	-
ANN	$12K$	*LIZ*	10/10	-/-	-

Zeitrelationen

Eine Zeitrelation ist definiert als eine Zusammenfassung von Tupelversionsmengen (*nicht* von Tupelversionen). Daher ist Ben-Zvi's Modell nicht in erster Normalform.

2.3.2 Operationen

Der Time View Operator

Der *Time View Operator* ist eine bitemporale Instanz (T_e, T_s), wobei T_e die Effektivzeit ist und T_s die Zeit, zu der die Operation ausgeführt wird. $(T_e, T_s)(r)$ ist der Time View einer Relation r; das ist der T_e*Schnappschuß* der Relation r wie sie zum Zeitpunkt T_s bekannt ist. Das heißt, von einer gegebenen Tupelversion $(a_1, a_2, ..., an, T_{es}/T_{rs}, T_{ee}/T_{re}, T_d)$ wird $(a_1, a_2, ..., an)$ vom Time View Operator (T_e, T_s) genau dann zurückgegeben, wenn folgende Bedingungen erfüllt sind:

- $(T_s < T_d)$ - das heißt, das Tupel ist nicht gelöscht - und
- $(T_s \in (T_{rs}, T_{re}) \wedge T_e \geq T_{es}) \vee (T_s \geq T_{re} \wedge T_e \in [T_{es}, T_{ee}])$.

Der voreingestellte Time View

Der Nutzer muß keinen Time View angeben. Wird vom Anwender kein Time View spezifiziert, wird der Time View $(now, current)$ benutzt. $Current$ gibt dabei die Zeit der letzten Transaktion an. Das heißt $current \leq now$.

Algebraische Operatoren

Ben-Zvi führt die Operatoren *time-union, set-difference, time selection, time projection* und *time-join* ein. Die Definition dieser Operatoren ist etwas vage. Um sie anzuwenden, ist es notwendig, erst den *Time-View*-Operator anzuwenden. Zum Beispiel ist der Operator *time-union* folgendermaßen definiert:

Seien r und s Zeitrelationen, dann ist $r(T_e, T_s)$ *time-union* s definiert als $(T_e, T_s)(r) \bigcup (T_e, T_s)(s)$.
Andere Operatoren sind ähnlich definiert. Daher ist das Resultat dieser Operationen für gewöhnlich eine Schnappschußrelation.

2.4 Cliffords historisches relationales Datenmodell (HRDM)

Clifford war ebenfalls einer der Pioniere im Bereich der temporalen Datenbanken. Er war der erste, der vorschlug, die temporale Dimension durch die Attribute zu repräsentieren. 1987 entwickelten Clifford und Crooker das HRDM [22, 21], ein Datenmodell, das dahingehend einzigartig ist, daß Zeitstempel sowohl an Attribute als auch an Tupel vergeben werden. Für dieses Modell wurde eine Algebra definiert, die die traditionellen Mengenoperationen erweitert, um die temporale Dimension des Modells zu berücksichtigen. Korrespondierend zu den unären Operatoren für die Wert- und die Attributdimension, *select* und *project*, wird ein unärer Operator für die temporale Dimension, *time-slice*, eingeführt. Der *join*-Operator wurde erweitert, und ein neuer Operator *when* eingeführt, um rein temporale Daten zu ermitteln. Doch zunächst ein Blick auf das Datenmodell.

2.4.1 Das Datenmodell

Zeit ist in Cliffords Modell eine Menge von *Chronons* $T = \{\ldots, t_0, t_1, \ldots\}$. Über diese Menge ist eine totale lineare Ordnung $<_T$ definiert, wobei $t_0 <_T t_1$ bedeutet, daß t_0 vor t_1 liegt.
Eine Lebensspanne L kann Tupeln zugeordnet werden und ist eine beliebige Teilmenge von T. Lebensspannen können auch abgeleitet werden. Wenn L_1 und L_2 Lebensspannen sind, dann sind auch

- $L_1 \cup L_2$,
- $L_1 \cap L_2$,

- $L_1 - L_2$ und
- $\neg L_1$

Lebensspannen.
Weiterhin wird eine Menge der Wertedomänen $D = \{D_1, D_2, \ldots, D_n\}$ analog zum Relationenmodell definiert. Um Attributen eine Lebensspanne zuzuweisen, wird die Menge $TD = \{TD_1, TD_2, \ldots, TD_n\}$ definiert mit $TD_i = \{f_i | f_i : T \to D_i\}$. Die Menge TD stellt die Menge aller partiellen Funktionen von T in die Wertedomäne D. Ähnlich wird die Menge $TT = \{g | g : T \to T\}$ als Menge aller partiellen Funktionen von T in sich selbst definiert, um explizit zu unterscheiden, welche Werte Zeit repräsentieren und welche nicht.
Wie im Relationenmodell existiert ein Universum $U = \{A_1, A_2, \ldots, A_n\}$ von Attributen. Die Attribute sind über den historischen Domänen $HD = (TD \cup \{TT\})$ definiert. Schlüsselattribute müssen zeitinvariant sein. CD ist die Menge der aller zeitinvarianten Domänen.
Wird eine Funktion f mit der Domäne D auf eine kleinere Domäne $D' \subset D$ eingeschränkt, wird das als $f|_{D'}$ notiert.

Mit diesen Voraussetzungen wird nun ein historisches Relationenschema $R =< A, K, ALS, DOM >$ als geordnetes 4-Tupel definiert, wobei

- $A = \{A_{R_1}, A_{R_2}, \ldots, A_{R_n}\} \subset U$ die Menge der Attribute von R,
- $K = \{A_{K_1}, A_{K_2}, \ldots, A_{K_m}\} \subset A$ die Menge der primären Schlüsselattribute von R,
- $ALS : A \times R \to 2^T$ die Funktion, die jedem Attribut seine Lebensspanne zuweist und
- $DOM : A \to HD$ die Funktion, die jedem Attribut eine Domäne zuordnet - mit der Einschränkung $DOM(A_{K_i}) \in CD \; \forall i \; 1 \leq i \leq m$ - ist.

Ein Tupel t über R ist ein geordnetes Paar $t =< v, l >$, wobei

- $t.l$ die Lebensspanne und
- $t.v$ der Wert des Tupels ist.

Der Wert des Tupels wird durch eine Abbildung $t.l \cap ALS(A, R) \rightarrow DOM(A)$ bestimmt.
Weil sowohl den Tupeln als auch den Attributen eine Lebensspanne zugeordnet wird, ergibt sich die Notwendigkeit, die Lebensdauer eines Attributwertes in einem Tupel herzuleiten. Das geschieht bei Clifford folgendermaßen:
Die Lebensspanne eines Wertes (vls) des Attributs A im Tupel t über dem Schema R ist definiert als $vls(A, t, R) = t.l \cap ALS(A, R)$. Für A kann auch eine Attributmenge eingesetzt werden. Die Lebensdauer der Werte der Attributmenge ergibt sich dann wiederum aus dem Durchschnitt der Lebensdauer aller Attribute.
Kurz schreibt Clifford für den Wert eines Attributs A im Tupel t zur Zeit s $t(A)(s)$.

2.4.2 Die Algebra

Um einige Operationen der Algebra zu erläutern, soll hier zunächst eine Minidatenbank als Beispiel angegeben werden.

Mengenoperationen

Die Mengenoperationen *cap*, *cup* und $-$ sind in Ihrer Definition dem Relationenmodell entnommen und sind auch im *HRDM* auf *vereinigungskompatible* Relationen eingeschränkt.
Das kartesische Produkt zweier Relationen $r1$ und $r2$ ist definiert als

Abbildung 2.5: Die Relationen EMP und MAN

ENAME	GEHALT	MANAGER	Lebensspanne
TOM	15 14K 25 18K	15 RON 19 GARY 22 AL	$\{[15, NOW]\}$
ANN	10 12K	10 LIZ	$\{[10, NOW]\}$

MANAGER	MGEHALT	ABTEILUNG	Lebensspanne
RON	15 22K 22 23K	15 SPIELZEUG 19 AUTOS 25 PR	$\{[15, NOW]\}$
GARY	15 19K 19 20K 22 23K	15 PR 19 SPIELZEUG 22 AUTOS	$\{[15, NOW]\}$
AL	22 22K	22 SPIELZEUG	$\{[22, NOW]\}$
LIZ	10 15K	10 WASCHMASCHINEN	$\{[10, NOW]\}$

$$
\begin{aligned}
r1 \times r2 \quad = \quad & \{t \text{ über } R3 | \exists t1 \in r1, \exists t2 \in r2[t.l = t1.l \cup t2.lwedge \\
& \forall A \in R1, \forall s \in t1.l[t.v(A)(s) = t1.v(A)(s)] \wedge \\
& \forall A \in R2, \forall s \in t2.l[t.v(A)(s) = t2.v(A)(s)] \wedge \\
& \forall A \in R1, \forall s \in t.l - t1.l[t.v(A)(s) = \perp] \wedge \\
& \forall A \in R2, \forall s \in t.l - t2.l[t.v(A)(s) = \perp]]\} \\
\text{wobei} \quad & R3 = < A_1 \cup A_2, K_1 \cup K_2, ALS_1 \cup ALS_2, DOM_1 \cup DOM_2 >
\end{aligned}
$$

Diese Definition kann in Nullwerte ($\perp$) resultieren, da die Lebensspanne der Ergebnistupel die Vereinigung der Lebensspanne der Operanden Tupel ist. Das würde nicht passieren, wenn man dem Ergebnistupel den Durchschnitt der Lebensspannen der Operandentupel zuweisen würde.

Projektion

Die Projektion Π bleibt bei Clifford gegenüber dem Relationenmodell unverändert. Sie entfernt einfach alle Attribute einer Relation, bis auf die Attribute, die Parameter der Projektion sind. Wenn also r eine Relation über R ist und $X \subset R$, dann ist eine Projektion von r auf X definiert als

$$\Pi_X(r) = \{t(X) | t \in r\}$$

Selektion

Clifford führt zwei Arten von Selektionen ein. Da jedes Tupel im HRDM aus den Attributwerten selbst und der zugeordneten Lebensspanne besteht, gibt es bei Clifford einmal die Möglichkeit, Tupel über ihre gesamte Lebensdauer auszuwählen und zum anderen, nur eine *relevante* Untermenge der Lebensspannen zu betrachten.
Die erstere Aufgabe erfüllt das *select-if*. Der select-if Operator schränkt eine Relation auf der Wertedimension ein. Wenn eine Selektionsbedingung Θ durch ein Tupel t der Relation r erfüllt wird, so ist das ganze Tupel im Ergebnis der Operation. Da Werte eine Funktion über einer Menge von Zeiten sind, muß auch angegeben werden, für welche Zeiten das Selektionskriterium erfüllt werden muß. Das wird durch existentielle oder universale Quantifikation erreicht. Clifford benutzt die Notation $Q(s \in S)$, wobei $Q \in \{\forall, \exists\}$. Als weiterer Parameter wird eine Lebens-

spanne L übergeben. Die Operation ist wie folgt definiert:

$$\sigma - IF_{(A\Theta a,Q,L)}(r) = \{t \in r | Q(s \in (L \cap t.l))[t(A)s\Theta a]\}$$

Ein Beispiel für diese Operation ist $\sigma - IF_{(ABTEILUNG=AUTOS,\exists,[19,25])}(MAN)$ (siehe Abbildung 2.5). Diese Operation gibt alle Manager an, die irgendwann zwischen Zeit 19 und 25 für die Autoabteilung gearbeitet haben. Der $\exists$-Quantor gibt an, daß die Selektionsbedingung nur an irgendeiner Zeit im vorgegebenen Zeitraum erfüllt sein muß. Die Operation würde folgendes Ergebnis liefern:

MANAGER	MGEHALT	ABTEILUNG	Lebensspanne
RON	15 22K 22 23K	15 SPIELZEUG 19 AUTOS 25 PR	$\{[15, NOW]\}$
GARY	15 19K 19 20K 22 23K	15 PR 19 SPIELZEUG 22 AUTOS	$\{[15, NOW]\}$

Außerdem gibt es bei Clifford noch den *select-when* Operator. Der select-when Operator gibt Tupel zurück, deren Lebensspanne genau die Zeitpunkte sind, an denen das Selektionskriterium erfüllt wird.

$$\sigma - WHEN_{A\Theta a}(r) = \{t | \exists t' \in r] t.l = \{s | t'(A)(s)\Theta a\} \wedge t.v = t'.v|_{t.l}]\}$$

Zum Beispiel würde die Operation $\sigma - WHEN_{MANAGER=RON}(EMP)$ (siehe Abbildung 2.5)alle Stars zu den Zeitpunkten zeigen, an denen sie mit Wilder zusammenarbeiteten.

ENAME	GEHALT	MANAGER	Lebensspanne
TOM	15 14K	15 RON	$\{[15, 25]\}$

Time-Slice

Korrespondierend zu den unären Operatoren *select* und *project*, wurde der *time-slice* Operator von Clifford eingeführt, um eine historische Relation in der temporalen Dimension einzuschränken. Clifford führte zwei Arten von time-slice Operationen ein: den statischen und den dynamischen time-slice.

Der statische time-slice Operator τ_L schränkt die Ergebnistupel einfach auf die Lebensspanne L ein.

$$\tau_{@_L}(r) = \{t | \exists t' \in r[t.l = L \cap t'.l \wedge t.v = t'.v|_{t.l}]\}$$

Die Operation $\tau_{@_{[25,40]}}(EMP)$ (siehe Abbildung 2.5) würde folgendes Ergebnis liefern:

ENAME	GEHALT	MANAGER	Lebensspanne
TOM	25 18K	25 AL	$\{[25, 40]\}$

Der dynamische time-slice

Der dynamische time-slice nutzt die Unterscheidung zwischen Domänen in TD (Abbildungen von T in D) und Domänen in TT (Abbildungen von T in T). Wenn $DOM(A) \subseteq TT$, dann ist für jedes Tupel in einer Relation, die über A definiert ist, das *Image* von $t(A)$ die Menge der Zeiten in die sich $t(A)$ abbildet. Diese Menge von Zeiten wird benutzt, um den dynamischen time-slice zu definieren, der die Lebensspannen der Tupel in der Operandenrelation begrenzt.

$$\tau_{@A}(A) = \{t | \exists t' \in r[forL, \text{the image of } t'(A), t.l = L \cap t'.l \wedge t = t'|_{t.l}]\}$$

When

Der unäre Operator *when* bildet Relationen auf Lebensspannen ab.

$$\Omega(r) = LS(r)$$

Der *when* Operator gibt die Lebensdauer der Relation zurück. Der *when* Operator dient hauptsächlich als Parameter für andere Operationen.

Der Θ-Join Operator

Der Θ-Join Operator spielt hier die gleiche Rolle wie im Relationenmodell. Er verknüpft zwei Tupel der Parameterrelationen in der Ergebnisrelation, wenn sie in einer Θ-Beziehung zueinander stehen. Clifford hat den Operator für die Anwendung auf historische Relationen im HRDM modifiziert.

Seien $R_1 =< A_1, K_1, ALS_1, DOM_1 >$
und $R_2 =< A_2, K_2, ALS_2, DOM_2 >$ die Operanden. Die Ergebnisrelation $R_3 =< A_1 \cup A_2, K_1 \cup K_2, ALS_1 \cup ALS_2, DOM_1 \cup DOM_2 >$ ist dann definiert als

$$\begin{aligned} r1[A\Theta B]r2 \quad = \quad & \{t | \exists t_{r1} \in r1, \exists t_{r2} \in r2 \\ & [t.l = \{s | t_{r1}(A)(s) \Theta t_{r2}(B)(s)\} \wedge \\ & t.v(R1) = t_{r1}.v(R1)|_{t.l} \wedge \\ & t.v(R2) = t_{r1}.v(R2)|_{t.l}]\} \end{aligned}$$

wobei $A \subseteq A_1$ und $B \subseteq A_2$
Die Lebensspanne der Ergebnistupel ist nach dieser Definition jede einzelne Zeit, zu der die beiden Attributmengen A und B aus $R1$ und $R2$ in der Relation Θ zueinander stehen.

Der Natural-Join Operator

Der *Natural-Join* ist ein Spezialfall des Θ-Join mit $A = B = A_1 \cap A_2$. Die Attributmenge $A_1 \cap A_2$ soll hier mit X bezeichnet werden. Der Natural-Join $r1 \bowtie r2$ zweier Relationen ist definiert als

$$\begin{aligned} r1 \bowtie r2 \quad = \quad & \{t | \exists t_{r1} \in r1, \exists t_{r2} \in r2 \\ & [t.l = vls(X, t_{r1}, R1) \cap vls(X, t_{r2}, R2) \wedge \\ & t.v(R1) = t_{r1}.v(R1)|_{t.l} \wedge \\ & t.v(R2) = t_{r1}.v(R2)|_{t.l}]\} \end{aligned}$$

Der Time-Join Operator

Der *Time-Join* wird von Clifford definiert, um einen Join zwischen einer Relation und einem zeitwertigen Attribut A, mit $DOM(A) \subset TT$, auszuführen. Sei $r1$ eine Relation über Schema $R1$ und $r2$ eine Relation über $R2$ und A ein Attribut von $r1$ mit $DOM(A) \subset TT$. Der Time-Join von $r1$ und $r2$ über Attribut A von $r1$, geschrieben als $r1[@A]r2$, ist folgendermaßen definiert:

$$
\begin{aligned}
r1[@A]r2 \quad = \quad & \{t | \exists t_1 \in r1 \exists t_2 \in r2 \\
& [t.l = \{t_1.v(A)(s) | s \in t_1.l\} \cap t_1.l \cap t_2.l \wedge \\
& t.v(R1) = t_1.v(R1)|_{t.l} \wedge \\
& t.v(R2) = t_2.v(R2)|_{t.l}]\}
\end{aligned}
$$

2.4.3 Beispielanfragen

1. Wie hoch ist das Gehalt von Tom zur Zeit 19?
   $\Pi_{GEHALT}(\sigma - IF_{(ENAME=TOM,\forall,19)}(EMP))$

2. Wie sind die Namen der Angestellten, die mehr Geld als ihre Manager verdienen oder verdienten? Wann war das, und wie hoch waren die Löhne von Angestellten und Managern?
   $\Pi_{ENAME,GEHALT,MGEHALT}(\sigma - IF_{(GEHALT>MGEHALT,\exists,T)}(EMP \bowtie MAN))$

3. Wie sind die Namen und die Gehaltsentwicklung von Toms Managern?
   $\Pi_{M.MANAGER,MGEHALT}(\sigma - IF_{(ENAME=TOM,\forall,T)}(EMP \bowtie M.MAN))$

4. Wie sind die Namen der Angestellten, die 18K verdienten oder in der Zeit $[18, 30)$ von Al gemanaged wurden?
   $\Pi_{ENAME}(\sigma - IF_{(GEHALT>18K \vee MANAGER=AL,\exists,[18,30))}(EMP))$

Die Algebra für HRDM ist allerdings nicht vollständig. Eine Anfrage nach dem Gehalt aller Angestellten, die irgendwann den Manager gewechselt haben, kann nicht gestellt werden.

2.5 Tansel

2.5.1 Datenmodell

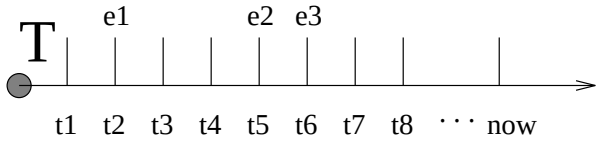
Tansel führte im Jahr 1986 ein Datenmodell mit Attributtimestamping ein [46]. Eine besonderheit daran ist die Vielzahl der erlaubten Attribute.

Tansel unterscheidet in seinem Modell vier Arten von Attributen (mengenwertige, atomare, mengenwertige und zeitvariante, atomare und zeitvariante). Die Algebra benötigt dafür natürlich zusätzliche Operationen, um zwischen den einzelnen Darstellungen der Attribute umzuschalten. Ein Überblick soll im Folgenden gegeben werden.

Zeit

Zeit ist in Tansels Modell eine Menge aufeinanderfolgender äquidistanter Punkte. Tansel gebraucht den Ausdruck Zeitpunkt, aber heute hat sich für eine nicht zerlegbare Zeiteinheit im diskreten Zeitmodell die Bezeichnung *Chronon* durchgesetzt.
T ist die Menge von Chronons, über die eine totale Ordnung $\leq$ definiert ist. Die Chronons werden relativ zum Ursprung t_0 identifiziert (siehe Abbildung 2.6).

Abbildung 2.6: Zeitachse



Außerdem definiert Tansel Zeitintervalle $[l, u)$ als Menge aufeinanderfolgender Chronons zwischen l und u.

$$[l, u) = \{t | t \in T \wedge l \leq t \leq u\}$$

Das Intervall ist also an seiner unteren Grenze geschlossen und an seiner oberen Grenze offen.

Ereignisse

Ereignisse sind bei Tansel die Transaktionen, Operationen oder Aktionen, die den Attributen der Objekte in der Datenbank einen neuen Wert zuweisen.
In Abbildung 2.6 seien $e1 \ldots e3$ Ereignisse, die Attributwerte a_1, a_2 und

a_3 des Attributs A eines Objektes o zuweisen. Der Gültigkeitszeitraum des Attributwertes a_1 ist dann $[t2, t5)$. Ähnliches ist aus der Abbildung für die Attributwerte a_2 und a_3 zu entnehmen.
Jeder Attributwert wird als geordnetes Paar <Zeit,Wert> aufgezeichnet. Die Zeit wird dabei durch ein Zeitintervall repräsentiert. Jedes Intervall, das als Obergrenze *now* enthält, ist ein expandierendes Intervall, da *now* mit fortschreitender Zeit immer größer wird. Wenn ein Ereignis zur Zeit l_e den Attributwert verändert, wird das Intervall $[l, now]$ geteilt in $[l, l_e)$ und $[l_e, now]$.
Wenn ein Objekt o in die Datenbank aufgenommen wird, erhält jedes Attribut als untere Intervallgrenze im Zeitstempel die Zeit t_b, zu der das Objekt erzeugt wurde. Die Attribute haben initial einen Nullwert oder einen Wert ungleich Null über dem Zeitintervall $[t_b, now]$.

Relationen

Sei U die Menge aller als atomar angenommenen Werte, wie Ganzzahlen, Realzahlen und Zeichenketten. Seien $D_{a_1}, \ldots, D_{a_m}$ Teilmengen von U und $D_{s_1}, \ldots, D_{s_m}$ Teilmengen von $P(U)$, wobei $P(U)$ die Potenzmenge von U ist. T_I ist die Menge der Intervalle über der Menge von Chronons T. Das heißt, $T_I \subseteq T \times T$, wobei $\times$ das kartesische Produkt bezeichnet.

$$T_I = \{[l, u) | l < u \wedge t_0 \leq l < now \wedge t_0 < u \leq now \wedge < l, u > \in T \times T\}$$

$D_{t_1}, \ldots, D_{t_m}$ seien Teilmengen von $T \times U$ und $P(D_{t_1}), \ldots, P(D_{t_m})$ die dazugehörigen Potenzmengen.
Es sei eine Anzahl von Mengen $E_1, E_2, \ldots, E_n$ angenommen, wobei E_i eine der oben definierten $D_{t_1}, \ldots, D_{t_m}$, $D_{a_1}, \ldots, D_{a_m}$, $D_{s_1}, \ldots, D_{s_m}$, $P(D_{t_1}), \ldots, P(D_{t_m})$, für $i = 1, \ldots, n$. Eine historische Relation (HR) ist eine Teilmenge des kartesischen Produkts $E_1 \times E_2 \times \ldots \times E_n$.
Wie man aus der Definition sieht, sind diese Relationen nicht in der ersten Normalform. Die Schachtelungstiefe ist höchstens Eins. Die Attributwerte können nur Mengen sein; nicht Mengen von Mengen. Zeitabhängige Attribute sind Funktionen von T_I in U.
$R(A_1, \ldots, A_n)$ ist ein historisches Relationenschema. n wird als der Grad

von R bezeichnet. $Atr(R)$ ist die Menge der Attribute von R. r ist eine Instanz des Relationenschemas R.
Eine historische Relation kann vier verschiedene Attributtypen haben. Atomare Attribute enthalten Werte aus der Menge U. zeitabhängige Attribute enthalten ein Zeitintervall und einen atomaren Wert. Außerdem gibt es Attribute, die Mengen atomarer Werte als Attributwerte haben und Attribute, die Mengen Zeitabhängiger Werte als Attributwert besitzen. Im letzteren Fall ist der Attributwert eine Anzahl von Triplets <Zeitintervall,Wert> über dem Intervall $[t_b, now]$. Die Intervallkomponente jedes Wertes ist eine Teilmenge von $[t_b, now]$. In Abbildung 2.7 ist ein Beispiel für eine historische Relation angegeben. Das Attribut *ENAME* fällt in die erste Kategorie der atomaren Attribute. Die Attribute *GEHALT* und *MANAGER* hingegen sind Mengen von Zeitabhängigen Attributwerten zugeordnet.

Abbildung 2.7: Beispiel einer historischen Relation nach Tansel

ENAME	GEHALT	MANAGER
TOM	$\{< [15,25), 14K >,$ $< [25, now], 18K >\}$	$\{< [15,19), RON >,$ $< [19,22, GARY >,$ $< [22, now], AL >\}$
ANN	$\{< [20,40), 12K\}$	$\{< [20,40), LIZ >\}$

Tansel beizeichnet Attribute mit A, wenn sie atomare Werte enthalten, mit $*A$, wenn ihnen Mengen atomarer Werte zugeordnet sind, mit $\bar{A}$, wenn es sich um ein zeitabhängiges Attribut handelt und schließlich mit $*\bar{A}$ im vierten Fall. Mit $\bar{A}_l$, $\bar{A}_u$ und $\bar{A}_v$ werden untere und obere Grenze des Zeitintervalls sowie der Wert eines zeitabhängigen Attributs bezeichnet.
Weiterhin kann man Instanzen von Relationen über einem bestimmten Zeitintervall definieren. Sei s ein Tupel der Relation r, dann ist $s[A]_{[l,u)}$ die Tupelkomponente, die zur Attributmenge A korrespondiert. Wenn A ein atomares Attribut ist, dann ist $s[A]_{[l,u)}$ gleichbedeutend mit $s[A]$. Desglei-

chen entsprechen $s[*A]_{[l,u)}$ und $s[*A]$. $s[\bar{A}]_{[l,u)}$ ist dem Tripel $< [l', u') >$ äquivalent, wobei $[l', u') = [l, u) \cap [l'', u'')$ und $[l'', u'')$ das Zeitintervall von $s[\bar{A}]$. Für mengenwertige, zeitabhängige Attribute gilt:

$$s[*\bar{A}]_{[l,u)} = \{< [l', u'), a > | x \in s[*\bar{A}] \wedge [l', u') = [l, u) \cap [x_l, x_u) \neq \phi \wedge a = x_v\}$$

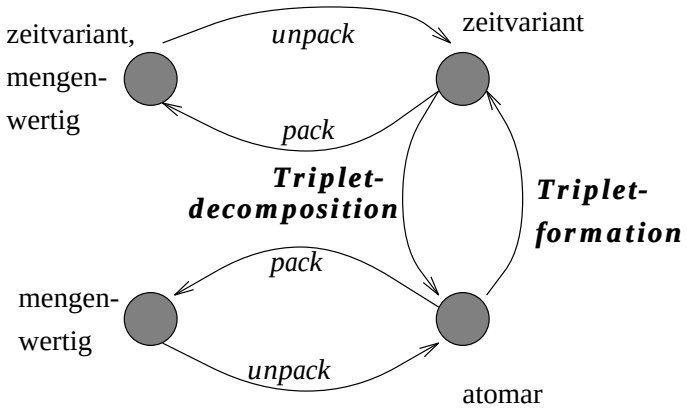
Eine Relationeninstanz über einem Intervall $[l, u)$ ist definiert als

$$r_{[l,u)} = \{s_{[l,u)} | s \in r\}.$$

2.5.2 Erweiterungen in der Algebra

Die relationale Algebra für Tansels Modell enthält einige Standardoperationen, die unverändert vom Relationenmodell übernommen wurden, wie Projektion, kartesisches Produkt. Die Mengenoperationen Vereinigung und Differenz wurden nur geringfügig verändert, um überlappende, adjazente und enthaltene Zeitintervalle zu behandeln.
Die Selektion wurde ebenfalls, entsprechend den verschiedenen Attributtypen, leicht modifiziert. Ähnliches gilt für den Join-Operator.
Abbildung 2.8 zeigt Operationen, die dem Datenmodell entsprechend eingeführt beziehungsweise geändert wurden.

Abbildung 2.8: Neue und erweiterte algebraische Operationen



Pack-Operation (P)

Die *pack*-Operation bewirkt, auf ein Attribut A angewendet, die Zusammenfassung der Werte von A in eine einzelne Tupelkomponente, für Tupel, deren restliche Attribute übereinstimmen [16]. Sei R eine Relation des Grades n und A eines seiner Attribute. Für jedes $(n-1)$-Tupel in $\pi_{C_A}(R)$ ist ein n-Tupel W_g wie folgt definiert:

$$W_g[C_A] = g$$

$$W_g[A] = \begin{cases} \{t[A] | t \in R \wedge t[C_A] = g\} \\ \text{wenn A ein atomares oder} \\ \text{einfach zeitabhängiges Attribut ist} \\ \\ \{x | \exists t[t \in R \wedge t[C_A] = g \wedge x \in t[A]]\} \\ \text{wenn A ein mengenwertiges oder} \\ \text{ein mengenwertiges und zeitabhängiges} \\ \text{Attribut ist} \end{cases}$$

Dann ist $P_A(R)$ definiert als

$$P_A(R) = \{W_g | g \in R[C_A]\}$$

Man sieht, daß die Schachtelungstiefe immer eins bleibt, da die *pack*-Operation beim Packen von mengenwertigen Attributen die Vereinigung der Mengen durchführt, anstatt ein neues Level der Verschachtelung zu erzeugen. Abbildungen 2.9 und 2.10 zeigen ein Beispiel für die Anwendung der *pack*-Operation.

Unpack-Operation(U)

Wird die *unpack*-Operation auf ein mengenwertiges Attribut der Relation R angewendet, wird ein Tupel für jedes Element der Menge im Attribut erzeugt. Es ist also die Umkehroperation von *pack*[16].

Abbildung 2.9: Die Relation EMP

ENAME	GEHALT	MANAGER
TOM	$< [15, 25), 14K >$	$\{< [15, 19), RON >,$ $< [19, 22), GARY >,$ $< [22, now], AL >\}$
TOM	$< [25, now], 18K >$	$\{< [15, 19), RON >,$ $< [19, 22), GARY >,$ $< [22, now], AL >\}$
ANN	$\{< [20, 40), 12K\}$	$\{< [20, 40), LIZ >\}$

Abbildung 2.10: $P_{GEHALT}(EMP)$

ENAME	GEHALT	MANAGER
TOM	$\{< [15, 25), 14K >,$ $< [25, now], 18K >\}$	$\{< [15, 19), RON >,$ $< [19, 22), GARY >,$ $< [22, now], AL >\}$
ANN	$\{< [20, 40), 12K\}$	$\{< [20, 40), LIZ >\}$

Sei t ein Tupel der Relation R.

$$W_A(t) = \begin{cases} \{t\} \\ \text{wenn A ein atomares oder} \\ \text{einfach zeitabhängiges Attribut ist} \\ \\ \{t' | t'[A] \wedge t'[C_A] = t[C_A]\} \\ \text{wenn A ein mengenwertiges oder} \\ \text{ein mengenwertiges und zeitabhängiges} \\ \text{Attribut ist} \end{cases}$$

Dann ist $U_A(R)$ definiert als

$$U_A(R) = \bigcup_{t \in R} U_A(\{t\})$$

Würde die *unpack*-Operation zum Beispiel auf das Attribut GEHALT der Relation in Abbildung 2.10 angewendet, würde die ursprüngliche Relation in Abbildung 2.9 wieder hergestellt.
Die sukzessive Anwendung der *unpack*-Operation auf alle Attribute einer Relation R konvertiert R in die erste Normalform. Wenn man ein Tupel nach dem Attribut $*\bar{A}$ „entpackt", entsteht eine Familie von Tupeln, die sich nur im Attributwert von A unterscheiden. Die Anwendung des *unpack*-Operators bewirkt die Erzeugung von redundanten Daten, da andere mengenwertige Attributwerte vervielfältigt werden (siehe Beispiel in Abbildung 2.9 und 2.10). Wird die Operation aufeinanderfolgend auf mehrere Attribute eine Relation angewendet, können sehr große Relationen entstehen. Um dem entgegenzuwirken, kann der *SLICE*-Operator verwendet werden, dessen Definition später noch vorgestellt wird.

Triplet-decomposition und Triplet-formation

Die Operation *triplet-decomposition* zerlegt das Tripel eines zeitabhängigen Attributs (obere Intervallgrenze, untere Intervallgrenze und Attributwert) in seine Einzelteile. Dafür werden der Relation zwei neue Attribute hinzugefügt, die die untere und die obere Intervallgrenze repräsentieren.

Der reine Attributwert ersetzt das Tripel des ursprünglichen zeitabhängigen Attributs.
Die Operation *triplet-decomposition* wird benötigt, um die traditionellen *Selektions-* und *Projektionsanweisungen* auf die einzelnen Komponenten von zeitabhängigen Attributen anwenden zu können.
Die Umkehroperation zur *triplet-decomposition* ist *triplet-formation*. Sie erzeugt aus einer durch oben beschriebene Operation erhaltene Relation wieder die Ausgangsrelation.

Slice-Operation(SLICE)

Der *SLICE*-Operator zerteilt Tripel (<Zeit,Wert>) eines Attributs entsprechend der Zeit eines anderen Attributs.
Sei R eine Relation und $\bar{A}$, $\bar{B}$ zwei zeitabhängige Attribute.

$$\begin{aligned} SLICE_{\bar{A},\bar{B}}(R) \quad = \quad & \{t | \exists t'[t' \in R \wedge t[C_{\bar{A}}] = t'[C_{\bar{A}}] \wedge \\ & t[\bar{A}_v] = t'[\bar{A}_v] \wedge t[\bar{A}_l] \geq t'[\bar{B}_l] \wedge \\ & t[\bar{A}_u] \leq t'[\bar{B}_u] \wedge t[\bar{A}_l] \geq t'[\bar{A}_l] \wedge \\ & t[\bar{A}_u] \leq t'[\bar{A}_u] \wedge \\ & (t[\bar{A}_l] = t'[\bar{A}_l] \vee t[\bar{A}_l] = t'[\bar{B}_l]) \wedge \\ & (t[\bar{A}_u] = t'[\bar{A}_u] \vee t[\bar{A}_u] = t'[\bar{B}_u])]\} \end{aligned}$$

Seien $\bar{A}$ und $\bar{B}$ Attribute des Relationenschemas R und t ein Tupel über R. Der Durchschnitt der Zeitintervalle von $t[\bar{A}]$ und $t[\bar{B}]$ wird der Intervallkomponente des neuen Wertes von $\bar{A}$ zugewiesen.
In ähnlicher Weise definiert Tansel noch zwei weitere Slice-Operationen, die dem Ergebnis die Vereinigung beziehungsweise die Differenz der Zeitintervalle der Ausgangstupel zuweisen.
Wie oben schon erwähnt, können die Slice-Operatoren vorteilhaft eingesetzt werden, um die Datenredundanz bei der Verwendung der *Unpack*-Operation zu verhindern. Andere Attribute können entsprechend der Zeit des „auszupackenden“ Tupels „gesliced“ werden.

Drop-time Operation (DROP-TIME)

Als letzte neue Operation wird von Tansel *DROP-TIME* eingeführt. Diese Operation bewirkt den einfachen Wegfall der Zeitkomponente eines oder mehrerer Attribute.
Sei R eine Relation und $A \in Atr(R)$:

$$DROP\text{-}TIME_A(R) = \begin{cases} \{t | \exists t'[t' \in R \wedge t[C_{bar A}] \\ = t'[C_{bar A}] \wedge t[A] = t[C_{bar A_v}]]\} \\ \text{wenn A ein einfach zeitabhängiges Attribut ist} \\ \\ \{t | \exists t'[t' \in R \wedge t[C_{bar A}] \\ = t'[C_{bar A}] \wedge x \in t'[*\bar{A}] \wedge x_v \in t[A]]]\} \\ \text{wenn A ein mengenwertiges und zeitabhängiges} \\ \text{Attribut ist} \\ \\ R \text{ sonst} \end{cases}$$

Die Operation ist einfach eine Abkürzung, um Relationen zu bearbeiten, ohne die Zeit zu berücksichtigen.

2.5.3 Beispielanfragen

1. Wie hoch ist das Gehalt von Tom zur Zeit 19?

   $\Pi_{GEHALT}(T\text{-}DEC_{GEHALT}(\sigma_{ENAME=TOM \wedge GEHALT_l \leq 19 \wedge GEHALT_u > 19}(U_{GEHALT}(EMP))))$

2. Wie sind die Namen der Angestellten, die mehr Geld als ihre Manager verdienen oder verdienten? Wann war das, und wie hoch waren die Löhne von Angestellten und Managern?
   $\Pi_{ENAME,GEHALT,MGEHALT}(\sigma(GEHALT_v > MGEHALT_v$
   $(SLICE_{MGEHALT,E.MANAGER}(U_{GEHALT}(M.MAN)$
   $M.MANAGER = \bowtie_{MANAGER_v} (SLICE_{M.MANAGER,GEHALT}(SLICE_{GEHALT_M.MANAGER}$
   $(U_{M.MANAGER}(U_{GEHALT}(EMP)))))))$

3. Wie sind die Namen und die Gehaltsentwicklung von Toms Managern?
   $\Pi_{M.MANAGER,MGEHALT}((\sigma ENAME = TOM(U_{M.MANAGER}(M.MAN))$
   $M.MANAGER =\bowtie_{MANAGER} EMP)$

4. Wie sind die Namen der Angestellten, die 18K verdienten oder in der Zeit $[18, 30)$ von Al gemanaged wurden?
   $\Pi_{ENAME}(\sigma_{GEHALT_t \supseteq [18,30)}$
   $(USLICE_{GEHALT,MANAGER}(\sigma_{GEHALT_v > 18K \vee MANAGER=AL}$
   $(U_{GEHALT}(U_{MANAGER}(EMP))))))$

Die Anfragen muten sehr kompliziert an. Das kommt daher, daß Tansel sein Modell möglichst nahe an der realen Welt und nicht so sehr am Relationenmodell halten wollte.

2.6 IXRM von Lorentzos

Nikos A. Lorentzos stellte in [29] ein Datenmodell vor, das sich von anderen temporalen Modellen vor Allem dadurch unterscheidet, daß die Unterstützung von Gültigkeitszeit und Transaktionszeit von Objekten nicht unbedingt der Primäre Aspekt, sondern eher eine Eigenschaft des Modells ist. Nichts desto trotz wurde das Modell im Rahmen der Forschung für temporale Datenbanken entwickelt.
Lorentzos nennt sein Modell intervallerweitertes Relationenmodell (IXRM). Die Idee ist, Intervalle aller Art zu unterstützen, sowohl im Modell, als auch in den algebraischen Operationen. Neben Zeitintervallen werden auch Intervalle in anderen Domänen möglich.

2.6.1 Das Datenmodell

Eine Intervallrelation wird bei Lorentzos wie folgt definiert:
Wenn $D_1, D_2, \ldots, D_n$ eindimensionale Räume sind, dann ist eine Intervallrelation eine Teilmenge von $X(D_1) \times X(D_2) \times \ldots \times X(D_n)$.
Als eindimensionalen Raum bezeichnet Lorentzos dabei eine endliche Menge mit einer totalen Ordnung „$<$“. $I(D)$ bezeichnet die Menge al-

ler möglichen Intervalle über einem solchen Raum. Mit $X(D)$ notiert Lorentzos die Abkürzung für $I(D)$ oder D. Zum Beispiel könnte man eine Relation EMP(ENR=INT, ENAME=ALPHA-15, GEHALT=INT, ZEIT=I(MONTH)) definieren.
Ein Beispiel für eine Minidatenbank ist in der Abbildung 2.11 angegeben.

2.6.2 Die Algebra

Die Mengenoperationen Vereinigung, Differenz, Durchschnitt und kartesisches Produkt bleiben für Intervallrelationen unverändert.
Die *Selektionsoperation* wurde nur geringfügig verändert.

$$\sigma_F(R) = \{t | t \in R \wedge F(t) = True\}$$

F ist dabei die Formel, die die Selektionsbedingung angibt. Im Unterschied zum Relationenmodell sind als Operanden in F auch Intervalle erlaubt und zusätzlich noch spezielle Operationen zum Vergleich von Intervallen. Die Abbildung 2.12 zeigt eine Reihe möglicher Lagen von Intervallen zueinander. Lorentzos schlägt vor, bei der Implementierung seines Modells für jede dieser Lagen einen Operator einzuführen. Zum Beispiel ergibt die Operation $\sigma_{(MANAGER=RON \wedge VALID=superinterval([14,@)))}(MAN)$ (siehe Abbildung 2.11) folgende Relation:

MANAGER	ABTEILUNG	VALID
LIZ	WASCHMASCHINEN	[10, @)

Außerdem führt Lorentzos die *Fold*-Operation ein. Wird eine n-fache Relation R über Attribut A_i „gefaltet“, werden alle Tupel, deren A_j Komponenten identisch sind $(\forall j \neq i)$ und deren A_i Komponenten zusammengelegt werden können, zu einem einzigen Tupel in der Ergebnisrelation zusammengefaßt.
Die Umkehroperation zu Fold ist *Unfold*. Wenn eine n-fache Relation R über Attribut A_i „entfaltet“ wird, wird jedes Tupel $(t_1, \ldots, t_i, \ldots, t_n)$ aus R durch eine Familie von Tupeln $(t_1, \ldots, t_{ij}, \ldots, t_n)$ wobei t_{ij} ein Punkt aus t_i ist. Die Unfold-Operation zeigt, warum Lorentzos eindimensionale Räume im IXRM-Modell als endliche Mengen definiert hat. Wäre das nicht der Fall, könnte man zum Beispiel in einem Attribut A einen Wert

Abbildung 2.11: IXRM Beispiel

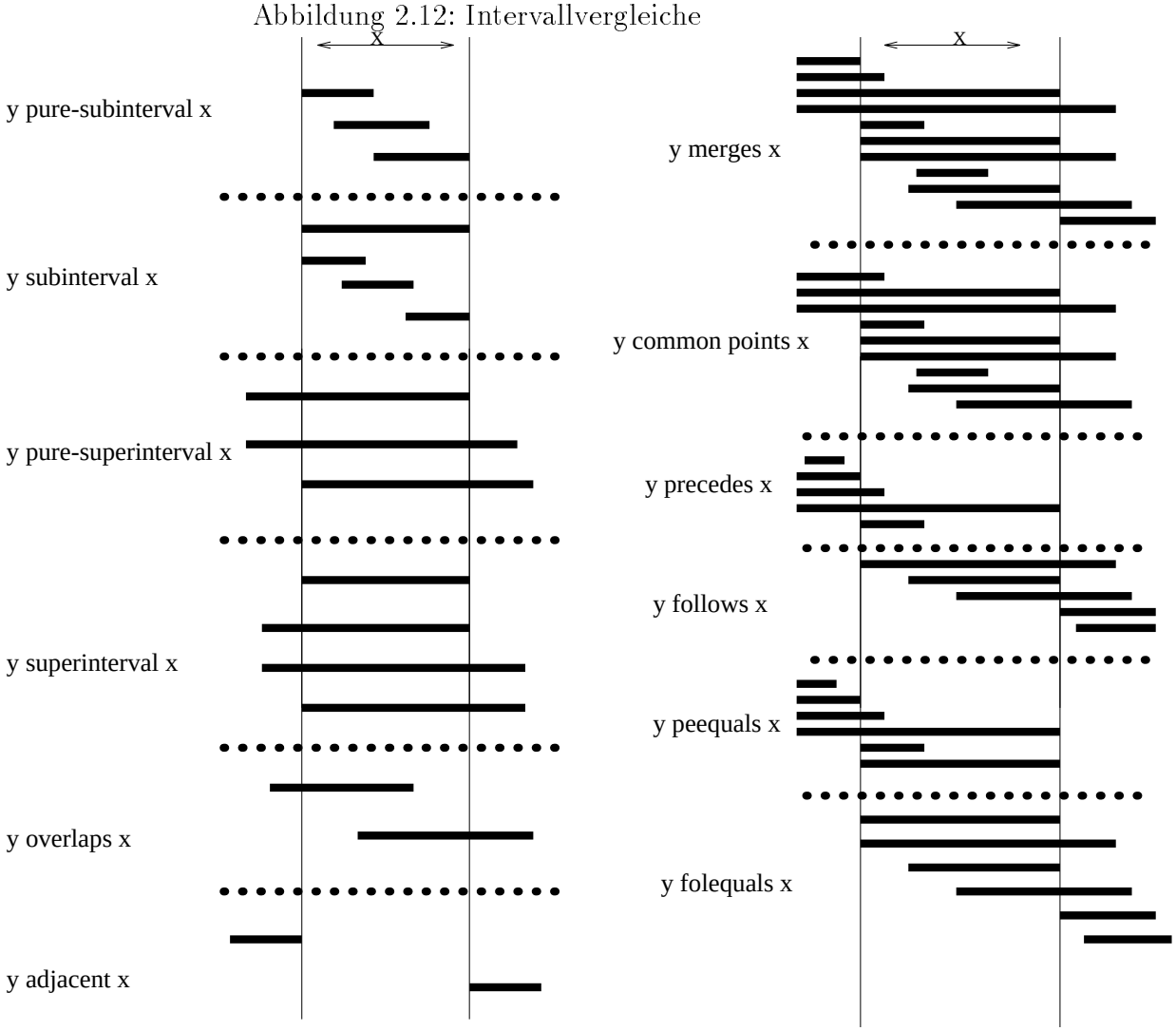
ENAME	GEHALT	VALID
TOM	14K	$[15, 25)$
TOM	18K	$[25, @)$
ANN	12K	$[10, @)$

ENAME	MANAGER	VALID
TOM	RON	$[15, 19)$
TOM	GARY	$[19, 22)$
TOM	AL	$[22, @)$
ANN	LIZ	$[10, @)$

MANAGER	GEHALT	VALID
RON	22K	$[15, 22)$
RON	23K	$[22, @)$
GARY	19K	$[15, 19)$
GARY	20K	$[19, 22)$
GARY	23K	$[22, @)$
AL	22K	$[22, @)$
LIZ	15K	$[10, @)$

MANAGER	ABTEILUNG	VALID
RON	SPIELZEUG	$[15, 19)$
RON	AUTOS	$[19, 25)$
RON	PR	$[25, @)$
GARY	PR	$[15, 19)$
GARY	SPIELZEUG	$[19, 22)$
GARY	AUTOS	$[22, @)$
AL	SPIELZEUG	$[22, @)$
LIZ	WASCHMASCHINEN	$[10, @)$

Abbildung 2.12: Intervallvergleiche



wie $[3, un)$ aufzeichnen. Eine Relation $S = Unfold : A!(R)$ hätte dann unendlich viele Tupel.
Es werden noch zwei weitere Operationen *P-Union* und *P-Diff* definiert. Diese Operationen können auch mit Hilfe von Fold und Unfold ausgedrückt werden, sind aber nützlich beim Management von Intervallrelationen.
$P - Union : A_1, A_2, \ldots, A_k!(R, S) =$
$Fold : A_1, A_2, \ldots, A_k!($
$Unfold : A_1, A_2, \ldots, A_k!(R) \cup Unfold : A_1, A_2, \ldots, A_k!(S))$
$P - Union : A_1, A_2, \ldots, A_k!(R, S) =$
$Fold : A_1, A_2, \ldots, A_k!($
$Unfold : A_1, A_2, \ldots, A_k!(R) - Unfold : A_1, A_2, \ldots, A_k!(S))$
Die *Join*-Operation wird einfach definiert als $Rx_F S = \sigma_F(R \times S)$.

2.6.3 Beispielanfragen

1. Wie hoch ist das Gehalt von Tom zur Zeit 19?
   $F := ENAME = TOM \wedge$ Time common-points 19
   $\Pi_{GEHALT}(\sigma_F(Unfold : Time!(EMP)))$

2. Wie sind die Namen der Angestellten, die mehr Geld als ihre Manager verdienen oder verdienten? Wann war das, und wie hoch waren die Löhne von Angestellten und Managern?
   $F := GEHALT > MGEHALT$
   $Fold : Time!\Pi_{ENAME,GEHALT,MGEHALT,Time}(\sigma_F(Unfold : Time!(E.EMP \bowtie_{E.Manager=M.Manager} M.MAN))$

3. Wie sind die Namen und die Gehaltsentwicklung von Toms Managern?
   $\Pi_{M.MANAGER,MGEHALT}(\sigma_{(ENAME=TOM)}(E.EMP \bowtie_{E.Manager=M.Manager} M.MAN))$

4. Wie sind die Namen der Angestellten, die 18K verdienten oder in der Zeit $[18, 30)$ von Al gemanaged wurden?
   $F := (GEHALT > 18K \vee MANAGER = AL) \wedge Timecommon -$

$points[18, 30)$
$\Pi_{ENAME}(\sigma_F(Unfold : Time!(EMP)))$

2.7 Eine Algebra für TQUEL von Snodgrass

In [38] stellt Richard Snodgrass eine Gültigkeitszeitalgebra vor, die er für seine Sprache TQUEL entwickelt hat. TQUEL soll eine minimale Erweiterung der Ingres Anfragesprache QUEL sein, die sowohl Transaktionszeit als auch Gültigkeitszeit unterstützt.

2.7.1 Das Datenmodell

Attribute unterliegen den gleichen Beschränkungen wie im Relationenmodell. Sie dürfen weder mengenwertig noch doppelt sein. Die Gültigkeitszeit wird durch mengenwertige Zeitstempel repräsentiert, die den Attributwerten zugeordnet werden.
Ein Relationenschema ist als eine endliche Menge von Attributen $N = \{n_1, n_2, \ldots, n_m\}$ definiert. Zu jedem Attribut $n \in N$ gehört eine Domäne $Dom(n)$, die eine beliebige, nicht leere, endliche oder zählbare Menge ist. Die positiven ganzen Zahlen bilden die Domäne T. Jedes Element von T repräsentiert ein Chronon. Die Menge $\mathrm{P}(T)$ sei die Potenzmenge von T. Ein Element von $\mathrm{P}(T)$ wird Gültigkeitszeitelement genannt.
Wenn $Wert$ über die Domäne $Dom(n_1) \cup Dom(n_2) \cup \ldots \cup Dom(n_m)$ reicht und $Gültigkeit$ über $\mathrm{P}(T)$, dann ist ein Gültigkeitstupel ht definiert als eine Abbildung der Attributnamen in die Menge geordneter Paare $(Wert, Gültigkeit)$ mit folgenden Einschränkungen:

- $\forall a, 1 \leq a \leq m | Wert(ht[n_a]) \in Dom(n_a)$ und
- $\exists a, 1 \leq a \leq m | Gültigkeit(ht[n_a]) \neq \emptyset$.

Es ist also möglich, daß alle Attribute, außer eines, einen leeren Zeitstempel haben.
Zwei Tupel ht und ht' heißen Wertäquivalent genau dann, wenn $\forall A \in N, Wert(ht[A]) = Wert(ht'[A])$.

Eine Gültigkeitszeitrelation h ist als eine endliche Menge von Gültigkeitstupeln definiert, von denen keine zwei Tupel wertäquivalent sind.

Abbildung 2.13: Beispielrelation

ENAME	GEHALT	MANAGER
TOM	$\{<[15,25),14K>,$ $<[25,now],18K>\}$	$\{<[15,19),RON>,$ $<[19,22),GARY>,$ $<[22,now],AL>\}$
ANN	$\{<[10,now),12K\}$	$\{<[10,now),LIZ>\}$

Abbildung 2.13 zeigt ein Beispiel für eine Gültigkeitszeitrelation.

2.7.2 Die Algebra

Snodgrass führt zunächst zwei Hilfsoperationen ein.

$$NotNull(R) = \{R | \exists r \in R \wedge \exists A \in N(Gültigkeit(R[A]) \neq \emptyset)\}$$

R ist eine Menge von Tupeln. Das Ergebnis dieser Operation enthält alle Tupel, die wenigstens ein Attribut enthalten, das einen nicht leeren Zeitstempel enthält.

$$\begin{aligned} Reduce(R) \quad = \quad & \{U^n | \forall A \in N \exists r \in R(r \equiv u \wedge \\ & \forall t \in Gültigkeit(u[A])(t \in Gültigkeit(r[A]))) \wedge \\ & \forall r \in R(r \equiv u \Rightarrow \forall A \in N(Gültigkeit(r[A]) \subseteq Gültigkeit(u[A])))\} \end{aligned}$$

Diese kompliziert anmutende Operation liefert die minimale Menge von Tupeln, die keine wertäquivalenten Tupel enthält.
Mit Hilfe dieser beiden Operationen lassen sich nun Mengenoperationen über Gültigkeitsrelationen definieren. Die Vereinigung zweier Gültigkeitsrelationen Q und R ist zum Beispiel die Menge aller Tupel, die in nur in Q, oder nur in R, oder in beiden Relationen sind. Jedes Paar wertäquivalenter Tupel wird durch ein einziges Tupel ersetzt. Der Zeitstempel dieser Tupel ist Vereinigung der korrespondierenden Attribute der wertäquiva-

lenten Tupel.

$$Q\hat{\cup}R = Reduce(\{u|u \in Q \vee u \in R\})$$

Die Gültigkeitsdifferenz von Q und R ist die Menge aller Wertäquivalenten Tupel in Q und R. Die Zeitstempel der Ergebnisrelation sind die Mengendifferenz der Zeitstempel der Ausgangstupel.

$$\begin{aligned} Q\hat{-}R \quad = \quad & \{q^m|\exists q \in Q \wedge \neg(\exists r \in R(r \equiv))] \\ & \bigcup NotNull(\{u^m|\exists q \in Q \exists r \in R(u \equiv q \equiv r \\ & \wedge \forall A \in N(valid(u[A]) = valid(q[A]) - valid(r[A])))\}) \end{aligned}$$

Das kartesische Produkt von zwei Gültigkeitszeitrelationen unterscheidet sich aufgrund des Attributtimestampings nicht von der Definition des kartesischen Produktes der Relationenalgebra.
Die Gültigkeitszeitselektion unterscheidet sich ebenfalls nicht vom Äquivalent in der Relationenalgebra. Es werden nur die Attributwerte und keine Zeitstempel betrachtet.
Sei R eine Gültigkeitsrelation mit m Tupeln und seien $a_1, \ldots, a_n$ verschiedene ganze Zahlen zwischen 1 und m. Die Gültigkeitsprojektion, $\hat{\Pi}_{N_{a_1},\ldots,N_{a_n}}$, ermittelt für jedes Tupel die Tupelkomponenten, die zu den Attributnamen $N_{a_1}, \ldots, N_{a_n}$ korrespondieren.

$$\hat{\Pi}_{N_{a_1},\ldots,N_{a_n}}(R) = Reduce(NotNull(\{u|\exists r \in R \forall i, 1 \leq i \leq n(u[i] = r[a_i])\}))$$

Neue temporale Operatoren

Als neuen temporalen Operator, der kein Äquivalent in der Relationenalgebra hat, führt Snodgrass den Ableitungsoperator ein. Der Ableitungsoperator, $\hat{\delta}_{G,V_1,\ldots,V_n}(R)$, bestimmt für ein Tupel $r \in R$ einen neuen Zeitstempel für die Attribute von r. Zuerst werden alle möglichen Intervalle, denen Attributnamen zugewiesen sind, bestimmt, für die das Prädikat G über den Zeitstempeln erfüllt ist. Für jedes dieser Intervalle berechnet der Operator V_a, mit $1 \leq a \leq n$. Die Menge der Zeiten, die aus der Berechnung von V_a resultiert, werden dann kombiniert und bilden den neuen Zeitstempel für das Attribut N_a. Der Ableitungsoperator erfüllt zwei Funktionen. Erstens führt er eine Selektion über der Gültigkeitskompo-

nente der Attribute eines Tupels aus. Zweitens wird ein neuer Zeitstempel für jedes Attribut N_a mit Hilfe der Formel V_a berechnet. Wenn $V_1, \ldots, V_n$ alle die gleiche Funktion sind, wird das Tupel quasi von Attributtimestamping zu Tupeltimestamping konvertiert. Zur Definition der Semantik der Ableitungsoperation benutzt Snodgrass die Hilfsfunktion *Apply*. Apply selektiert ein Intervall aus der Gültigkeitszeitkomponente des Zeitstempels jedes Attributs, wendet das Prädikat G darauf an, und wenn G erfüllt ist, wird V_i berechnet, um ein Ergebnisintervall zu generieren.

$$\begin{aligned} Apply(G, V_1, \ldots, V_n, R) \quad = \quad & \{u | \exists r \in R(u \equiv r \\ & \wedge \exists I_i \in interval(valid(r[i])), \\ & \ldots \\ & \wedge \exists I_i \in interval(valid(r[i])), \\ & (G(I_1, \ldots, I_m) \wedge valid(u[i]) = V_i(I_1, \ldots, I_m))\} \end{aligned}$$

Die Ableitungsoperation wird nun wie folgt definiert werden:

$$\hat{\delta}_{G,V_1,\ldots,V_m}(R) = NotNull(Reduce(Apply(G, V_1, \ldots, V_m, R)))$$

Ein weiterer Operator, $\hat{SN}$, berechnet eine normale Schnappschußrelation zu einer bestimmten Zeit τ. Ein Tupel wird nur selektiert, wenn alle Attribute zur Zeit τ gültig sind.

$$\hat{SN}_\tau(R) = \{(value(r[1], \ldots, r[n]))|R(r) \wedge \forall A \in N(\tau \in valid(r[A]))\}$$

Das Gegenstück ist der AT-Operator, der eine Schnappschußrelation in eine Gültigkeitszeitrelation transformiert, die als gültig zur spezifizierten Zeit τ betrachtet wird.

$$AT_\tau(R) = \{r | \exists r' \in R' \forall A \in N(r'[A] = value(r[A]) \wedge valid(r[A]) = \{\tau\})\}$$

Ein weiterer Operator ist der „Schnappschußtransaktionszeitoperator“. Der Operator ρ hat als Parameter den Namen einer Relation und eine Zeit. Er liefert den Schnappschuß einer Relation zur angegebenen Zeit. Der aktuelle Status der Relation EMP kann durch

$$\rho_{now}(EMP)$$

erfragt werden.

2.7.3 Beispielanfragen

1. Wie hoch ist das Gehalt von Tom zur Zeit 19?
   $\hat{\Pi}_{GEHALT}(\hat{\sigma}_{ENAME=TOM}(\rho_{19}(EMP)))$

2. Wie sind die Namen der Angestellten, die mehr Geld als ihre Manager verdienen oder verdienten? Wann war das, und wie hoch waren die Löhne von Angestellten und Managern?
   $\hat{\Pi}_{ENAME,GEHALT,MGEHALT}(\hat{\sigma}_{(GEHALT_v > MGEHALT_v)}$
   $\hat{\sigma}_{E.MANAGER=M.MANAGER}(E.EMP \hat{\times} M.MAN)))$

3. Wie sind die Namen und die Gehaltsentwicklung von Toms Managern?
   $\hat{\Pi}_{M.MANAGER,MGEHALT}((\sigma ENAME = TOM$
   $\hat{\sigma}_{E.MANAGER=M.MANAGER}(E.EMP \hat{\times} M.MAN)))$

Kapitel 3

Objektorientierte Modelle

Dieses Kapitel beschäftigt sich mit Implementation des Zeitaspektes in objektorientierte Datenmodelle und entsprechenden Algebren. Obwohl der weitaus größte Teil der Forschung auf dem Gebiet der temporalen Datenbanken auf die Erweiterung des Relationenmodells entfällt, gibt es auch eine Anzahl von Vorschlägen, die Gültigkeitszeit von Objekten der realen Welt in objektorientierten Modellen zu unterstützen. Seit dem Ende der 80er Jahre wurden einige objektorientierte Modelle und entsprechende Anfragesprachen vorgestellt, so zum Beispiel OSQL [6], eine auf SQL basierende Anfragesprache oder TOSQL [13].

3.1 Unterstützung des Zeitaspektes in objektorientierten Modellen

3.1.1 Direkte Nutzung des Objektmodells

Der erste Ansatz, die Gültigkeitszeit von Objekten zu unterstützen, nutzt die Ausdruckskraft des objektorientierten Datenmodells direkt. Daher werden keine Änderungen im Datenmodell oder der Anfragesprache notwendig. Das OODAPLEX System, das parametrisierte Typen wie Menge, Multimenge, Tupel und Funktion unterstützt, ist ausreichend, um temporale Informationen zu modellieren. Zum Beispiel kann Attributtimestam-

ping genutzt werden, um zu spezifizieren, daß *GEHALT* und *MANAGER* zeitvariant sind.

```
type employee is object
        function ENAME(e:employee->n:string)
        function GEHALT(e:employee->f:(t:time->m:gehalt))
        function MANAGER(e:employee->f:(t:time->n:manager))
```

Das Attribut ENAME ist eine Funktion, die eine bestimmte Instanz des Objektes *employee* auf eine bestimmten String abbildet. Die Funktion GEHALT gibt, auf eine bestimmte Instanz von *employee* angewendet, eine Funktion zurück, die bei Angabe einer Zeit das dazugehörige Gehalt zurück gibt. Analog wurde das Attribut MANAGER definiert. Andere Funktionen werden durch das System bereitgestellt. Die Funktion *extent* gibt die Instanzen eines bestimmten Objektes zurück. Die folgende Anfrage gibt alle Gehälter aus, die Tom hatte, während Al sein Manager war.

```
for the e in extend(employee) where ENAME(e) = ´TOM´
        for each t where MANAGER(e)(t) = ´AL´
                GEHALT(e)(t)
        end
end
```

Als Alternative könnte Objekttimestamping [23] verwendet werden. Dazu wird eine Funktion *state* definiert, die die Zeit auf ein nicht temporales Objekt abbildet.

```
type employee is object
        function ENAME(e:employee->n:string)
        function sate(e:employee->f:(t:time->d:snapshot))
type snapshot is object
        function GEHALT(e:employee->gehalt)
        function MANAGER(e:employee->manager)
```

Die Vorteile dieser beiden Ansätze sind, daß der Anwender unterschiedliche Semantik der Gültigkeitszeit selbst spezifizieren kann und daß die Anfragesprache unverändert bleibt. Auf der anderen Seite muß der Anwender die Unterstützung der Zeit vollständig selbst handhaben. Außerdem gibt es Probleme bei der Optimierung, da der Optimierer keinen Hinweis auf zeitvariante Daten erhält.

3.1.2 Andere Erweiterungen, um Zeit zu unterstützen

Ein anderer Ansatz als der oben beschriebene ist die Erweiterung des Datenmodells und der Anfragesprache um Möglichkeiten, den Zeitaspekt von Objekten zu modellieren.
Sciore schlägt die Verwendung von *Annotationen* vor (wurden in [34] eingeführt), um verschiedene Arten der Versionierung, historische Daten, Revisionen und Alternativen zu unterstützen [14]. Die letzten beiden Aspekte betreffen nur die Transaktionszeit. Sciore hat auch das EXTRA Datenmodell [32] um das Schlüsselwort *versioned* erweitert, um zeitvariante Attribute von anderen zu unterscheiden [15]. Sciore erweiterte die Anfragesprache EXCESS, um den Iterator *inall* einzuführen. Eine Anfrage, die im orginalen EXCESS ausgedrückt wird, operiert nur über der Standardversion des Objektes. Eine Anfrage mit *inall* betrachtet alle Versionen des Objektes. Außerdem können Prädikate in der where-Klausel die gewünschten Versionen auswählen.
Sciores Vorschlag unterstützt Annotationen durch die Zuweisung einer versteckten Variable für jede konventionelle Variable. Jede versteckte Variable ist an ein Objekt einer Annotationsklasse gebunden, die eine konventionelle Klasse ist. Das Objekt *employee* kann folgendermaßen ausgedrückt werden.

```
class employee inherits object=
        variables
                ENAME:string;
                GEHALT:HistoryFn, gehalt;
```

```
                    MANAGER:HistoryFn, manager;
    end employee;
```

Die Variable ENAME ist nicht annotiert. Die Variablen GEHALT und MANAGER sind annotiert mit der Annotationsklasse HistoryFn, die Versionierung unterstützt. Normale Annotationen sind für den Anwender nicht sichtbar. Wenn auf die Variable GEHALT zugegriffen wird, wird eine Methode von HistoryFn aufgerufen, die mit Hilfe der versteckten Variable, die mit GEHALT verbunden ist, ein bestimmtes Gehalt errechnet. Auf der anderen Seite müssen Methoden der Annotationsklasse explizit angewendet werden, wenn eine Anfrage über die Historie eines Objektes gemacht werden soll. Die Anfrage, die schon im OODAPLEX-Beispiel beschrieben wurde, kann mit Hilfe von Annotationen so ausgedrückt werden.

```
((employee select:[:e|e.ENAME = ´TOM´]).GEHALT.choose:
        (timeWhen:[:e|e.MANAGER = ´AL´])
).value
```

Annotationen, wie die Klasse HistoryFn, machen es einfacher, die Gültigkeitszeit zu implementieren. Die Optimierung ist immer noch ein Problem, da alle Zeitmanipulationen durch benutzerdefinierte Funktionen ausgeführt werden.
Das System VISION vertritt einen ähnlichen Ansatz [30, 31]. Attribute sind im VISION-Modell, genau wie in OODAPLEX, Funktionen. Aber das Modell macht den zusätzlichen Schritt, Funktionen als Objekte zu betrachten. Funktionen können dann sogenannte Metafunktionen zugewiesen werden. Diese Metafunktionen sind den oben beschriebenen Methoden der Annotationsklassen sehr ähnlich.

3.1.3 Direkte Aufnahme der Zeit in das Datenmodell

Die meisten Forscher haben spezielle Datenmodelle und Anfragesprachen vorgeschlagen, um zeitvariante Informationen zu unterstützen. Seit 1990 wurden sieben solcher Datenmodelle vorgeschlagen.

Zwei dieser Vorschläge erweitern Anfragesprachen für das Entity-Relationship-Modell. Das temporale Entity-Relationship-Modell (TEER) von Elmasri ist eine Erweiterung von GORDAS [8]. Im TEER Modell werden sowohl an Entities als auch an Relationships Zeitstempel in Form von temporalen Elementen [9] vergeben. Die where-Klausel kann zusätzliche Konstrukte enthalten, um temporale Elemente zu manipulieren. Außerdem hat Wuu eine separate Anfragesprache definiert, die sowohl auf dem Standard E-R-Modell als auch auf SQL basiert. Diese Sprache nennt er SERQL [19] (Structured Entity-Relationship Query Language).
Käfer schlug ein Datenmodell und eine Anfragesprache mit komplexen Objekten vor. Temporale Konstrukte wurden implementiert, indem sie auf ein nichttemporales Datenmodell (genannt MAD) aufgesetzt wurden [49].
Die anderen Anfragesprachen werden über objektorientierte Modelle definiert. TOODM, das Datenmodell hinter TOSQL und TOOSQL, betrachtet Attribute als Zeitsequenzen, die Sequenzen von Paaren (Wert und temporales Element) sind [13, 10]. Die Anfragesprache OQL/T für das Modell OSAM/T beinhaltet eine when-Klausel sowie eine Reihe von temporalen Funktionen und Operatoren [14]. Sciore erweiterte das Datenmodell EXTRA, um zwischen versionierten und nicht versionierten Attributen zu unterscheiden. Gadia stellte die Sprache OOSTSQL, eine Erweiterung von SQL, vor. Diese Sprache unterstützt, ähnlich dem IXRM von Lorentzos, sowohl räumliche als auch temporale Daten [47].

3.1.4 Transaktionszeit

Auch die Transaktionszeit kann in objektorientierten Modellen unterstützt werden. Um die Unterstützung der Transaktionszeit zu betrachten, ist eine wichtige Unterscheidung vorzunehmen. Werden Objektinstanzen oder Attributwerte versioniert, spricht man vom *Extension Versioning*. Werden dagegen die Definitionen der Objekte versioniert, nennt man das *Schema Versioning*. Wenn Extension Versioning vorliegt, *kann* zusätzlich Schema Versioning unterstützt werden. Wenn kein Extension

Versioning vorhanden ist, ist Schema Versioning nicht relevant, weil nur die letzte Version des Schemas gespeichert bleiben muß.

Extension Versioning

Genau wie bei der Unterstützung der Gültigkeitszeit gibt es drei Hauptansätze, um Extension Versioning zu unterstützen. Der erste Weg ist, das Modell direkt zu nutzen. OODAPLEX folgt diesem Ansatz. Da die Semantik der Zeit willkürlich ist, kann die Transaktionszeit durch die Anhängung eines zusätzlichen Wertes an die Zeitfunktion dargestellt werden.
Im zweiten Ansatz werden allgemeine Erweiterungen zum Datenmodell genutzt, um zeitvariante Informationen zu modellieren. Sciores Annotationen können dazu benutzt werden, um Revisionen und Alternativen zu unterstützen (z.B. verzweigte Transaktionszeit). Er stellt die Annotationsklasse RevisionFn mit Methoden wie asOf:, checkin: und checkout: vor. Die Annotationsklasse AlternativeFn soll für verzweigte Transaktionszeit genutzt werden [15]. Auch entsprechende Metafunktionen im Modell VISION [30] erfüllen diese Funktionen.
Im dritten Ansatz werden Sprache und Modell so modifiziert, daß die Transaktionszeit explizit unterstützt wird. Das MANTISSE OODBMS vergibt Zeitstempel an den gesammten Objektgraph [1]. Konzeptionell wird der Objektgraph bei jeder Transaktion kopiert. In der Implementation werden die verschiedenen Versionen des Objektgraphes mittels Sharing genutzt. Käfer differenziert zwischen Objektattributen und Versionsattributen. Die letzteren variieren in den verschiedenen Versionen eines Objektes [49]. Im TOODM ist ein Attribut vom Typ $TS[]$ eine Menge von drei Tupeln, Objekt ID, Historie und Korrekturen [13]. Die Historie und die Korrekturen sind wiederum Mengen von drei Tupeln, die den Objekt ID des historischen Wertes, den Attributwert und den Zeitstempel des temporalen Elements enthalten. Die Informationen in den Mengen Historie und Korrekturen ermöglichen eine Rekonstruktion jedes früheren Zustandes der Datenbank. In Postgres, dem System, das die Sprache Postquel

unterstützt, werden die Tupel mit zwei Transaktionszeitstempeln versehen, die das Intervall bezeichnen, in dem sie logisch in der Datenbank präsent waren [33]. Ong und Goh haben gezeigt, daß ein DBMS, das lineare Transaktionszeit, Regeln, Prozeduren und gemeinsame Anfragen unterstützt, auch die Hauptkonzepte des Versioning inklusive simulierter verzweigter Transaktionszeit unterstützen kann [28].
Das Modell des Chou-Kim-Versioning [20] hat im Moment die weiteste Akzeptanz. Es wurde in ORION, in IRIS und in OQL implementiert. Das Chou-Kim Modell verwendet Objektversioning. Ein Schüsselwort wird dazu benutzt, um anzuzeigen, ob eine Klasse versionierbar ist. Die Versionen werden durch das System automatisch, beginnend mit der Eins, numeriert. Die Versionen werden mit einem Namen, einem Zeitstempel und einem Status versehen. Versionen können eingeteilt werden in transiente Versionen, die jederzeit upgedatet werden können, Arbeitsversionen, die nicht upgedatet, aber shared verwendet werden können und freigegebene Versionen, die überhaupt nicht upgedatet oder gelöscht werden können. Versionen können innerhalb dieser Klassifikation jeden Zustand annehmen. Das Modell unterscheidet zwischen generischen und spezifischen Referenzen. Eine generische Referenz zeigt auf ein Objekt und alle seine Versionen und kann dazu benutzt werden, um die Anbindung eines Attributs an eine bestimmte Version des Objektes zu verzögern. Eine spezifische Referenz dagegen zeigt auf eine fixe Version des Objektes. Versionen können auch zueinander in Beziehung gesetzt werden. Dazu wird ein azyklischer Graph der Versionen jedes generischen Objektes angelegt. Automatisch verwendet das System zwei Beziehungen. Die Beziehung *version-of* besteht zwischen jedem versionierbaren Objekt und seinen Versionen, und die Beziehung *derived-from* besteht zwischen Versionen und den direkt davon abgeleiteten Versionen. In den gespeicherten Daten verwaltet das System ein generisches Objekt, das die gesamte Versionsableitungshierarchie enthält.

3.1.5 Schema Versioning

Durch die Schemaevolution kann ein Schema sich mit den Anforderungen der Applikation verändern. Beim Schema Versioning gibt es verschiedene Schemata zu verschiedenen Transaktionszeiten. Zum Beispiel könnte ein Attribut *NumKinder* zur Klasse *employee* hinzugefügt werden.
Es gibt zwei Wege, um Schemaevolution zu implementieren. In MANTISSE wird der „eifrige" Weg benutzt. Das heißt, daß alle Instanzen des Objektes augenblicklich geändert werden, damit sie das neue Attribut enthalten. In ORION wird der „faule" Weg beschritten. Das bedeutet, daß die Änderung einer Objektinstanz erst stattfindet, wenn eine Objekt durch eine Applikation von der Festplatte gelesen wird. Die Objektinstanz wird dann in das aktuelle Schema transformiert. In beiden Implementationen ist logisch nur ein Schema in Gebrauch. ORION verteilt die Last des Updatings nur.
Beim Schemaversioning gibt es mehrere Schemata, die logisch genutzt werden. Im Beispiel brauchen alle Instanzen vor der Änderung des Objektes *employee* kein Attribut *NumKinder*. Wenn man die Datenbank zu einem Zeitpunkt vor dieser Änderung zurückrollt, wird auch das Schema zurückgerollt. Dieses Verhalten ist konsistent mit der Semantik von Transaktionszeit.
Drei objektorientierte Datenmodelle unterstützen Schema Versioning. In MANTISSE wird das gesamte Schema mit einem Transaktionszeitstempel versehen. Postgres implementiert Schema Versioning dadurch, daß der Systemkatalog aus Transaktionszeitrelationen besteht. Beide betrachten Transaktionszeit als linear.
Das Modell des Chou-Kim-Versioning sieht, ähnlich wie MANTISSE und Postgres, das gesamte Schema als ein versioniertes Objekt. Es unterscheidet sich jedoch von MANTISSE und Postgres dadurch, daß es einen Status (transient oder arbeitend) mit jedem Schema verbindet. Außerdem wird eine Versionsableitungshierarchie verwaltet. Das bedeutet, daß die Transaktionszeit auch verzweigt werden kann. Das Modell schließt auch Regeln ein, welche Schemaversion welche Objekte „besitzt". Allerdings

wurde das Schema Versioning mit dem Chou-Kim-Modell noch nicht implementiert.

3.2 TOOA von Rose und Segev

Die temporale objektorientierte Algebra (TOOA) von Rose und Segev [10] basiert auf dem temporalen objektorientierten Datenmodell (TOODM), das in [13] vorgestellt wurde.
Im TOODM wird eine Klasse als eine Erweiterung des Typs einer Objektinstanz gesehen. Ein Typ dient als Schablone, um die Struktur und das Verhalten eines Objektes einzukapseln. Jedes Objekt hat einen globalen systemgenerierten Identifier, der unabhängig vom Status oder dem Ort des Objektes ist. Der Status eines Objektes wird durch die Werte seiner Attribute beschrieben. Neue Typen müssen als Untertypen bereits existierender Typen definiert werden. Ein Untertyp erbt die Eigenschaften seines Supertyps.
Benutzerdefinierte komplexe Typen werden als Untertyp von V-TYPE (versionable type) definiert. Für jeden Untertyp von V-TYPE wird eine Historie gespeichert. Metadaten sind Untertypen von NV-TYPE (non versionable type). Das heißt, die Historie der Typdefinitionen wird nicht berücksichtigt. Die Instanzen der Untertypen von NV-TYPE sind versionierbar. PTYPE (primitive type) haben atomare Werte und eine Menge von Operationen. Kollektionstypen wie SET, TUPLE, SEQUENCE und LIST sind parametrisierte Typen, die benutzt werden, um andere Typen zu erweitern.
Ein besonderer Kollektionstyp ist die *Zeitsequenz*. Die Zeitsequenz $TS[O_i]$ hat den Typ O_i als Parameter. Jedes Zeitsequenzobjekt enthält eine Historie und eine Korrekturhistorie, die jeweils aus Paaren $(A;TL)$ bestehen. A repräsentiert dabei ein Tupel von Attributwerten und TL ein Tupel von Time-Line Werten. Die Gültigkeitszeit (vt) und die Transaktionszeit (tt) sind in diesem Tupel enthalten. Jedes Attribut hat einen Datentyp, der entweder schrittweise konstant (SWC), ein Ereignis, kontinuierlich oder benutzerdefiniert ist. Wenn der Datentyp benutzerdefiniert

ist, sollte eine Interpolationsfunktion (interp-f) gestellt werden, um Werte für Zeitpunkte zu berechnen, die nicht als Datenpunkte aufgezeichnet werden. Jede Time-Line TL_i hat eine Granularität, die entweder eine Kalenderzeit oder eine ordinale Nummer ist. Der Transaktionszeitwert für die Korrekturhistorie (corr-history) muß größer sein als der der Objekte in der entsprechenden Historie. Das Lebensspannenattribut ist als Vereinigung von temporalen Elementen aus der Historie definiert. Temporale Elemente sind eine endliche Vereinigung von Time-Lines. Temporale Elemente sind geschlossen in Bezug auf Mengendurchschnitt, Mengenvereinigung und Mengenkomplement. Die Lebensspanne eines Objektes ist die Vereinigung der Lebensspannen aller beteiligten Klassen.
Die Operatoren für Zeitsequenzen werden in [2] definiert. Allerdings erfolgte die Definition nur für die Time-Line vt. Diese Operatoren wurden erweitert, um multiple Time-Lines zu handhaben. Im TOODM geschieht das durch die Repräsentation von TL_i als Tupel von Time-Lines. Weil man Informationen mit oder ohne Korrekturen betrachten kann wurde im TOODM ein Operator *merge* entwickelt. *Count(), First(), Last()* und *Nth()* geben die Nummer von Datenpunkten in der Sequenz zurück, das erste, das letzte und das n-te Paar von Werten.
Abbildung 3.2 zeigt die prinzipielle Definition einer Zeitsequenz.

3.2.1 Der Schemagraph

Abbildung 3.2.1 zeigt ein Beispiel für einen Schemagraphen (SG). Unbeschriftete Pfeile bezeichnen eine „ist"-Beziehung, ausgehend von einem Obertyp. Der Schemagraph funktioniert als die logische Repräsentation der Datenbank. Er besteht aus Nodes (Typen) und Links (Eigenschaften der Typen). Die Menge der Typen ist eine disjunkte Vereinigung der primitiven Typen (PTYPE) und der komplexen Typen. Jeder komplexe Typ hat einen Identifier, Attribute, eine Liste der Obertypen, Operationen und Integritätsbedingungen. Wenn eine Eigenschaft eines Typs einwertig ist, werden deren Werte als $TS[O_i]$ gespeichert. Bei mehrwertigen Eigenschaften werden die Werte als $TS[SET/LIST[O_i]]$ abgelegt.

Abbildung 3.1: Definition einer Zeitsequenz $TS[Oi]$ als Subklasse von Oi

```
surr:OID

history:{(A;TL)} where          A=TUPLE(Ai)
                                TL=TUPLE(TLi:Ti)
        and data.type(Ai):      {SWC, Event, CONT, User}
        and gran(TLi):          CalendarSet or Ordinal
        and interp-f(Ai):       Functions

default-order:  TLi in TL

corr-history:   same as history except value(tt)>value(tt)
                in the history sequence

lifespan:       Union of Temporal Elements in history

        .
        . (Attribute)
        .

COUNT() FIRST() LAST() Nth()

Accumulate(Agg-Op(Target-Ai),Grp(acc-pred))
Aggregate(Agg-Op(Target-Ai),Grp(TLi,gran-value))
Select(LIST[Ai],Conditions(Ai or TLi))
Restrict(TS,condition(another property of Oi))
Merge(History-TS,Corr-TS,Temporal-Element)

        .
        . (Operatoren)
        .
```

Ein Link im Schemagraphen repräsentiert eine Eigenschaft und ist vom definierenden Objekt zu seiner Domäne gerichtet. Weil mehrere Links zwischen den beiden gleichen Typen existieren können, hat jeder Link einen eindeutigen Namen. Eigenschaften mit primitiven Domänen können nicht als inverse Links repräsentiert werden.

3.2.2 Objekte und der Objektgraph

Jedes Objekt hat einen eindeutigen Identifier für jede Instanz (IID). Er ist eine Aneinanderreihung des Typidentifiers und des Objektidentifiers. Der Objektgraph, OG, ist die Erweiterung des Schemagraphen. Seine Nodes sind IID's, und die Links sind Zuweisungen von IID's zu Zeitsequenzobjekten.
Abbildung 3.2.2 zeigt ein Beispiel für einen Objektgraphen und bezieht sich auf den Schemagraphen in Abbildung 3.2.1.

3.2.3 Überblick über TOOA

Voraussetzungen

Ereignisse werden so betrachtet, als ob sie keine Dauer haben. Sie verursachen die Veränderung des Status eines Objektes. Die Taxonomie der Zeit, die in [39] für Gültigkeitszeit und Transaktionszeit gegeben wird, wird verwendet. Der Status eines Objektes hält über eine bestimmte Zeitdauer und wird durch die Werte seiner Attribute repräsentiert. Die Zeitperiode, in der ein Status unverändert bleibt, wird in Gültigkeitszeit angegeben. Die Transaktionszeit gibt an, wann ein Objekt aufgezeichnet wurde. Zeit wird durch einen primitiven Typen repräsentiert, der unabhängig von Ereignissen ist. Die Zeit kann sowohl absolut als auch relativ angegeben werden. Außerdem können spezialisierte Zeittypen mit verschiedenen Ordnungen definiert werden.
Die Semantik von existierenden Operationen – Anfragen, Korrekturen und Updaten – unterscheiden sich in TOODM von denen in einem statischen Modell. Eine Anfrage kann zum Beispiel eine Zeitsequenz von Wer-

Abbildung 3.2: Schemagraph

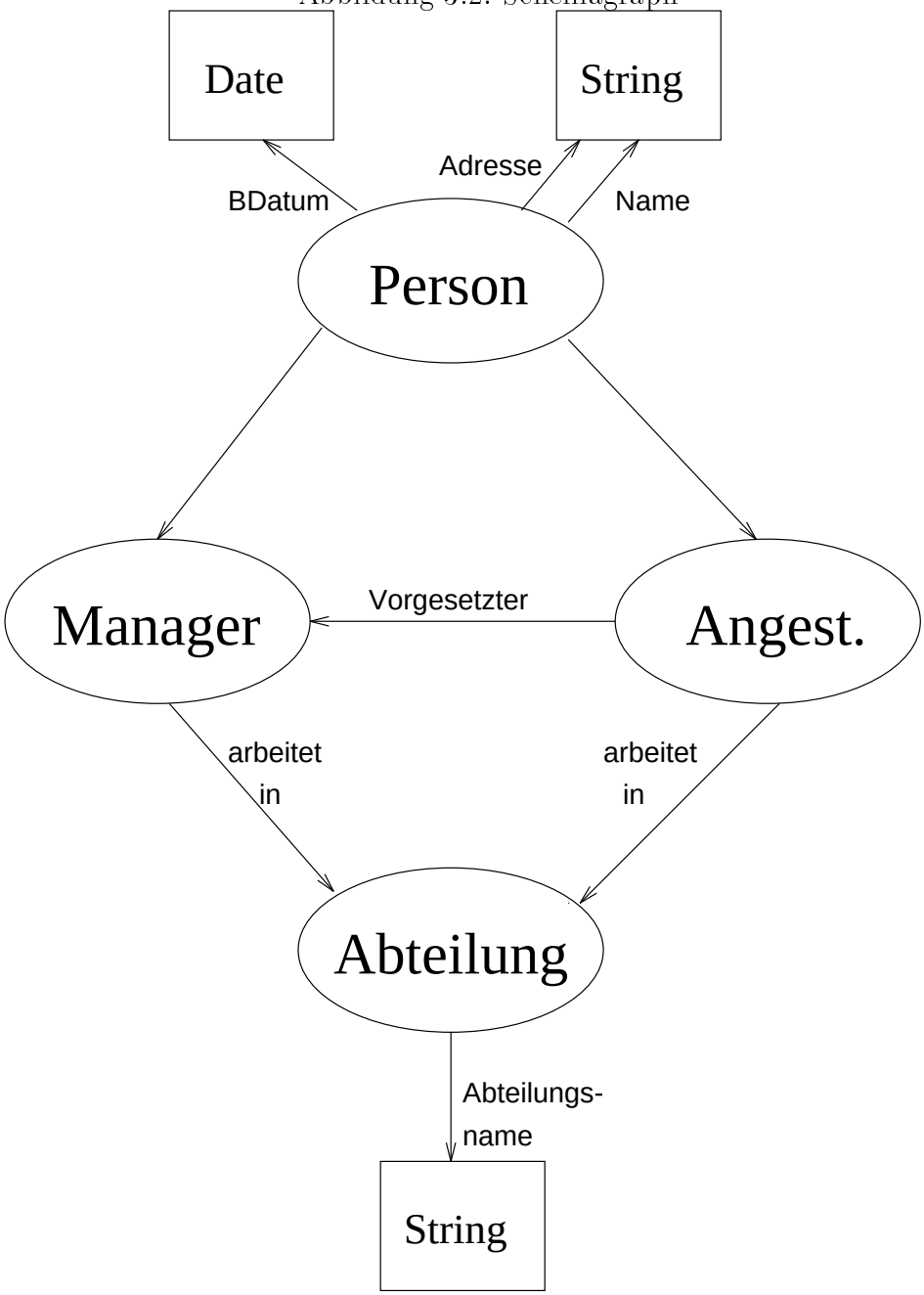
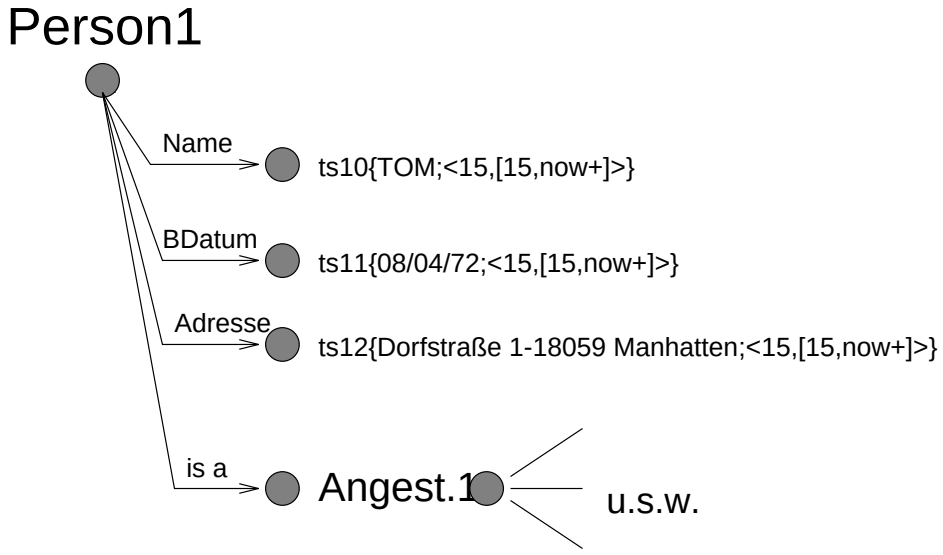


Abbildung 3.3: Objektgraph



ten zurückgeben. Wenn ein Attributwert geändert wird, wird ein neuer Wert in die Zeitsequenz des Attributs hinzugefügt. Wenn ein existierendes Objekt geändert wird, endet sein Gültigkeitsintervall eine Zeiteinheit vor der Änderung $(t-1)$. Das neue Zeitintervall beginnt zur Zeit t. Der Endpunkt des neuen Intervalls kann ein spezifischer Punkt oder now+ sein, wenn der Wert für jetzt und zukünftig gilt. Das Löschen eines Objektes bewirkt das Ende seiner Lebensspanne bei gleichzeitiger Beibehaltung der Informationen in der Datenbank.

Objektleveloperatoren

1. $*_T^k$, **Temporale Zuweisung**
   Dieser binäre Operator gibt eine Sammlung von Graphen von Objekten zurück. Für jeden Graph im Resultat wird einer Instanz der Klasse oder der Klassenhierarchie, die durch den ersten Operanden spezifiziert ist, eine Instanz oder Instanzen der Klasse oder der Klassenhierarchie mit der Wurzel in der Klasse des zweiten Operanden durch den Link k zur Zeit T zugewiesen. Wenn die Klasse, die durch den zweiten Operanden repräsentiert wird, nicht primitiv

ist, kann der Zuweisungsoperator noch einmal angewendet werden, um einen Attributwert dieser Klasse zu erhalten. Man kann den Zuweisungsoperator auch als Navigationsoperator sehen, mit dem man sich durch den Schemagraphen arbeitet, der für jeden Link zu einer bestimmten Zeit existiert. Das heißt, daß T in der Lebensspanne des Links k enthalten sein muß.

2. $|_T$, **Temporal Complement**
   Diese Operation liefert als Ergebnis die Member des zweiten Operanden, die zu einer Zeit T *nicht* mit Objekten verbunden sind, die zum ersten Operanden gehören. Man könnte so zum Beispiel erfragen, wann ein Angestellter zu einer bestimmten Zeit in einer bestimmten Abteilung *nicht* gearbeitet hat. Die Operation $Angest.|_{[15,20]}^{arbeitet-in} A1$ liefert zum Beispiel, wenn das Objekt $A1$ ein Objekt der Kasse *Abteilung* ist und die Spielzeugabteilung repräsentiert, alle Angestellten, die im Zeitintervall $[15, 20]$ nicht in der Spielzeugabteilung gearbeitet haben.

3. $\sigma - IF_{TP}$, **Temporale Selektion unter Benutzung eines temporalen Prädikates**
   Select-IF ist ein unärer Operator. Er liefert die Menge der Objekte, die die Bedingung, die im temporalen Prädikat, TP, formuliert wird, erfüllen. Das temporale Prädikat besteht aus einem nichttemporalen Prädikat (NTP), einem Quantifizierer ($\forall$, $\exists$) und einem temporalen Element, über dem die Selektion ausgeführt wird. Das NTP kann aus mehreren Prädikaten mit Hilfe von *and* oder *or* kombiniert werden. Vergleiche können nur zwischen Objekten gleichen Typs vorgenommen werden; primitives Objekt mit primitivem Objekt oder IID mit IID.

4. $\sigma - WHEN_{NTP}$, **Temporale Selektion unter Benutzung eines nichttemporalen Prädikates**
   Select-WHEN ist ein unärer Operator, der die Menge der Objekte liefert, die die Bedingung erfüllen, die durch das nichttemporale

Prädikat bestimmt wird. Die Lebensspanne der resultierenden Objekte ist die Menge der Zeiten, zu denen die NTP-Bedingung erfüllt ist.

5. $\cap_T$, **Temporaler Durchschnitt**
   $\cap_T$ ist ein binärer Operator ähnlich dem Join-Operator in der Relationenalgebra. Es gibt nur ein Resultat, wenn die Graphen, die „gejoint" werden sollen, mindestens ein Node gemeinsam haben.

6. $\cup_T$, **Temporale Vereinigung**
   $\cup_T$ ist ein binärer Operator ähnlich dem Vereinigungsopeator der Relationenalgebra, außer daß er über heterogenen Mengen von Objekten arbeitet. Die Graphen der Operanden müssen nicht die gleiche Topologie oder gleiche Nodes haben. Nur Objekte, die zur Zeit T in einem der Operanden existieren erscheinen in der Ergebnismenge von Graphen.

7. $-_T$, **Temporale Differenz**
   $-_T$ ist ein binärer Operator ähnlich der Mengendifferenz in der Relationenalgebra, außer daß er über Mengen von heterogen Objekten arbeitet. Jedes Objekt, das im ersten Operand zur Zeit T existiert und ein Element des zweiten Operanden enthält, das zur Zeit T existiert, erscheint nicht im Ergebnis der Operation.

Updateoperatoren

Die Updateoperatoren sind in COMPLEX TYPE definiert und werden von allen benutzerdefinierten Typen geerbt, außer von Untertypen primitiver Typen und Kollektionstypen.

- $Add - Obj - Instance(o, A)$
  Diese Operation fügt ein Objekt zu EXT(A) hinzu. A ist der definierende Typ von Objekt o. Die Operation beinhaltet auch das Anlegen eines neuen IID's für das Objekt. Der Klassenteil des IID's ist der Identifier von A. Der Anwender muß das Gültigkeitszeitintervall selbst angeben. Die Transaktionszeit ist die aktuelle Zeit.

- $Del - Obj - Instance(o, A)$
  Diese Operation löscht ein Objekt von EXT(A). A ist der definierende Typ von Objekt o. Diese Operation beendet die Lebensspanne aller Eigenschaften des Objektes. Auch Beziehungen zwischen den Objekten sind Eigenschaften und werden ebenfalls beendet.

- $Mod - Obj - Instance(o, A, [List_of_property : value])$
  Modifiziert ein Objekt o in EXT(A). Die Operation beinhaltet auch ein Update aller Zeitsequenzen, die mit jeder Eigenschaft in der Liste verbunden sind. Der Endpunkt der Gültigkeitszeit des Objektes wird auf die aktuelle oder eine anwenderspezifizierte Zeit gesetzt. Die Transaktionszeit des neuen Objektes wird auf die aktuelle Zeit gesetzt.

Schemaleveloperatoren

Die Domäne der Schemaleveloperatoren ist die Extension der Metadatentypen. Diese Klassen enthalten Definitionen von Typen, Variablen, Operationen und Kollektionstypen. Die Metadatenklassen können mit Hilfe der Objektleveloperatoren abgefragt werden, weil sie wie Daten behandelt werden. Darum braucht man auf dem Schemalevel nur Updateoperatoren um Typen hinzuzufügen, zu löschen oder umzubenennen und so weiter.

1. $Add - Msg(Type - OID, Name, LIST[O_i], TUPEL[Meta - Vars], SET[Meta - Msgs], UserID, SET[BOOL], vt)$
   Der Anwender spezifiziert einen String für den Namen, eine Liste existierender Identifier, die den Supertyp des neuen Typen angeben, seinen User-ID und die Gültigkeitszeit, wann der Typ benutzt werden darf. Danach wird der Anwender aufgefordert, Instanzvariablen zu definieren, die die Struktur des Objektes, der Botschaften und der Integritätsbedingungen repräsentieren. Der Operator bewirkt die Erstellung einer Instanz von $Meta - Types$, einer Menge von Instanzen von $Meta - Vars$, einer Menge von Instanzen von $Meta - Msgs$ und eine Klasse mit dem Namen $SET[Name]$, die die Extension des neuen Typs repräsentiert.

2. $Del - Type(OID, Name, UserID, END(vt))$
   Der Anwender spezifiziert den Namen des zu löschenden Typs, den User-ID und das Datum, an dem die Löschung aktiv wird. Namen von Typen müssen eindeutig sein. Darum kann der Name des Typs benutzt werden, um den OID zu finden. Die Lebensspannen aller Eigenschaften, die mit diesem OID verbunden sind, enden.

Außerdem gibt es auf dem Schemalevel auch Operatoren, um Variablen und Botschaften zu definieren und zu löschen, sowie Operatoren um Typen umzubenennen.

Kapitel 4

Zusammenfassung

Wie die vorliegende Arbeit zeigt, gibt es eine Vielzahl von Möglichkeiten, um den Zeitaspekt in Datenbanken zu integrieren. Durchgesetzt hat sich davon noch keine. Ausgehend von TQuel wird heute versucht, eine benutzerfreundlichere Sprache zu definieren. Diese Bestrebungen gehen auch mit dringend benötigten Standardisierungsbemühungen einher, denn der fehlende Standard ist einer der Gründe dafür, daß sich noch kein temporales Datenbanksystem zur Nutzungsreife entwickelt hat. Die Standardisierungsanstrengungen mündeten in der Sprache TSQL2 und inzwischen auch TSQL3, an deren Entwicklung fast alle wichtigen Forscher auf dem Gebiet der relationalen temporalen Datenbanken beteiligt waren (z.B. Snodgrass, Ahn, Ariav, Batory, Clifford, Elmasri, Käfer, Leung, Lorentzos, Segev u.s.w.). Algebren, die TSQL unterstützen, werden damit an Bedeutung gewinnen. Der Schwerpunkt der Forschungen wird auch weiterhin auf der Erweiterung des Relationenmodells liegen.
Ein wichtiger Aspekt dieser Forschungen wurde in dieser Arbeit weitestgehend nicht berücksichtigt. Die sogenannte Schemaevolution bereitet den Forschern immer noch Probleme. Eine Relation in einer Datenbank muß mit der Zeit neuen Anforderungen angepaßt werden, ohne dabei Daten zu verlieren. Diesen Vorgang bezeichnet man als Schemaevolution. Dabei bereiten neue zeitabhängige Attribute ebenso wie die Veränderung der Zeitgranularität Probleme. Die Lösung ist noch immer Gegenstand der Forschung. Aus diesem Grund, und weil es für eine allgemeine, verglei-

chende Betrachtung mir nicht wichtig erschien, wurde dieses Thema in dieser Arbeit zum größten Teil ausgeklammert.

Literaturverzeichnis

[1] ADB. Matisse versioning. Technical report, ADB/Intellic, 1993.

[2] A.Soshani A.Segev. Logical modeling of temporal databases. *Proceedings of ACM SIGMOD International Confrence on Management of Data*, pages 154–162, Mai 1987.

[3] A.Soshani A.Segev. The representation of a temporal data model in the relational environment. *Lecture Notes in Computer Science*, pages 454–466, 1988.

[4] Jacov Ben-Zvi. A time-view od data in a relational database system. Computer Science Department UCLA, Juni 1981.

[5] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[6] B.Mahbod D.Beech. Generalized version control in an object-orientated database. *Proceedings of the International Confrence on Data Engeneering*, pages 14–22, Februar 1988.

[7] Veda C. Storey Debarata Dey, Terence M. Barron. A complete temporal relational algebra. *The VLDB Journal*, 5:167–180, 1996.

[8] G.Wiederhold E.Elmasri. Gordas: A formal high-level query language for the entity-relationship model. *Entity-Relationship Approach to Software Engeneering*, 1983.

[9] G.Wuu E.Elmasri, R.Wuu. A temporal model and query language for er databases. *Proceedings of the 9th International Confrence on Entity-Relationship Approach*, pages 76–83, Februar 1990.

[10] Arie Segev Ellen Rose. Tooa: A temporal object-orientated algebra. *Proceedings of ECOOP'93*, 1993.

[11] Navathe Elmasri. *Fundamentals of Database Systems.* Benjamin/Cummings Pub. Co., 1989.

[12] E.McKenzie. An algebraic language for query and update of temporal databases. Computer Science Department, University of North Carolina at Chapel Hill, September 1988.

[13] A.Segev E.Rose. A temporal object-orientated data model with temporal constraints. *Proceedings of the 10th International Confrence on Entity Relationship Approach*, Oktober 1991.

[14] E.Sciore. Using annotations to support muliple kinds of versioning in an object-orientated database system. *ACM Transaction on Database Systems*, 16(3):417–438, September 1991.

[15] E.Sciore. Versioning and configuration management in an object-orientated data model. *VLDB Journal*, 1994.

[16] V.Matos G. zsoyoglu, M.Z.zsoyoglu. Extending relational algebra and calculus with set valued attributes and aggregate functions.

[17] G.Ariav. A temporally orientated data model. *ACM Trans. Database Syst.*, 11:499–527, Dezember 1986.

[18] J.Clifford G.Ariav. Temporal data management: Models and systems. *New Directions for Database Systems*, pages 168–185, 1986.

[19] G.Wuu. Serql:an er query language supporting temporal data retrieval. *ACM*, (0-89791-317-5/89/0005/0284):284–293, 1989.

[20] W.Kim H.Chou. A unifying framework for version control in a cad environment. *Proceedings of the 12th International Confrence on VLDB*, pages 336–344, 1986.

[21] Alber Croker James Clifford. The historical relational data model (hrdm) revisited. *Temporal Databases - Theory, Design and Implementation*, pages 6–26, 1993.

[22] Albert Croker James Clifford. The historical relational data model (hrdm) and algebra based on lifespans. *Proceedings of the international Conference on Data Engineering*, pages 528–537, 1987.

[23] A.Crooker J.Clifford. Objects in time. *IEEE Data Engineering*, pages 189–196, Dezember 1988.

[24] A.Tuzhilin J.Clifford, A.Crooker. On completeness of historical relational query languages. Technical Report IS-91-41, New York University, Dezember 1991.

[25] A.Tuzhilin J.Clifford, A.Crooker. On completeness of historical relational query languages. Technical Report IS-91-41, New York University, Bezember 1991.

[26] A.U.Tansel J.Clifford. On algebra for historical relational databases: Two views. *Proceedings of ACM SIGMOD International Confrence on Management of Data*, pages 247–265, Mai 1985.

[27] Richard T. Snodgrass L. Edwin McKenzi. Evaluation of relational algebras incorporating the time dimension in databases. *ACM Computing Surveys*, 23(4):501–540, Dezember 1991.

[28] J.K.S.Goh L.Ong. A unified framework for version modeling using production rules in a database system. Technical Report UCB/ERL/ M90/33, Electronics Research Laboratory, Collage of Engineering, University of California, April 1990.

[29] Nikos A. Lorentzos. The interval-extendet relational model and its application to valid-time databases. *Temporal Databases*, pages 67–91, 1993.

[30] C.Sciore M.Caruso. Meta-fuctions and contexts in an object-orientated database language. *Proceedings of ACM SIGMOND International Confrence on Management of Data*, pages 56–65, Juni 1988.

[31] E.Sciore M.Caruso. The vision object-orientated database management system. *Advances in Database Programming Languages*, pages 147–163, 1990.

[32] S.L.Vandenburg M.J.Caray, D.J.DeWitt. A data model and query language for exodus. *Proceedings of ACM SIGMODInternational Confrence on Management of Data*, pages 413–423, Juni 1988.

[33] M.Hirohama M.L.Stonebreaker, L.Rowe. The implementation of postgres. *IEEE Transactions on Knowledge and Data Engineering*, pages 125–142, März 1990.

[34] K.Kahn M.Stefik, D.Bobrow. Integrating access-orientated programming into a multiparadigm environment. *IEEE Software*, 3(1):10–18, Januar 1986.

[35] N.A.Lorentzos. A formal extension of the relational model for the representation and manipulation of generic intervals. Birkbeck College, University of London, 1988.

[36] N.L.Sarda. Hsql: A historical query language. *Temporal Databases*, pages 110–140, 1993.

[37] R.Snodgrass. The temporal query language tquel. *ACM Trans. Database Syst.*, 12:247–298, Juni 1987.

[38] R.Snodgrass. An overview of tquel. *Temporal Databases*, pages 141–182, 1993.

[39] I.Ahn R.Snodgrass. A taxonomy of time in databases. *Proceedings of ACM SIGMOD International Confrence on Management of Data*, pages 236–246, Mai 1985.

[40] I.Ahn R.Snodgrass. Temporal databases. *IEEE Comput.*, 19:35–42, 1986.

[41] Rafi Ahmed S. B. Navathe. A temporal relational model and a query language. UF-CIS Technical Report TR-85-16, Database Systems Research and Development Center Computer and Information Sciences Department University of Florida, Gainesville, Florida 32611, Aprill 1986.

[42] P.Mason S.Jones. Legol 2.0: A relational specification language for complex rules. *Information Sytems 4*, 4:293–305, November 1979.

[43] S.K.Gadia. Toward a multihomogenous model for a temporal database. *Proceedings of the 1986 ACM Computer Science Confrence*, 1986.

[44] S.K.Gadia. A homogenous relational model and query language for temporal databases. *ACM Transactions on Database Systems*, 13(4):418–448, Dezember 1988.

[45] C.S.Yeung S.K.Gadia. A generalized model for a relational temporal database. *Proceedings of ACM SIGMOD International Confrence on Management of Data*, pages 251–259, Juni 1988.

[46] Abdullah Uz Tansel. Adding time dimension to relational model and extending relational algebra. *Information Systems*, 11(4):343–355, 1986.

[47] S.K.Gadia T.S.Cheng. A seamless object-orientated model for spatio-temporal databases. (TR-92-46), 1992.

[48] W.Käfer. Temporal selection, temporal projektion, and temporal join revisited. 1992.

[49] H.Schöning W.Käfer. Mapping a version model to a complex-object data model. *Proceedings of ACM SIGMOD International Confrence on Management of Data*, pages 266–275, Juni 1992.