

# SEMINAR IMPERFEKTION UND DATENBANKEN

## Schemaerweiterung zur Abbildung von imperfekten Daten

Andreas Merkel

1. Dezember 2003

### 1 Einführung

Diese Seminararbeit beschäftigt sich mit der Frage, wie imperfekte Daten in einem relationalen Datenbanksystem verarbeitet werden können.

#### 1.1 Eigenschaften des relationalen Modells

Das relationale Modell scheint auf den ersten Blick recht ungeeignet zur Abbildung von unscharfer Information. Jedem Attribut ist ein wohldefinierter Wertebereich zugeordnet, der festlegt, welche Werte das Attribut annehmen kann. Dabei sind im relationalen Modell nur atomare Attributwerte zugelassen, d.h. keine zusammengesetzten Datentypen oder gar Mengen. Impräzise Informationen wie „die Länge des Staus beträgt 3 – 4 Kilometer“ können damit nicht modelliert werden.

Darüber hinaus können auch keine stochastischen Unsicherheiten modelliert werden. Hat ein Attribut im relationalen Modell einen Wert, so ist dieser sicher. Eine Ausnahme bilden die Nullwerte, die anzeigen, dass ein Wert nicht oder noch nicht bekannt ist. Es besteht jedoch keine Möglichkeit, mit Hilfe des Attributs „Dauer des Staus“ auszudrücken, dass ein Stau sich mit 60% Wahrscheinlichkeit innerhalb der nächsten und mit 80% Wahrscheinlichkeit innerhalb der nächsten zwei Stunden auflösen wird.

Weiter ist es nicht möglich, Daten einzufügen, die zwar von der Intention her zu einem bestimmten Attribut passen, aber nicht in dessen vorgesehenen Wertebereich. Ist ein Attribut „Ort des Stauendes“ vorgesehen, als dessen Wert die Angabe eines bestimmten Autobahnkilometers erwartet wird und man weiß nur, dass das Stauende sich auf der Höhe von Karlsruhe befindet, so kann diese Information nicht eingefügt werden, da eine präzise Angabe verlangt wird.

Auch unvollständige Daten können nicht modelliert werden. Existiert in einer Relation „Autobahnbaustellen“ das Attribut „Datum des Baubeginns“, von einer bestimmten Baustelle steht jedoch nur fest, dass mit den Bauarbeiten irgendwann im Jahr 2005 begonnen werden soll, kann man diese Information nicht in die Datenbank einfügen, da ein genaues Datum mit Tag und Monat verlangt wird. Weist man den Tag und Monat willkürlich zu, indem man z.B. den 01.01.2005 einträgt, entsteht die Illusion einer Genauigkeit, die in Wirklichkeit nicht vorhanden ist, denn die Unvollständigkeit des Datums wird nicht modelliert.

Ebenso wie die in ihr enthaltenen Daten, sind auch Anfragen an eine relationale Datenbank scharf. Die Merkmalswerte in der Datenbank müssen mit einem vorgegebenen Abfragewert übereinstimmen oder nicht übereinstimmen, müssen größer oder kleiner als ein vorgegebener Wert sein. Eine Anfrage nach Tupeln, die mit einem vorgegebenen Wert „mehr oder weniger“ übereinstimmen, ist jedoch nicht möglich. Auch bei Anfragen, die mit Wildcards arbeiten, wie z.B. der „like“-Ausdruck in SQL, muss stets eine perfekte Übereinstimmung des Attributwertes mit dem vorgegebenen Muster gegeben sein, damit ein Tupel selektiert wird.

#### 1.2 Erweiterungsmöglichkeiten

Um unscharfe Information nun doch mit Hilfe eines relationalen Schemas modellieren zu können, muss dieses erweitert werden. Von den vielen Ansätzen zur Erweiterung relationaler Schemata werden nun zwei etwas genauer betrachtet: Zum einen gibt es die Möglichkeit, nicht die in der Datenbank enthaltenen Daten selbst

als unscharf zuzulassen, sondern lediglich unscharfe Anfragen zu ermöglichen. Der zweite Ansatz basiert auf dem Fallenlassen der Forderung nach Atomizität der Attributwerte, jedoch nicht generell, sondern nur in Ausnahmefällen.

## 2 Schemaerweiterung zur unscharfen Klassifikation

Der im Folgenden beschriebene Ansatz erweitert das relationale Schema durch ein Kontextmodell zur unscharfen Klassifikation. Mit Hilfe der Anfragesprache fCQL und linguistischer Variablen sind so unscharfe Anfragen an Datenbanken, die scharfe Daten enthalten, möglich. Vorgestellt wurde dieser Ansatz in [1] und [2].

### 2.1 Motivation

Bei umfangreichen Datenbeständen, wie sie z.B. in vielen Unternehmen existieren, ist es wünschenswert, die Daten zur Komplexitätsreduktion in verschiedene Klassen einzuteilen. Die existierenden relationalen Anfragesprachen wie z.B. SQL setzen voraus, dass die Anfragen mit der selben Präzision formuliert werden, mit der die Daten in der Datenbank vorliegen, es sind keine ungenau formulierten oder unscharfe Anfragen möglich. Oft ist man aber gar nicht an einer scharfen Einteilung interessiert, z.B. ist man bei einer Anfrage an eine Verkehrsdatenbank eher an allen Staus im näheren Umkreis interessiert, als dass man einen genauen Radius für diesen Umkreis spezifizieren will.

Mit dem Ansatz, auf Datenbanken mit präzisen Daten unscharfe Anfragen zu ermöglichen, vermeidet man den Aufwand, bestehende scharfe Datenbestände in ein Unschärfe unterstützendes Datenbankformat zu konvertieren und umgeht den Overhead, zusätzlich zu jedem Datum Informationen über dessen Unschärfe abzuspeichern, kommt aber doch dem Anwender entgegen, der eine Einteilung der Daten nach unscharfen Kriterien vornehmen will.

### 2.2 Klassifikation

Umfangreiche Datenbestände werden schnell unübersichtlich. Zur Analyse und Auswertung solcher umfangreichen Bestände ist deswegen eine Strukturierung und Einteilung notwendig. Nehmen wir als Beispiel eine Datenbank, in der Stauinformationen über gewisse Streckenabschnitte gespeichert sind. Es ist sinnvoll, Strecken mit ähnlichem Stauverhalten in Klassen zusammenzufassen, man erhält dann zum Beispiel die Klasse der Strecken, die man lieber weiträumig umfahren sollte, die derjenigen, für die man zusätzlich Zeit einplanen sollte und die der unbedenklich befahrbaren. Diese Einteilung erleichtert einem Autofahrer die Auswahl seiner Route und ist sicher hilfreicher als eine bloße Auflistung der Stauhäufigkeiten auf den einzelnen Streckenabschnitten.

Es gibt zwei Möglichkeiten, Tupel aus Relationen zu klassifizieren: Bei einer scharfen Klassifikation wird jedes Tupel genau einer Klasse zugewiesen, d.h. eine Strecke ist entweder unbedenklich befahrbar oder man muss zusätzlich Zeit einplanen usw. Bei einer unscharfen Klassifikation kann ein Tupel zu mehreren Klassen gehören und zwar jeweils zu einem gewissen Grad. Eine Strecke kann also zur Klasse der unbedenklich befahrbaren Strecken die Zugehörigkeit 0,7 haben und zur Klasse der Strecken, für die man zusätzliche Zeit einplanen muss, die Zugehörigkeit 0,3.

#### 2.2.1 Erweiterung des Schemas durch Kontexte

Im Kontextmodell wird jedem Attribut  $A_i$ , das auf einem Wertebereich  $D(A_i)$  definiert ist, ein Kontext  $K(A_i)$  zugeordnet. Ein Kontext  $K(A_i)$  ist eine Einteilung des Wertebereichs  $D(A_i)$  von  $A_i$  in Äquivalenzklassen. Diese Einteilung stellt eine Partition des Wertebereichs dar. Das relationale Schema mit den Attributen  $A = (A_1, A_2, \dots, A_n)$  wird also durch die Kontexte  $K = (K_1(A_1), K_2(A_2), \dots, K_n(A_n))$  erweitert.

**Beispiel:**

Gegeben sei ein Datenbankschema mit den Attributen

$$A = (\textit{Autobahn}, \textit{Abschnitt}, \textit{Stauhäufigkeit}, \textit{Staulänge})$$

und den entsprechenden Kontexten

$$K = (K(\textit{Autobahn}), K(\textit{Abschnitt}), K(\textit{Stauhäufigkeit}), K(\textit{Staulänge}))$$

Der Definitionsbereich des Attributs „Stauhäufigkeit“ sei

$$D(\textit{Stauhäufigkeit}) = \{\textit{häufig}, \textit{gelegentlich}, \textit{selten}, \textit{nie}\}$$

das Attribut „Staulänge“ beschreibe die durchschnittliche Staulänge in Kilometer. Die Kontexte sind dann z.B. die Partitionen

$$\begin{aligned} K(\textit{Stauhäufigkeit}) &= \{\{\textit{häufig}, \textit{gelegentlich}\}, \{\textit{selten}, \textit{nie}\}\} \\ K(\textit{Staulänge}) &= \{\{1, 2, 3, 4\}, \{5, 6, 7, 8, 9, 10\}\} \end{aligned}$$

Für die Kontexte  $K(\textit{Autobahn})$  und  $K(\textit{Abschnitt})$  gilt:

$$\begin{aligned} K(\textit{Autobahn}) &= \{D(\textit{Autobahn})\} \\ K(\textit{Abschnitt}) &= \{D(\textit{Abschnitt})\} \end{aligned}$$

Da die Attribute „Autobahn“ und „Abschnitt“ für die Klassifikation keine Rolle spielen, liegen sämtliche möglichen Attributwerte jeweils in der selben Äquivalenzklasse, die Kontexte dieser Attribute enthalten deswegen als einziges Element den Wertebereich des jeweiligen Attributs.

Durch die Definition der Kontexte, also die Partitionierung jedes einzelnen Wertebereichs, ergibt sich wiederum eine Einteilung der Gesamrelation in Äquivalenzklassen. Die Äquivalenzklassen der Relation sind die Kreuzprodukte der Äquivalenzklassen der Wertebereiche der Attribute:

$$\begin{aligned} C1 &= D(\textit{Autobahn}) \times D(\textit{Abschnitt}) \times \{\textit{häufig}, \textit{gelegentlich}\} \times \{1, 2, 3, 4\} \\ C2 &= D(\textit{Autobahn}) \times D(\textit{Abschnitt}) \times \{\textit{selten}, \textit{nie}\} \times \{1, 2, 3, 4\} \\ C3 &= D(\textit{Autobahn}) \times D(\textit{Abschnitt}) \times \{\textit{häufig}, \textit{gelegentlich}\} \times \{5, 6, 7, 8, 9, 10\} \\ C4 &= D(\textit{Autobahn}) \times D(\textit{Abschnitt}) \times \{\textit{selten}, \textit{nie}\} \times \{5, 6, 7, 8, 9, 10\} \end{aligned}$$

Den Klassen können nun zur Verdeutlichung der inhaltlichen Bedeutung noch Klassennamen zugeordnet werden, z.B. könnte man die Klasse C1 mit dem Namen „unbedenklich befahrbar“ versehen, dies erleichtert eine Interpretation der Klasseneinteilung. Die Einteilung der Relation in Klassen kann man sich am besten graphisch veranschaulichen:

10	<b>C3</b> weiträumig umfahren	<b>C4</b> unvorhersehbare Verzögerungen		
9				
8				
7				
6				
5	<b>C1</b> zusätzlich Zeit einplanen	<b>C2</b> unbedenklich befahrbar		
4				
3				
2				
1				
	häufig	gelegentlich	selten	nie

## 2.2.2 Kontextredundante Tupel

Zwei Tupel heißen kontextredundant, wenn sie in der selben Äquivalenzklasse der Relation liegen. Dies ist gleichbedeutend damit, dass sämtliche Tupelkomponenten jeweils in den selben Äquivalenzklassen der einzelnen Wertebereiche liegen. Die Beispielrelation von oben enthalte nun folgende Tupel:

Autobahn	Abschnitt	Stauhäufigkeit	Staulänge
A5	A5-1	gelegentlich	3
A5	A5-2	selten	2
A5	A5-3	nie	4
A8	A8-1	häufig	7
A8	A8-2	gelegentlich	8
A81	A81-1	nie	3
A81	A81-2	selten	5

Ein Tupel gehört zu einer bestimmten Klasse, wenn sein Merkmalsvektor im jeweiligen Teilgebiet der Relation liegt. Es ergibt sich also folgende Zuordnung der Tupel zu Klassen (Stauhäufigkeit und Staulänge sind der Übersichtlichkeit halber weggelassen):

Autobahn	Abschnitt	Klasse
A5	1	C1
A5	2	C2
A5	3	C2
A8	1	C3
A8	2	C3
A81	1	C2
A81	2	C4

Wie man sieht, existieren zahlreiche kontextredundante Tupel, z.B.  $\{A5, 2, \text{selten}, 2\}$  und  $\{A5, 3, \text{nie}, 4\}$ , die beide in C2 liegen, oder  $\{A8, 1, \text{häufig}, 7\}$  und  $\{A8, 2, \text{gelegentlich}, 8\}$ , welche beide in C3 liegen. Diese lassen sich beseitigen, indem man für jede Klasse alle kontextredundanten Tupel zu einem einzigen Tupel zusammenfasst. Dies geschieht, indem aus den Werten der Attribute aller kontextredundanten Tupel einer Klasse Mengen gebildet werden. Die resultierende kontextredundanzfreie Relation enthält nur noch ein Tupel für jede Klasse:

Autobahn	Abschnitt	Stauhäufigkeit	Staulänge	Klasse
{A5}	{A5-1}	{gelegentlich}	{3}	C1
{A5, A81}	{A5-2, A5-3, A81-1}	{selten, nie}	{2, 3, 4}	C2
{A8}	{A8-1, A8-2}	{häufig, gelegentlich}	{7, 8}	C3
{A81}	{A81-2}	{selten}	{5}	C4

## 2.2.3 Unschärfe Zuordnung

Bei der Zuordnung der einzelnen Strecken zu den verschiedenen Klassen im obigen Beispiel handelt es sich um eine scharfe Zuordnung, d.h. jede Strecke wird genau einer Klasse zugeordnet. Dies ist jedoch nicht der Fall, wenn man die Zuordnung von den Abschnitten unabhängig vornimmt:

Autobahn	Stauhäufigkeit	Staulänge	Klasse
{A5}	{gelegentlich}	{3}	C1
{A5, A81}	{selten, nie}	{2, 3, 4}	C2
{A8}	{häufig, gelegentlich}	{7, 8}	C3
{A81}	{selten}	{5}	C4

Man erhält so eine unscharfe Zuordnung, denn die A5 liegt sowohl in der Klasse C1 als auch in der Klasse C2 und die A81 liegt in den Klassen C2 und C4.

Die Zuordnung ein und des selben Objekts zu mehreren Klassen ist einerseits leicht nachvollziehbar, denn eine Autobahn als Ganzes weist eben auf ihren einzelnen Abschnitten ein unterschiedliches Stauverhalten auf und kann deswegen nicht einer einzigen Klasse zugeordnet werden. Andererseits bringt eine unscharfe Zuordnung auch einige Schwierigkeiten mit sich. Was fängt man z.B. mit einem Objekt an, welches zwei sehr unterschiedlichen Klassen zugeordnet wird? Was ist mit einem Objekt, das in allen Klassen liegt? Hier wäre es sinnvoll, Aussagen darüber machen zu können, wie stark die Zugehörigkeit des Objekts zu jeder Klasse ist. Dies wird durch das Konzept der linguistischen Variablen ermöglicht.

## 2.3 Linguistische Variable

Linguistische Variablen sind ein Konzept zur semantischen Beschreibung unscharfer Klassen. Sie erlauben eine Zuordnung von verbalen Begriffen zu Äquivalenzklassen. Dies kommt dem menschlichen Verständnis entgegen und erleichtert so den Umgang mit unscharfen Zuordnungen.

Der Wert einer linguistischen Variablen ist ein sprachlicher Begriff, der Term genannt wird. Ein Term beschreibt eine unscharfe Menge über dem Wertebereich des entsprechenden Attributes der Relation. Betrachten wir als Beispiel eine linguistische Variable „Staulänge“, die als Werte die Terme „kurz“ und „lang“ hat. „kurz“ und „lang“ beschreiben jeweils eine unscharfe Menge über dem Wertebereich des zugehörigen Attributes,  $D(\text{Staulänge})$ .

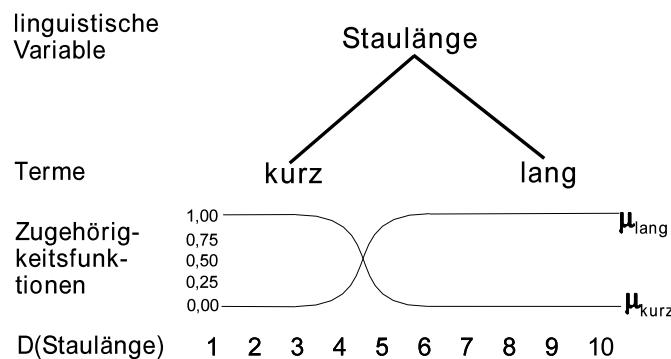


Abbildung 1: linguistische Variable

Jede unscharfe Menge ist durch eine Zugehörigkeitsfunktion  $\mu$  auf dem Wertebereichbereich des zur linguistischen Variablen korrespondierenden Attributes definiert. Die Zugehörigkeitsfunktion gibt für jeden Wert den Grad der Zugehörigkeit zu einer bestimmten unscharfen Menge an. Ein Stau der Länge 5 Kilometer ist also nicht wie bei scharfen Klassen entweder kurz oder lang, sondern kann z.B. zum Grad 0,3 kurz und zum Grad 0,7 lang sein.

### 2.3.1 Klassenzuordnung durch Aggregation

Mit Hilfe linguistischer Variablen lässt sich der Wertebereich jedes einzelnen Attributes der Relation unscharf partitionieren, so könnte man z.B. zum Attribut „Stauhäufigkeit“ eine linguistische Variable mit den Termen „hoch“ und „niedrig“ einführen. Dadurch ergibt sich eine Unterteilung der Relation in unscharfe Klassen. Der Grad der Zugehörigkeit eines Tupels zu einer dieser unscharfen Klassen lässt sich aus den Zugehörigkeitsfunktionen der einzelnen Attribute berechnen. Dies geschieht durch Aggregatfunktionen, die mit speziellen Operatoren aus der Fuzzy-Logic berechnet werden. Aus den Werten der Zugehörigkeitsfunktionen der zu den einzelnen Attributen gehörenden linguistischen Variablen berechnen die Aggregatfunktionen den Grad der Zugehörigkeit zu den Klassen. Das Ergebnis der Aggregation könnte z.B. so aussehen:

Autobahn	Stauhäufigkeit	Staulänge	Klasse
{A5: 0,3}	hoch	kurz	C1
{A5: 0,7, A81: 0,4}	niedrig	kurz	C2
{A8: 1,0}	hoch	lang	C3
{A81: 0,6}	niedrig	lang	C4

Die Zahlen hinter den Autobahnbezeichnern sind das Ergebnis der Aggregatfunktion und geben an, wie stark die Autobahn zur jeweiligen Klasse gehört. Die A5 beispielsweise gehört zu den Klassen C1 und C2, jedoch ist die Zugehörigkeit zu C2 mit 0,7 stärker als die zu C1 mit nur 0,3.

## 2.4 Die Anfragesprache fCQL

fCQL (fuzzy Classification Query Language) ist eine Klassifikationssprache, mit Hilfe derer unscharfe Klassen, wie sie oben beschrieben werden, definiert und Anfragen an die darunterliegende Datenbank gerichtet werden können. Die Datenbank selbst ist eine gewöhnliche relationale Datenbank mit scharfen Daten, sie wird lediglich wie beschrieben durch Kontexte, Klassen und Zugehörigkeitsfunktionen erweitert. fCQL-Anfragen werden in SQL-Anfragen übersetzt, welche auf den scharfen Daten arbeiten.

Die Sprache fCQL wurde an der Universität Fribourg entwickelt<sup>1</sup>, es existiert ein Compiler für die Umsetzung nach SQL, der gerade an Datenbeständen aus der Industrie getestet wird.

Klassifikationsanfragen in fCQL arbeiten nicht mit Merkmalswerten, sondern vollständig auf linguistischer Ebene. Das hat den Vorteil, dass der Anwender keine scharfen Zielwerte und Kontexte kennen muss, sondern lediglich deren verbale Beschreibung. Die Definition der Kontexte und linguistischen Variablen kann vom Datenbankadministrator oder von einem Experten vorgenommen werden.

Eine Klassifikationsanfrage hat folgende Form:

```

classify Objekt
from Tabelle
with Klassifikationsbedingung

```

Im konkreten Beispiel könnte eine Anfrage also so aussehen:

```

classify Autobahn from Staurelation with class is unbedenklich_befahrbar

```

Es ist auch möglich, nicht nach einer bestimmten vorgegebenen Klasse, sondern nach den Werten linguistischer Variablen zu selektieren:

```

classify Autobahn from Staurelation with Staulänge is kurz and Stauhäufigkeit is niedrig

```

## 3 Schemaerweiterung durch eingebettete Subrelationen

Bisher wurden Erweiterungen des relationalen Modells betrachtet, die es ermöglichen, scharfe Daten unscharf zu klassifizieren. Im Folgenden soll es um Erweiterungen gehen, die es erlauben, Daten zu erfassen, die nicht mit dem definierten Schema vereinbar sind. Vorgestellt wurde dieser Ansatz in [3].

### 3.1 Motivation

Bei der Definition eines relationalen Schemas wird festgelegt, was die Attribute einer Relation sind und aus welchem Wertebereich ihre Ausprägungen stammen. Diese Festlegung ist notwendig, um eine Strukturierung zu gewährleisten, welche ein Arbeiten mit den Daten erst ermöglicht. Dabei wird eine idealisierte Vorstellung der Welt zugrunde gelegt, die zwar in den meisten Fällen, aber nicht immer mit der Realität übereinstimmt. So mögen die meisten Datensätze in das Schema passen; jedoch gibt es immer wieder Ausnahmen, an die entweder bei der Definition des Schemas nicht gedacht wurde, oder die zu selten sind, um ein Schema, das ihnen gerecht wird, zu rechtfertigen. Der Existenz dieser Ausnahmen wird jedoch im relationalen Modell keine Rechnung getragen.

<sup>1</sup><http://diuf.unifr.ch/is/research/fcql/index.php>

Beispiel: Ein System zur Verkehrsüberwachung erfasst die Kennzeichen aller vorüberfahrenden Fahrzeuge und trägt diese in eine Datenbank ein. Ist im Schema festgelegt, dass Autokennzeichen aus zwei Buchstaben- und einer Zahlenkombination bestehen (was sinnvoll ist, wenn die Kennzeichen z.B. nach Landkreisen gruppiert werden sollen), entstehen Schwierigkeiten, wenn ein ausländisches Kennzeichen erfasst wird, das nur aus Zahlen besteht, da dieses nicht in das Schema passt. Behelfen könnte man sich allenfalls, indem man Nullwerte für die Buchstabengruppen und einen Vermerk „ausländisches Kennzeichen“ in ein Kommentarfeld einträgt.

### 3.2 Exceptional Conditions

Eine solche Ausnahme, die von der Schemadefinition abweicht, genauer die Art einer solchen Abweichung wird als *Exceptional Condition* bezeichnet. Die genaue Definition einer Exceptional Condition umfasst die folgenden vier Punkte:

1. Daten, die in dieser Art von der Schemadefinition abweichen sind zu erwarten, entweder im Voraus oder zumindest im Nachhinein.
2. Die abweichenden Daten können auf irgendeine Art repräsentiert werden.
3. Die Art der Abweichung tritt nicht oft genug auf, um eine Änderung des Schemas, die die abweichenden Daten respektieren würde, zu rechtfertigen.
4. Die Schemadefinition kann lokal erweitert werden, um eine Repräsentation der abweichenden Daten zu erlauben.

Im Beispiel mit den Autokennzeichen könnten Exceptional Conditions z.B. sein:

- Kennzeichen, die nicht in das Muster „Buchstaben–Buchstaben–Zahlen“ passen
- Kennzeichen, bei denen eine zusätzliche Erklärung notwendig ist wie „Saisonkennzeichen, gilt nur von März bis Oktober“
- kein Kennzeichen verfügbar, sondern ein semantisch gleichwertiges Datum, wie z.B. die Versicherungsplakette bei einem Moped
- Unterschiedliche Kennzeichen auf der Vorder- und Rückseite

All diese Beispieldaten liegen in der Intention des Schemas, nämlich ein Fahrzeug mit einem Kennzeichen zu identifizieren, passen aber nicht in den Wertebereich der Schemadefinition.

#### 3.2.1 eingebettete Attribute

Fahrzeuge, deren Kennzeichen auf der Vorder- und Rückseite sich unterscheiden, könnten durch die lokale Einführung eines zusätzlichen Attributs „Seite“ repräsentiert werden:

Farzeugtyp	Kennzeichen		Farbe
PKW	Seite	Kennzeichen	rot
	Vorderseite	KA-XY-42	
	Rückseite	KA-XY-23	

Das zusätzliche Attribut wird eingebettetes Attribut (e-Attribut) genannt, das eigentliche Attribut, hier „Kennzeichen“ nennt man Schemaattribut (s-Attribut). Auch die anderen Beispiele für Exceptional Conditions lassen sich durch die Einführung von e-Attributen ausdrücken.

### 3.2.2 Schemaerweiterung zur Repräsentation von Exceptional Conditions

Um auch Daten mit Exceptional Conditions in eine Relation einfügen zu können, muss das relationale Modell erweitert werden. Eine Erweiterung zur Repräsentation von Daten, die Exceptional Conditions enthalten, sollte folgendes gewährleisten:

1. Zulassen von Exceptional Conditions unter Beibehaltung des idealen Schemas
2. Einfügen oder Löschen von Tupeln, die Exceptional Conditions enthalten, ohne Änderung des idealen Schemas
3. Ermöglichen von Anfragen nach Tupeln, die ein bestimmtes e-Attribut besitzen
4. Ermöglichen von Anfragen nach Tupeln, deren e-Attribut einen bestimmten Wert hat

Die Frage ist nun, wie das Schema erweitert werden kann, um die e-Attribute zu berücksichtigen. Die einfachste Möglichkeit wäre es, die Informationen der Exceptional Conditions als Zeichenkette in einem Kommentarfeld abzuspeichern:

Fahrzeugtyp	Kennzeichen	Farbe	Kommentar
PKW	KA-XY-42	rot	Kennzeichen vorne
PKW	KA-XY-23	rot	Kennzeichen hinten
LKW	KA-YZ-4711	gelb	mit grünen Streifen
PKW	KA-XZ-5	blau	Saisonkennzeichen März bis Oktober

Dies erfüllt zwar die Forderungen (1) und (2), nicht jedoch (3), da sämtliche e-Attribute zu einem einzigen Attribut zusammengefasst werden. Auch (4) ist nicht erfüllt, z.B. eine Anfrage nach allen Fahrzeugen mit Saisonkennzeichen, die im Juli gültig sind, wäre nicht ohne weiteres möglich.

Eine andere Möglichkeit wäre natürlich, sämtliche vorkommenden e-Attribute der Relation zuzuschlagen, was dann z.B. folgendermaßen aussehen würde:

Fahrzeugtyp	Kennzeichen	Seite	Farbe	Muster	Saison von	Saison bis
PKW	KA-XY-42	Vorderseite	rot			
PKW	KA-XY-23	Rückseite	rot			
LKW	KA-YZ-4711		gelb	grüne Streifen		
PKW	KA-XZ-5		blau		März	Oktober

Dies würde zwar Anfragen nach e-Attributen möglich machen, Forderungen (3) und (4) sind also erfüllt, jedoch liegt nicht mehr das ideale Schema vor, so dass Forderung (1) nicht erfüllt ist; es existieren Attribute, die nur in wenigen Ausnahmefällen mit Werten belegt sind, was einen erheblichen und nicht gerechtfertigten Mehraufwand verursacht. Außerdem muss beim Einfügen von Tupeln mit Exceptional Conditions, die bisher noch nicht vorhandene e-Attribute erfordern, das Schema abgeändert werden, was Forderung (2) verletzt.

Eine Lösung, welche alle vier Bedingungen erfüllt, ist die Erweiterung des Schemas durch die Einbettung von Subrelationen. Bei einem Tupel, das eine Exceptional Condition enthält, steht anstelle des Merkmalswertes eine Subrelation, die das s-Attribut und alle benötigten e-Attribute enthält:

Fahrzeugtyp	Kennzeichen			Farbe	
PKW	KA-AB-1			rot	
PKW	Kennzeichen	Seite		rot	
	KA-XY-42	Vorderseite			
	KA-XY-23	Rückseite			
LKW	KA-YZ-4711			Farbe	Muster
				gelb	grüne Streifen
PKW	Kennzeichen	Saison von	Saison bis	blau	
	KA-XZ-5	März	Oktober		
Moped	Kennzeichen	Versicherungs-Nr		silbergrau	
		ABC-123			

Das ideale Schema wird also beibehalten, auch bei Einfüge- und Löschooperationen (Bedingungen (1) und (2)). Anfragen, die den Bedingungen (3) und (4) gerecht werden, werden in Abschnitt 3.3 behandelt.

Natürlich ist diese Beispieltabelle hier kein typischer Fall, Exceptional Conditions zeichnen sich ja gerade dadurch aus, dass sie Ausnahmen darstellen. Tatsächlich wäre also der Großteil der Datensätze so beschaffen, dass er in das ideale Schema passt, die meisten Einträge in der Tabelle wären also von der Form desjenigen in der ersten Tabellenzeile.

### 3.2.3 Einteilung von Exceptional Conditions

Es gibt mehrere Arten von Exceptional Conditions, die sich darin unterscheiden, ob das s-Attribut mit einem Wert belegt ist und welcher Zusammenhang zwischen s- und e-Attribut besteht. Exceptional Conditions können in drei Kategorien eingeteilt werden:

1. Ein Wert für das s-Attribut ist vorhanden und passt in dessen Wertebereich, wird aber durch zusätzliche Informationen in e-Attributen ergänzt. Eine solche Exceptional Condition wird auch als *Exceptional Comment* bezeichnet. Im obigen Beispiel fallen der LKW, dessen gelbe Farbe durch das grüngestreifte Muster ergänzt wird, und der PKW mit den Saisonkennzeichen in diese Kategorie.
2. Für das s-Attribut ist kein oder kein passender Wert verfügbar, eine semantisch äquivalente Information ist in einem e-Attribut enthalten. Dies bezeichnet man als *Exceptional Alternative*. Im Beispiel oben ist dies beim Moped der Fall. Bei Exceptional Alternatives wird dem s-Attribut in der Subrelation ein Nullwert zugewiesen.
3. Für das s-Attribut sind mehrere Werte verfügbar, die von einem oder mehreren e-Attributen abhängen. Diese Art von Exceptional Condition heißt *Exceptional Constraint*. Ein Beispiel hierfür ist der PKW, dessen Kennzeichen von der Fahrzeugseite abhängt. Diese Form der Exceptional Condition ist die einzige, bei der die Subrelation mehr als ein Tupel enthält.

## 3.3 Anfragen an die erweiterte Relation

Grundsätzlich lassen sich zwei Arten von Anfragen an die erweiterte Relation unterscheiden: Anfragen, die auf dem idealen Schema basieren, sich also nur auf die s-Attribute beziehen und Anfragen, die auf dem erweiterten Schema basieren und sich auf s- und e-Attribute beziehen.

Der Einfachheit halber sollen im Folgenden nur Exceptional Constraints betrachtet werden.

### 3.3.1 Anfragen auf der Grundlage des idealen Schemas

Bei Anfragen, die auf dem idealen Schema beruhen, sind zum Zeitpunkt der Anfrage lediglich die Attribute des idealen Schemas bekannt. Daher wird als Ergebnis der Anfrage eine Relation erwartet, die auch nur diese Attribute enthält. Berücksichtigt man, dass Exceptional Conditions eine Ausnahme darstellen und deswegen fast alle Tupel der Relation, auf die sich die Anfrage bezieht, nur s-Attribute enthalten, wird diese Erwartung in den meisten Fällen auch erfüllt sein.

In einigen Fällen jedoch, werden als Ergebnis Tupel zurückgeliefert, die eingebettete Attribute besitzen. Die Antwort auf die Anfrage entspricht nicht mehr dem, was erwartet wird, man spricht daher von einer „Big Surprise“; das Anfrageergebnis fällt komplexer aus als erwartet.

Richtet man an die obige Beispielrelation eine Anfrage nach den Kennzeichen aller roten Fahrzeuge, so erwartet man als Ergebnis eine Menge von Kennzeichen. Zurückgegeben wird jedoch:

Kennzeichen	
KA-AB-1	
Kennzeichen	Seite
KA-XY-42	Vorderseite
KA-XY-23	Rückseite

Die Antwortrelation enthält die Informationen, nach denen gefragt wurde und zusätzlich Exceptional Constraints, die die Gültigkeit der zurückgegebenen Informationen einschränken und von deren Existenz die anfragende Instanz zuvor noch nichts wusste.

Als Reaktion auf eine „Big Surprise“ muss die anfragende Instanz die Anfrage unter Berücksichtigung der e-Attribute, deren Kenntnis sie durch die erhaltene Antwort erlangt hat, verfeinern. Für jedes Tupel mit Exceptional Condition gilt es aus den Tupeln der Subrelation eines zu selektieren, denn für das s-Attribut des Tupels mit Exceptional Condition muss einer der Werte, die dieses s-Attribut in der Subrelation hat, ausgewählt werden. Beispielsweise könnte die verfeinerte Anfrage aus der Subrelation nur die Kennzeichen auf der Vorderseite selektieren, so dass als Endergebnis dann folgendes entsteht:

<u>Kennzeichen</u>
KA-AB-1
KA-XY-42

### 3.3.2 Anfragen unter Kenntnis der erweiterten Attribute

Anders als im vorangegangenen Abschnitt sollen nun zum Zeitpunkt der Anfrage sämtliche e-Attribute, die in eingebetteten Relationen vorkommen, bekannt sein. Die Anfrage kann deswegen neben s-Attributen auch e-Attribute enthalten. Es könnte also z.B. nach allen Tupeln gefragt werden, bei denen das Attribut „Seite“ den Wert „Vorderseite“ hat, wenn man sich für die vorderen Kennzeichen der Fahrzeuge interessiert. Problematisch hierbei ist, dass fast alle Tupel gar kein Attribut „Seite“ besitzen. Da die durch sie beschriebenen Fahrzeuge aber sehr wohl vordere Kennzeichen haben, sollten auch solche Tupel zurückgegeben werden.

Um dies zu gewährleisten, wird die Relation so umgewandelt, dass jedes Tupel über alle Attribute verfügt, auch über die e-Attribute anderer Tupel mit Exceptional Conditions. Hierzu wird eine spezielle Art von Nullwert benötigt, der die Bedeutung „jeder beliebige Wert“ hat. Er wird als Wert für ein e-Attribut eingesetzt, das ein Tupel in der ursprünglichen Relation nicht besaß. Die Art von Nullwert soll durch einen Stern (\*) symbolisiert werden.

Aus der Relation

Fahrzeugtyp	Kennzeichen		Farbe
PKW	KA-AB-1		rot
PKW	<b>Kennzeichen</b>	<b>Seite</b>	rot
	KA-XY-42	Vorderseite	
	KA-XY-23	Rückseite	
Motorrad	KA-CD-7		weiß

wird also

Fahrzeugtyp	Kennzeichen	Seite	Farbe
PKW	KA-AB-1	*	rot
PKW	KA-XY-42	Vorderseite	rot
PKW	KA-XY-23	Rückseite	rot
Motorrad	KA-CD-7	*	weiß

Die Anfrage nach allen Kennzeichen auf der Vorderseite ergäbe dann:

<u>Kennzeichen</u>
KA-AB-1
KA-XY-42
KA-CD-7

die nach denen auf der Rückseite:

<u>Kennzeichen</u>
KA-AB-1
KA-XY-23
KA-CD-7

Da nicht alle Exceptional Conditions dem selben Schema gehorchen, kann es in einer Relation eine Vielzahl unterschiedlicher e-Attribute geben. Für jedes dieser e-Attribute benötigt man eine eigene Spalte, die fast nur „\*“ als Einträge hat. Dadurch erhält man eine sehr dünn besetzte Relation, was zu erheblichem Overhead führt.

Eine Lösung dieses Problems wäre, die Spalten für die e-Attribute nicht explizit zu generieren, sondern bei Anfragen, die Bedingungen für e-Attribute enthalten, immer auch alle Tupel zurückgeben, die dieses e-Attribut nicht besitzen, somit würde bei diesen Tupeln als Wert für das e-Attribut implizit „\*“ angenommen.

## 4 Vergleich der beiden Ansätze

### 4.1 Unterschiede

Die beiden vorgestellten Ansätze zur Erweiterung relationaler Schemata sind von recht unterschiedlicher Natur, da sie unterschiedliche Ziele verfolgen: der erste Ansatz will keine Unschärfe in den Daten an sich, sondern lediglich eine Einteilung scharfer Daten nach unscharfen Kriterien, während der zweite Ansatz zum Ziel hat, Daten, die nicht in ein vorgegebenes relationales Schema passen, durch die Einführung von Unschärfe doch in die Relation einzufügen.

Daher ist auch die Weise, in der die Relationen erweitert werden bei beiden Ansätzen sehr verschieden: Der zuerst vorgestellte Ansatz behält die gewohnte Art der Schemadefinition bei, fügt aber noch die Definition von Kontexten und darauf aufbauend die von Klassen hinzu. Der andere Ansatz hingegen erweitert die Schemadefinition um den zusätzlichen Attributwert „Subrelation“ und um e-Attribute.

Auch was Anfragen an die erweiterte Relation betrifft, unterscheiden sich die beiden Ansätze: Beim ersten wird eine völlig neue Anfragesprache definiert, während für Relationen mit eingebetteten Subrelationen Anfragen mit existierende Anfragesprachen formuliert werden können.

Neben den inhaltlichen Unterschieden gibt es weitere: der erste Ansatz ist sehr aktuell; die Anfragesprache fCQL befindet sich zur Zeit in der Erprobung. Inwiefern der Ansatz sich durchsetzen wird, ist ungewiss. Der zweite Ansatz wurde bereits Ende der achtziger Jahre veröffentlicht, hatte aber so gut wie keinen Einfluss.

### 4.2 Gemeinsamkeiten

Neben der im vorangegangenen Abschnitt beschriebenen Unterschiede besitzen die beiden Ansätze jedoch eine große Gemeinsamkeit, die sich am besten mit dem Begriff „Abwärtskompatibilität zum eigentlichen relationalen Modell“ beschreiben lässt, d.h. mit beiden Ansätzen ist es möglich, vorhandene scharfe Datenbestände unmodifiziert beizubehalten. Zur unscharfen Klassifikation müssen diese nur um die Definition der Kontexte und der linguistischen Variablen erweitert werden; nutzt man den Ansatz der Einbettung von Subrelationen, ist lediglich eine neue Art von Wert, den man einem Attribut zuweisen kann, nämlich die Subrelation, gegeben. Ein vorhandener Datenbestand kann also relativ leicht durch „Unschärfe“ erweitert werden. Dies ist insbesondere deswegen wichtig, weil bereits zahl- und umfangreiche scharfe relationale Datenbestände vorhanden sind.

### 4.3 Kombination der beiden Ansätze

Ob der unterschiedlichen Zielsetzungen der beiden Ansätze ließe sich höchstens noch überlegen, ob sie vielleicht kombinierbar sind und sich gegenseitig ergänzen könnten. Es dürfte jedoch sehr schwierig werden, für ein durch Subrelationen erweitertes relationales Modell Kontexte zu definieren, denn ein Kontext stellt ja eine Partitionierung des Wertebereichs eines Attributs dar. Wie jedoch sollte ein Wertebereich partitioniert werden, der Relationen enthalten kann? Es wäre allenfalls möglich, Tupel mit Exceptional Conditions, die ja den Ausnahmefall darstellen, außen vorzulassen und lediglich die in das ideale Schema passenden Tupel zu klassifizieren. Synergieeffekte ließen sich aber auf diese Weise keine erzeugen, zusätzlich handelte man sich noch unklassifizierbare Tupel ein bzw. benötigte ein gesondertes Klassifikationsverfahren für Tupel mit Exceptional Conditions.

## Literatur

- [1] Andreas Meier, Christian Mezger, Nicolas Werro, Günter Schindler: *Zur unscharfen Klassifikation von Datenbanken mit fCQL*
- [2] Andreas Meier, Christian Savary, Günter Schindler, Yauheni Veryha: *Database Schema with Fuzzy Classification and Classification Query Language*
- [3] Howard M. Dreizen, Shi-Kuo Chang: *Imprecise Schema: A Rationale for Relations with Embedded Subrelations*