

<b>Veranstaltung</b>	Neuer Lehrplan AINF
<b>Titel</b>	Arbeitsunterlagen, 3. Jahrgang, Einheit 5, DB
<b>Vortragender</b>	DI Schöndorfer

Anmerkung : die nachfolgenden Arbeitsunterlagen stellen einen Auszug aus dem **ADIM** Band 66 dar.

Als erstes Modell zur Realisierung soll nun das Relationenmodell besprochen werden. Es gilt heute als das Standarddatenmodell.

Anmerkung: Im klassischen Datenbankentwurf wird manchmal vor dem Relationenmodell ein Modell zur Datenmodellierung vorgestellt. Aus didaktischen Gründen wird jedoch im vorliegenden Skript zunächst ein Datenmodell vorgestellt. Dies findet darin seine Begründung, dass Modelle zur Datenmodellierung oftmals abstrakt sind – und sich somit ein konkreter Einstieg besser eignet. Der interessierte Leser sei für eine einsprechendes Modell auf Kapitel 5 verwiesen.

Das Relationenmodell wurde von Codd 1970 eingeführt. Es hat sich mittlerweile als das am weitesten verbreitete Datenbankmodell in der Praxis etabliert. Innerhalb der Forschung ist es schon seit geraumer Zeit anerkannt, da die bestechende Einfachheit und Exaktheit des Relationenmodells weitreichende Ergebnisse in vielen Gebieten der Datenbankforschung ermöglicht.

## 4.1 Schemata und Attribute

Im Relationenmodell werden Objekttypen der zu modellierenden Anwendungswelt durch Relationenschemata beschrieben.

### Definition 4.1.

Relationenschemata beschreiben Objekttypen der zu modellierenden Anwendungswelt. Relationenschemata bestehen aus Attributen.

Diese bestehen aus einer Menge von Attributen, die die gemeinsamen Eigenschaften der Objekte repräsentieren, die zu einem darstellbaren Objekttyp gehören. Attributen werden Wertbereiche (auch Domänen) zugeordnet, die in der Praxis meist Standard – Datentypen wie Integer oder String sind.

### Definition 4.2.

Eine Datenbank besteht (vereinfacht) aus einer Menge von Relationenschemata.

Eine Relation ist nun in ihrer einfachsten Beschreibung eine Teilmenge des kartesischen Produkts über den Wertebereichen der Attribute des Relationenschemas. Sie stellt die zu einem Relationenschema passenden und aktuell vorhandenen Daten dar – die Instanz zu diesem Schema. Die in der Datenbank aktuell vorhandenen Relationen zu einem im Datenbankschema definierten Relationenschema heißt Basisrelation.

### Definition 4.3.

Die in einer Datenbank vorhandenen Relationen heißen Basisrelationen.

### Definition 4.4.

Die Menge aller Basisrelationen heißt Datenbank.

### Definition 4.5.

Ein Element einer Relation wird als Tupel bezeichnet.

Um die Relation von ihrem Relationenschema zu unterscheiden, bezeichnen wir sie mit einem kleinen  $r$ . Eine Relation zum Relationenschema  $R$  wird dann mit  $r(R)$  gekennzeichnet. Die eben angesprochene Zweiteilung in Schema und Instanz findet sich auch bei den meisten neueren Datenbankmodellen Anwendung, sie wird nur langsam durch die Einführung von Metatypen und Metaklassen überwunden.

Die Repräsentanz des Relationenschemas erfolgt typischer Weise in einer Tabelle:

Nummer	Name	Ort	Straße
1	Schulze	Leipzig	Sasstr. 6
2	Meier	Bremen	Hauptstr. 123
3	Müller	Frankfurt	Dorfstr. 2

Diagramm zur Darstellung einer Tabelle als Relationenschema:

- Die Spaltenüberschriften (Nummer, Name, Ort, Straße) sind als **Spalte** beschriftet.
- Die gesamte Tabelle ist als **Datensatz TUPEL** beschriftet.
- Ein einzelnes Element in der Tabelle (z.B. die Zeile mit Nummer 1) ist als **Datenfeld** beschriftet.

Abb.: 4.1.: Repräsentanz des Relationenschemas als Tabelle

Es ist zu beachten, dass an eine einzelnen Tabellenpositionen nur Datenwerte, aber keine weiteren Tabellen stehen können. Man sagt, die dargestellten Datenwerte sind „atomar“, also nicht weiter zerlegbar. Man bezeichnet eine Darstellung eines Relationenmodells durch atomare Werte als eine Darstellung in der erster Normalform (1. NF).

---

---

*Definition 4.6.*

Datenwerte werden als atomar bezeichnet, wenn sie nicht weiter zerlegbar sind.

---

---

---

---

*Definition 4.7.*

Eine Relation, die nur aus atomaren Werten besteht, wird als Relation in 1. Normalform bezeichnet.

---

---

---

---

*Definition 4.8.*

Eine Relation  $r$  auf einem Relationenschema  $R$ , definiert als  $r(R)$ , ist eine endliche Menge von Abbildungen  $\{t_1, \dots, t_m\}$  von  $R$  nach  $D$ , wobei für jede Abbildung  $t_j \in r$ ,  $t_j(A_i)$  aus  $D_i$  stammt,  $1 \leq j \leq m$ . Die Abbildungen werden Tupel genannt.

---

---

Fassen wir daher nochmals die wesentlichen Eigenschaften des relationalen Datenbankmodells zusammen (nach Codd, E.F. „A Relational Modell of Data for Large Shared Data Banks“, Communications of the ACM, June 1970)

- ∅ Die Daten werden einheitlich durch Werte repräsentiert, die in Form von Tabellen dargestellt werden.
- ∅ Der Benutzer sieht keine speziellen Verweisstrukturen zwischen den Tabellen.
- ∅ Es gibt Operationen zur Auswahl von Tabellenzeilen (Selektion), Tabellenspalten (Projektion) sowie zur Verbindung ("join") von Tabelleneinträgen. Keine dieser Operatoren ist jedoch auf Kontrollstrukturen angewiesen oder durch vordefinierte Zugriffsstrukturen beschränkt. Diese Operationen werden über eine Datenmanipulationssprache (beispielsweise SQL) realisiert, die auf Daten zugreifen, Daten einfügen, löschen, korrigieren und Anfragen beantworten kann.

Basisbestandteil von Relationen sind die Attribute. Jedes Attribut kann dabei verschiedene Wertebereiche, sogenannte Domänen annehmen.

---

---

*Definition 4.9*

Sei  $U$  eine nichtleere, endliche Menge.  $U$  wird als Universum der Attribute bezeichnet. Der Wertebereich von Attributen wird als Domäne bezeichnet. Eine Menge  $R \subseteq U$  heißt Relationenschema.

---

---

---

---

*Definition 4.10*

Eine Menge von Relationenschemata  $S := \{R_1, \dots, R_p\}$  mit  $p \in \mathbb{N}$  heißt Datenbankschema. Ein Datenbankwert - oder kurz Datenbank über einem Datenbankschema  $S$  ist eine Menge von Relationen

---

---

Da eine Relation eine Menge ist, kann es keine mehrfach gleichen Tupel mit identischen Werten für alle Attribute eines Relationenschemas in einer Relation geben. Es muss deshalb ausgezeichnete Attribute geben, welche die Tupel eindeutig identifizieren. Würde es entsprechende Tupel geben, so würde dadurch jedes Tupel einer Relation eindeutig identifiziert werden. Derartige Attribute werden als Schlüssel bezeichnet.

---

---

*Definition 4.11.*

Ein Attribute, welche Tupel einer Relation eindeutig identifizieren, werden als Schlüssel bezeichnet.

---

---

Ein Schlüssel kann dabei aus einem einzelnen Attribut bestehen – oder aus mehreren Attributen zusammengesetzt werden. Im ersten Fall spricht man von einem Primärschlüssel, sonst von einem zusammengesetzten oder partiellen Schlüssel.

---

---

*Definition 4.12.*

Setzt sich ein Schlüssel aus mehreren Attributen zusammen, so bezeichnet man die Teilattribute als partielle Schlüsselattribute.

---

---

Ein Fremdschlüssel ist eine Attributliste  $X$  in einem Relationenschema  $R_1$ , wenn in einem Relationenschema  $R_2$  eine kompatible Attributliste  $Y$  Primärschlüssel ist und die Attributwerte zu  $X$  in der Relation  $r_1(R_1)$  auch in den entsprechenden Spalten  $Y$  der Relation  $r_2(R_2)$  enthalten sind.

Im Relationenmodell werden Schlüsselattribute unterstrichen dargestellt.

## **4.2 Normalisierungstheorie**

Die Darstellung der Daten in einer relationalen Datenbank folgt speziellen Vorschriften. Eine davon haben wir bereits kennen gelernt: die 1. Normalform. Diese Vorschriften (Theorie des relationalen Datenbankentwurfs) bezeichnet man üblicherweise als Normalisierungstheorie. Normalisieren bedeutet dabei: Darstellung des logischen Schemas einer relationalen Datenbank in der Form einfacher, nicht geschachtelter Tabellen.

Wie bereits dargestellt, ist das relationale Datenbankmodell heute die übliche Organisationsform, in der Daten für die Abbildung im Rechner beschrieben werden. Datenbanken, die auf diesem Modell aufbauen, werden daher auch als relationale Datenbanken bezeichnet. Für die Darstellung gelten allgemein folgende Regeln:

- ∅ Alle Datensätze (Tabellenzeilen) sind gleich lang
- ∅ Jeder Datensatz (Tupel) kommt nur ein einziges Mal in einer Tabelle vor, die Reihenfolge der Sätze (Tupel) ist beliebig. Jede Tabellenzeile beschreibt einen Datensatz. Die eindeutige Identifizierung eines Tupels erfolgt über den „Schlüssel“
- ∅ Der Wert eines Attributs kommt aus einem bestimmten Wertebereich (Domäne). Jede Tabellenspalte beschreibt eine bestimmte Eigenschaft (Attribut) des Datenobjekts.
- ∅ Durch Kombination einzelner Spalten (Attribute) und Zeilen (Datensätze) können neue Tabellen (Relationen) gebildet werden (Abgeschlossenheit).

Unter Normalisierung wird nun das Beseitigen von sogenannten funktionellen Abhängigkeiten verstanden.

## 4.2.1 funktionale Abhängigkeiten

Eine funktionale Abhängigkeit ist zwischen Attributmengen, z.B.  $A_i$  und  $A_j$  dann gegeben, falls in jedem Tupel der Relation der Attributwert unter  $A_i$ -Komponenten den Attributwert unter den  $A_j$ -Komponenten festlegt. Unterscheiden sich also zwei Tupel in den  $X$  – Attributen nicht, so haben sie auch gleiche Werte für alle  $Y$ - Attribute. Die funktionale Abhängigkeit wird dann mit  $A_i \rightarrow A_j$  bezeichnet.

### Definition 4.9.

Eine funktionale Abhängigkeit gilt dann innerhalb einer Relation zwischen Attributmengen  $X$  und  $Y$ , wenn in jedem Tupel der Relation der Attributwert unter den  $X$  – Komponenten den Attributwert unter den  $Y$  – Komponenten festlegt.

Folgendes Beispiel soll dies verdeutlichen

### Fallbeispiel 4.1.

In nachfolgender Relation gilt die funktionale Abhängigkeit ISBN  $\rightarrow$  Titel, Verlag da zwei in der Relation ISBN übereinstimmende Buch – Tupel auch unter den Attributen Titel und Verlag übereinstimmen müssen.

ISBN	Titel	Autor	Verlag
0-8053-1753-8	Princ.of DBS	Elmasri	Manz
0-8053-1783-8	Princ.of DBS	Navathe	Goldstein
0-8083-1753-8	Princ.of DBS	Elmasri	Wesley

Abbildung 4.2.: Beispieldaten für Fallbeispiel 4.1.

Eine funktionale Abhängigkeit ISBN  $\rightarrow$  Autor, Stichwort gilt nicht, da ein Buch unterschiedliche Autoren und Stichworte hat. Trivialer Weise gilt natürlich ISBN  $\rightarrow$  ISBN.

Als Spezialfall funktionaler Abhängigkeiten müssen Schlüssel betrachtet werden. Ein Schlüssel  $X$  beschreibt für ein Relationenschema eine funktionale Abhängigkeit, wenn  $X$  minimal ist (d.h. aus der kleinsten Menge Tupel identifizierender Attributwerte gebildet ist).

### Definition 4.10.

Ein Schlüssel ist minimal, wenn er aus der kleinsten Menge der Tupel identifizierenden Attributwerte gebildet wird.

Ziel eines sich auf Abhängigkeiten abstützenden Datenbankentwurfs einer relationalen Datenbank ist daher das Umformen aller funktionalen Abhängigkeiten in Schlüsselabhängigkeiten. Die Menge der Abhängigkeiten ist äquivalent zur Menge der Schlüsselbedingungen im resultierenden Datenbankschema (Abhängigkeitstreue).

### Fallbeispiel 4.2.

In nachfolgender Relation seien Studentendaten verwaltet ( $M\#$  bezeichne die Matrikelnummer,  $SR$  bezeichne die Studienrichtung).

Relation: Studenten

M#	NAME	ADRESSE	SR#	STUDR.	Start
123	Meier	Berggasse 19	880	Informatik	01.10.1989
124	Müller	Lange Gasse 19	890	Physik	01.03.1989
123	Meier	Berggasse 19	790	Psychologie	01.10.1985
125	Schmidt	Grabenweg 4	790	Psychologie	01.03.1990
128	Lang	Brückenweg 23	880	Informatik	01.10.1989
129	Lang	Brückenweg 23	890	Physik	01.10.1989
127	Bauer	Hochstraße	880	Informatik	01.10.1977

Abbildung 4.3.: Relation Studenten zum Fallbeispiel 4.2.

Die Relation Studenten zeigt ein wesentliches Problem: sie enthält redundante Informationen: Die Studiengänge – Kennziffer für die Studiengänge und das Startdatum des entsprechenden Studiengangs. Die Abhängigkeit dieser Daten (Bezeichnung der Studienrichtung, Startdatum) ist abhängig von der Studiengänge – Kennziffer. Diese Abhängigkeit sollte daher in einer eigenen

Tabelle verwaltet werden.

Eine weitere Abhängigkeit besteht in der Beziehung Name und Adresse, welche von der Matrikelnummer abhängig sind. Auch diese Abhängigkeit sollte in einer eigenen Tabelle dargestellt werden.

Es ist daher der Versuch zu unternehmen, die funktionalen Abhängigkeiten in der Relation Studenten entsprechend zu entschärfen, also die Relation zu Normalisieren. Betrachten wir dazu das Ergebnis der Relation:

Relation: STUDENT

MATNR	NAME	ADRESSE
123	Meier	Berggasse 19
124	Müller	Lange Gasse 19
125	Schmidt	Grabenweg 4
127	Bauer	Hochstraße 2
128	Lang	Brückenweg 23
129	Lang	Brückenweg 23

Relation: STUDIUM

MATNR	SRNR	ANFANGSDAT
123	88	01.10.1989
124	81	01.03.1989
123	79	01.10.1985
125	79	01.03.1989
127	88	01.10.1977
128	81	01.10.1989
129	88	01.10.1989

Relation: STUDIENRICHTUNG

SRNR	STUDR
88	Informatik
81	Physik
79	Psychologie

Abbildung 4.4. bis 4.6.: Normalform der Relation Studenten

Der vorliegende Zerlegungsprozess zeigt eine geeignete Auswahl von Spalten der Ausgangsrelation. Der Vorgang der Normalisierung kann daher auch als Zerlegung von Relationenschemata in kleinere, übersichtlichere Relationen betrachtet werden. Allerdings gilt: Es sind nur sinnvolle Zerlegungen (Kriterium: funktionale Abhängigkeiten) zugelassen, d.h.: Die Ausgangsrelation muss sich durch Zusammenfügen von Teilrelationen der Zerlegung rekonstruieren lassen. Es darf durch die Zerlegung weder

∅ Information verloren gehen

∅ Noch Information gewonnen werden.

#### *Definition 4.11.*

---

Eine Zerlegung einer Ausgangsrelation - durch den Vorgang der Normalisierung - in Teilrelationen, bei denen weder zusätzliche Informationen entstehen, noch Informationen verloren gehen, wird als **Verbundtreue Zerlegung** bezeichnet.

---

Zerlegungen sind nur dann sinnvoll, wenn sie verlustfrei sind. Beim Zusammensetzen der Teilrelationen müssen wieder genau die Tupel der Ausgangsrelation erzeugt werden. Es sollen aber möglichst wenige Tabellen erzeugt werden, die den Forderungen genügen.

#### *Definition 4.12.*

---

Verbundtreue Normalisierungen müssen minimal sein, d.h. die Anzahl der gewonnenen Relationen muss minimal sein.

---

Nachdem nun grundlegende Begriffe hinsichtlich der Zerlegung von Relationen im Zuge einer Normalisierung diskutiert wurden, sei auf die Eigenschaften der

∅ Minimalität

∅ Verbundtreue und

∅ Abhängigkeitstreue

näher eingegangen.

Wie bereits dargestellt, ist bei diesen Zerlegungen darauf zu achten, dass

nur semantisch sinnvolle und konsistente Anwendungsdaten dargestellt und alle Anwendungsdaten aus den Basisrelationen hergeleitet werden können.

Das erste Kriterium wird zur Transformationseigenschaft Abhängigkeitstreue, das zweite zur Verbundtreue führen. Transformationseigenschaft steht dabei für den Vorgang der Transformation einer Relation in normalisierte Teilrelationen.

## 4.2.5 Normalformen

Nachdem nun funktionale Abhängigkeiten ausführlich diskutiert wurden, sollen weitere Normalformen besprochen werden.

### 4.2.5.1 Relationen in der 1. Normalform

Wie in Definition 4.7. angegeben, befinden sich Relationen in der ersten Normalform (ENF), wenn die Attribute nur aus atomaren Werten bestehen.

Auf Attribute einer Relation in erster Normalform sind folgende Operationen zulässig:

- ∅ Nur ganze Sätze können gelöscht oder eingefügt werden
- ∅ Ein Schlüssel darf nicht isoliert modifiziert werden

Zur Erinnerung sollen nochmals kurz anhand von zwei Fallbeispielen die Möglichkeiten zur Implementierung eines Sachverhalts im Relationenmodell dargestellt werden.

#### *Fallbeispiel 4.11.*

Es soll die Beziehung zwischen Studenten und Dozenten repräsentiert werden.

Studenten werden durch deren Entitätsschlüssel S# eindeutig gekennzeichnet, Dozenten durch den Entitätsschlüssel D#. Über jeden Studenten wird sein Name und Alter abgelegt (Entitätsattribute). Jeder Dozent lehrt eine Programmiersprache – und beurteilt dabei Studenten (Beziehungsattribut).

Es ergeben sich somit folgende Relationen:

NAME	(S#   NAME)	ALTER	(S#, WERT)
S1	Karl		S1   20
S2	Vera		S2   35
S3	Vera		S3   26

Doziert	(S#   NAME)
D1	C++
D2	Java

Sprachkenntnisse	(S#, Sprache)	Kriterium
S1	C++	gut
S1	Java	mittel
S2	C++	schlecht
S2	Java	gut
S3	Java	gut
S3	C++	gut

Abbildung 4.13. – 4.15.: Einzelrelationen zum Fallbeispiel 4.11

Es wäre daher zunächst am einfachsten, alle Informationen in einer einzigen Relation zusammenzufassen:

Student	(S#	Sprache	Krit	Name	Alter)
S1	C++	gut	Karl	20	
S1	Java	mittel	Karl	20	
S2	C++	schlecht	Vera	35	
S2	Java	gut	Vera	35	
S3	Java	gut	Vera	26	
S3	C++	gut	Vera	26	

Abbildung 4.16.: Gesamtrelation zu Fallbeispiel 4.11.

Die in Abbildung 4.12. dargestellte Relation entspricht jedoch nicht dem Sachverhalt. Betrachten wir dazu ein Beispiel:

Angenommen, die Relation Student aus Abbildung 4.12. wird durch nachfolgendes Tupel ergänzt:

Student	(S#	Sprache	Krit	Name	Alter)
S1	C++	gut	Karl	20	
S1	Java	mittel	Karl	20	
S2	C++	schlecht	Vera	35	
S2	Java	gut	Vera	35	
S3	Java	gut	Vera	26	

S3	C++	gut	Vera	26
S3	C	mittel	Hans	32

Abbildung 4.17.: Gesamrelation zu Fallbeispiel 4.11 mit zus. fehlerhaftem Tupel.

Die gewählte Repräsentanz (eine einzige Relation) würde das Tupel ‚Hans‘ akzeptieren, die Schlüsselbedingung (S#, Sprache) ist nicht verletzt. Allerdings würde der Student S3 einmal Hans, einmal mit Vera identifiziert werden können. Obwohl die Relation Student (Abbildung 4.12.) in erster Normalform ist, werden nicht alle Zusammenhänge dem Sachverhalt entsprechend dargestellt. Es sind daher weitere Transformationsschritte notwendig.

Betrachten wir zu diesem Problem noch ein weiteres Beispiel:

### *Fallbeispiel 4.12.*

Im vorliegenden Beispiel soll die Möglichkeit zur Modellierung von Beziehungsattributen nochmals dargestellt werden:

Belehrt	(D#	S#)
	D1	S1
	D1	S3
	D2	S1
	D2	S2
	D2	S3

Beurteilung	(D#	S#	Kriterium)
	D1	S1	gut
	D1	S3	mittel
	D2	S1	gut
	D2	S2	schlecht
	D2	S3	gut

Abbildung 4.18 – 4.19.: Relationen für Beziehungsattribute

Betrachten wir nun die Entität Sprache genauer, es können diesbezüglich zwei Punkte festgehalten werden:

- ∅ Werden Sprachen im Sinne eines Attributs behandelt, so kann eine bestimmte Sprache nur im Zusammenhang mit einer die Sprache sprechenden Person genannt werden.
- ∅ Werden Sprachen im Sinne von Entitäten aufgefasst, so kann eine bestimmte Sprache auch eigenständig (d.h. ohne Nennung einer die Sprache sprechenden Person) behandelt werden.

Es kommt dem Entitätsbegriff somit folgende Bedeutung zu: Mit der Definition einer Entitätsmenge wird beabsichtigt, die der Entitätsmenge angehörenden Entitäten eigenständig (d.h. ohne Nennung anderweitiger Begriffe) behandeln zu können.

#### **4.2.5.2 Die zweite Normalform**

Im Fallbeispiel 4.11 wurde bereits gezeigt, dass eine Relation in erster Normalform noch immer fehlerhafte Tupel zulassen kann. Betrachten wir daher nochmals die ursprünglich aufgestellten Anforderungen an die Normalisierung:

- ∅ Eliminieren von Redundanz
- ∅ Eliminieren von Schwierigkeiten in Zusammenhang mit Einfügen von Tupeln (Beispielsweise Abbildung 4.12.)
- ∅ Eindeutiges Festhalten realitätskonformer Sachverhalte, d.h.: Ermitteln von Relationen, die keine Möglichkeiten bieten realitätskonforme, funktionale Abhängigkeiten zu verletzen.

Forderung für die zweite Normalform:

Die zweite Normalform versucht, aufgrund der Struktur von funktionalen Abhängigkeiten Redundanzen zu entdecken. Die zweite NF erlaubt keine partiellen Abhängigkeiten zwischen Schlüsseln des Relationenschemas und weiteren Attributen. Dies bedeutet, dass jedes Nichtschlüsselattribut nur vom Gesamtschlüssel – und nicht von Teilschlüsseln funktional abhängig sein darf.

Betrachten wir dazu ein Beispiel:

### *Fallbeispiel 4.13.*

Gegeben sei eine Relation in erster Normalform zur Verwaltung von Büchern [GSA01]:

Inventar #	Titel	ISBN	Autoren
0007	Dr. No	3-125	Flemming
1201	Objektdbs	3-111	Heuer
1201	Objektdbs	3-111	Schmidt
4711	Datenbanken	3-768	Witt
4711	Datenbanken	3-768	Vossen

4712	Datenbanken	3-891	Ullmann
4717	Pascal	3-999	Dijkstra
4717	Pascal	3-999	Wirth

Abbildung 4.20. Beispielrelation in erster Normalform zu Fallbeispiel 4.13

Folgende funktionalen Abhängigkeiten können identifiziert werden:

$Invnr \rightarrow Titel, ISBN$  und

$Invnr, Autor \rightarrow Invnr, Titel, ISBN, Autor$ .

Steht auf der rechten Seite einer funktionalen Abhängigkeit das ganze Relationenschema und ist die linke Seite minimal, dann ist die linke Seite automatisch Schlüssel für das Relationenschema.

#### Definition 4.14.

Steht auf der rechten Seite einer funktionalen Abhängigkeit das ganze Relationenschema und ist die linke Seite minimal, dann ist die linke Seite automatisch Schlüssel für das Relationenschema.

#### Definition 4.15

Eine partielle Abhängigkeit liegt nun vor, wenn ein Attribut funktional schon von einem Teil des Schlüssels abhängt.

Im Beispiel ist durch die zweite FD  $Invnr$  und  $Autor$  zusammen ein Schlüssel. Der  $Titel$  hängt aber allein von  $Invnr$  ab, also einem Teil des Schlüssels.

Die zweite Normalform kann nun durch die Elimination der rechten Seite der partiellen Abhängigkeit und einer Kopie der linken Seite erreicht werden.

Zu beachten ist, dass das partiell abhängige Attribut nur „stört“, wenn es kein Primattribut ist. Primattribute sind Attribute aus Schlüsseln des Relationenschemas. Attribute, die in Schlüsseln des Relationenschema vorkommen, werden also nicht auf partielle Abhängigkeit überprüft.

#### Definition 4.16.

Primattribute sind Attribute aus Schlüsseln des Relationenschemas.

Sei  $R = (R, K)$  ein erweitertes Relationenschema und  $F$  über  $R$ .  $A$  sei ein Attribut aus  $R$  und  $X \rightarrow Y$  eine FD aus  $F$ .

$A$  heißt unwesentlich in  $X \rightarrow Y$  bezüglich  $F$ , wenn

$$[X = AZ, Z \neq X \Rightarrow (F - \{X \rightarrow Y\} \cup \{Z \rightarrow Y\} \equiv F] \vee$$

$$[Y = AW, W \neq Y \Rightarrow (F - \{X \rightarrow Y\}) \cup \{X \rightarrow W\} \equiv F]$$

$A$  kann also aus der FD  $X \rightarrow Y$  entfernt werden, ohne dass sich die Hülle von  $F$  ändert.

Eine FD  $X \rightarrow Y$  heißt linksreduziert, wenn kein Attribut in  $X$  unwesentlich ist.

$Y$  hängt partiell von  $X$  bzgl.  $F$  ab, wenn die FD  $X \rightarrow Y$  nicht linksreduziert ist.  $Y$  hängt voll von  $X$  ab, wenn die FD  $X \rightarrow Y$  linksreduziert ist.

$R$  ist dann in zweiter Normalform, wenn  $R$  in 1.NF ist und jedes Nicht – Primattribut von  $R$  voll von jedem Schlüssel von  $R$  abhängt. Ein Datenbankschema  $S$  ist in 2.NF bezüglich  $F$  genau dann, wenn alle  $R \in S$  in 2. NF bezüglich  $F$  sind.

#### Definition 4.17.

$R$  ist dann in zweiter Normalform, wenn  $R$  in 1.NF ist und jedes Nicht – Primattribut von  $R$  voll von jedem Schlüssel von  $R$  abhängt.

#### Anmerkung

Die zweite Normalform reicht nicht aus, um Anomalien auszuschließen, sie ist daher nur von historischer Bedeutung.

#### Fallbeispiel 4.14.

Beispiel: Lagerhaltungsrelation

$T\#$  bezeichne die Teilnummer,

$L\#$  bezeichne die Lagernummer.

Vorrat	(T#)	<u>L#</u>	MENGE	LAGERADRESSE)
	101	1	25	Waagg. 10
	102	3	410	Krugerstraße 42
	102	1	300	Waagg. 10

112	4	10	Brunner Str. 105
-----	---	----	------------------

Abbildung 4.21.: Relation Vorrat

Folgende Probleme können identifiziert werden:

- ∅ funktionale Abhängigkeit zwischen Teil, Lager und Lageradresse !
- ∅ Redundanz: Lageradresse ist redundant gespeichert

Annahme: Lager1 wird verlegt

Vorrat	(T#)	L#	MENGE	LAGERADRESSE)
	101	1	25	Kroißbergstraße 1
	102	3	410	Kruggerstraße 42
	102	1	300	Waagg. 10
	112	4	10	Brunner Str. 105

Abbildung 4.22.: Relation Vorrat mit Inkonsistenz.

Inkonsistenz = Anomalie durch Adressänderung, da Lager kein Einzelschlüssel

Annahme: Teil 102 wird aus Lager 3 gelöscht:

Vorrat	(T#)	L#	MENGE	LAGERADRESSE)
	101	1	25	Kroißbergstraße 1
	102	1	300	Waagg. 10
	112	4	10	Brunner Str. 105

Abbildung 4.23.: Relation Vorrat mit Inkonsistenz

Adresse von Lager 3 geht verloren , Löschanomalie

Schlussfolgerung: Transformation in 2. NF !

Vorrat	( <u>TEIL</u> )	<u>LAGER</u>	MENGE)
	101	1	25
	102	3	410
	102	1	300
	112	4	10

Lager	<u>LAGER</u>	LAGERADRESSE)
	1	Kroißbergstraße 1
	3	Kruggerstraße 42
	4	Brunner Str. 105

Abbildung 4.24.: Vorrat-Relation aus Fallbeispiel 4.14 in 2. NF

### Fallbeispiel 4.15

Funktionale Abhängigkeiten in der Relation Student:

*Student(S#, Sprache, Kriterium, Name, Alter)*

Nur die Spalte Kriterium ist nicht funktional abhängig vom Schlüsselteil S#. Alle übrigen Nichtschlüsselspalten sind vom Schlüsselteil S# funktional abhängig. Daher Überführung in 2. Normalform, also verletzende Nichtschlüsselspalten werden aus der problembehafteten Relation eliminiert und zu neuen problemlosen Relationen vereinigt:

*Student(S#, Name, Alter)*

*sprachk(S#, sprache, Kriterium)*

### 4.2.5.3 Die dritte Normalform

Ein weiteres Problem im Bezug auf Normalformen stellt die Transitivität dar. Bestimmt ein Schlüssel K eine Attributmenge X funktional, die Attributmenge X aber selbst wieder eine Attributmenge Y (des gleichen Schemas), so liegt eine transitive Abhängigkeit vor:  $K \rightarrow X \rightarrow Y$ . Es gilt daher die Transitivität zu brechen – sprich das transitiv abhängige Attribut in ein neues Relationenschema zu übertragen.

Sind alle Transitivitäten beseitigt, ist das Relationenschema in dritter Normalform.

### Definition 4.18.

Ist ein Relationenschema in 2. Normalform, und sind keine transitiven Abhängigkeiten vorhanden, so ist das Relationenschema in 3. Normalform.

Formal sei wieder angegeben: Sei R ein Relationenschema,  $X \subseteq R$  und F eine FD – Menge über R. Ein  $A \in R$  heißt transitiv abhängig von X bezüglich F genau dann, wenn es ein  $Y \subseteq R$  gibt mit  $X \rightarrow Y$ ,  $Y \not\rightarrow X$ ,  $Y \rightarrow A$ ,  $A \notin XY$ .

Ein erweitertes Relationenschema  $R = (R, K)$  ist in 3. NF bezüglich F genau dann, wenn

$\nexists A \in R: A$  ist Nicht – Primattribut in R

^ A transitive abhängig von  
einem K ε K bezüglich F.

### Fallbeispiel 4.16.

Relation Personal (in 2. NF):

Personal	(P#	Abteilung	Gebäude)
	1215	Buchh.	A12
	2410	Vertrieb	A14
	2412	PR	B8
	809	Buchh.	A12

Abbildung 4.25.: Relation Personal in 2. NF

$FD = Personal\# \rightarrow Abteilung, Gebäude$   
 $Abteilung \rightarrow Gebäude$

Annahme: Verlegung der Buchhaltungsabteilung:

Personal	(P#	Abteilung	Gebäude)
	1215	Buchh.	A12
	2410	Vertrieb	A14
	2412	P	B8
	809	Buchh.	A12

Abbildung 4.26.: Relation Personal in 2. NF

Problem: Redundanz bewirkt Anomalie nach Adressänderung

Annahme: Entfernen des Angestellten 2412:

Personal	(P#	Abteilung	Gebäude)
	1215	Buchh.	A12
	2410	Vertrieb	A14
	809	Buchh.	A12

Abbildung 4.27.: Relation Personal in 2. NF

Problem: Gebäudeinformation geht verloren

Daher Transformation in 3. NF:

Personal	(P#	Abteilung)
	1215	Buchh.
	2410	Vertrieb
	2412	P
	809	Buchh.

Abteilung	(A#	Gebäude)
	Buchh.	A7
	Vertrieb	A14
	P	B8
	Buchh.	A12

Abbildung 4.28 – 4.29.: Relationenschema in 3. NF

### Fallbeispiel 4.17

Relation Student:

FB# steht für Fachbereich

Student(S#,Name,Gebdat,Adr,FB#,FBName,Dekan)

u.A. können folgende FDs identifiziert werden:

$S\# \rightarrow FB\# \rightarrow Dekan$

$S\# \rightarrow Dekan \rightarrow FBName$

Student ist in der zweiten Normalform (ZNF), nicht aber in dritter Normalform (DNF). Daher Überführung in dritte NF:

Student(S#, Name, Adr, Gebdat, FB#)

Fachbereich(FB#,FBName,Dekan)

#### 5.2.5.4 Spezielle Normalformen

Meist ist mit Erreichen der dritten Normalform der Normalisierungsprozess abgeschlossen. In manchen Fällen kann aber ein unbefriedigendes Ergebnis erreicht worden sein:

∅ die Relation hat mehrere Schlüsselkandidaten

∅ die Schlüsselkandidaten sind zusammengesetzt, bestehen also aus mehreren Attributen

∅ die Schlüsselkandidaten überlappen sich mit dem Primärschlüssel, d.h.: Sie haben mindestens ein Attribut mit dem Primärschlüssel gemeinsam

Falls Relationen mehrere zusammengesetzte und sich überlappende Schlüsselkandidaten aufweisen, wird die Boyce/Codd - Normalform zur Normalisierung herangezogen.

### 5.2.5.5 Boyce/Codd- Normalform

In der Definition der dritten Normalform wurden nur Nicht – Primattribute auf transitive Abhängigkeit von einem Schlüssel getestet. Manchmal lassen sich jedoch auch Redundanzen innerhalb der Schlüsselattribute feststellen.

#### Fallbeispiel 4.18

Betrachten wir das System der Postleitzahlen:

im Bereich der Attribute *PLZ, Ort, Straße, Hausnummer* können folgende funktionale Abhängigkeiten festgestellt werden :

*Ort, Straße, Hausnummer* -> *PLZ*

*PLZ* -> *Ort*

Schlüssel für diese vier Attribute ist neben der Attributmenge *Ort, Straße Hausnummer* auch noch *PLZ, Straße, Hausnummer*. Da alle Attribute nur Primattribute sind, müssen keine funktionalen Abhängigkeiten gesucht werden. Dabei wäre bei Ausdehnen der Definition von transitiven Abhängigkeiten auf alle Attribute eine Redundanz zu erkennen:

*PLZ, Straße, Hausnummer* -> *PLZ* -> *Ort*

Genau genommen wäre dies sogar eine partielle Abhängigkeit. Die Zuordnung von Orten zu Postleitzahlen enthält also Redundanzen.

Dehnen wir die Untersuchung von transitiven Abhängigkeiten auf Primitivattribute aus, so sind wir bei der Definition der BCNF:

Ein erweitertes Relationenschema  $R = (R, K)$  ist bezüglich  $F$  in BCNF genau dann, wenn

$\exists A \in R: A$  transitiv abhängig von einem  $K \in K$  bezüglich  $F$ .

$S$  ist in BCNF bezüglich  $F$  genau dann, wenn alle  $R \in S$  in BCNF bezüglich  $F$  sind, ist leicht zu zeigen, dass  $S$  ist in BCNF  $\Rightarrow S$  ist in 3. NF.

Dieser Zusammenhang gilt aber nicht umgekehrt !

#### Definition 4.19.

Eine Relation ist dann in BCNF, falls jedes (funktional) determinierendes Attribut zugleich Schlüsselkandidat ist.

Ein (funktional) determinierendes Attribut (z.B.  $A_i$ ) bzw. eine determinierende Attributkombination liegt dann vor, wenn jeder Attributwert des determinierenden Attributs (bzw. der determinierenden Attributkombination) genau einen Attributwert eines anderen Attributs (z.B.  $A_j$ ) bzw. einer anderen Attributkombination festlegt. Man spricht in diesem Zusammenhang auch von Determinanten. Eine Determinante ist demnach ein Attribut oder eine Gruppe von Attributen (Schlüsselkandidat), von der beliebig andere Attribute funktional abhängig sind.

#### Fallbeispiel 4.18.

Gegeben sind folgende Fakten

Ein Student belegt eine bestimmte Vorlesung bei einem Dozenten

Ein Dozent hält Vorlesungen nur zu einem Thema (d.h. lehrt nur ein Fach)

Ein Vorlesungsthema kann von mehreren Dozenten unterrichtet werden

Lösungsversuch1:

Belegung(Dozent#, Student#, Vorlesung)

Das Schema dieser Relation zeigt eine teilweise Abhängigkeit ( $Dozent\# \rightarrow Vorlesung$ ) und ist nicht einmal in 2. Normalform

Lösungsversuch2:

Belegung(Student#, Vorlesung, Dozent#)

Die Relation ist in DNF, nicht aber in BCNF, da die Abhängigkeit  $Dozent\# \rightarrow Vorlesung$  besteht und  $Dozent\#$  ein Schlüsselkandidat ist. Die Lösung zeigt folgende Mängel:

∅ Bevor nicht ein Student die Vorlesung belegt, kann kein Dozent, der die Vorlesung hält, eingetragen werden.

∅ Für jeden Studenten, der eine bestimmte Vorlesung belegt hat, wird der Dozent, der die Vorlesung hält, redundant gespeichert.

Lösungsversuch3:

Kann durch Zerlegung der Relation in zwei Relationen erreicht werden. Die Attribute, die die BCNF verletzen, werden in eine abgetrennte Relation herausgezogen. Das determinierende Attribut übernimmt die Funktion des Primärschlüssels.

Besucht(Dozent#, Student#)

Hält(Dozent#, Vorlesung)

Der Grund für die fehlerhafte Lösungsmöglichkeit (unter 1. und 2.) liegt in der falschen Darstellung der Beziehung DOZENT,

STUDENT, VORLESUNG. Es liegt hier keine Dreifachbeziehung vor, sondern nur zwei unabhängige Zweierbeziehungen !.

Ziel eines Datenbankentwurfs ist ein Datenbankschema, dessen Relationenschemata möglichst in BCNF sind. Das kann durch fortschreitendes Zerlegen von Schemata, die diese Kriterien nicht erfüllen, erreicht werden.

### 5.2.5.6 Vierte Normalform

Betrachten wir dazu ein einführendes Beispiel:

#### *Fallbeispiel 4.19.*

Gegeben ist die folgende, nicht normalisierte Relation:

VO_Dozent	Vorlesung	Dozent	Inhalt
	Datenbanken	Heinrich, Ernst	Grundlagen
	C++	Maier	Rekursion, MFC

Abbildung 4.30.: Beispielrelation, nicht normalisiert

In dieser Relation werden folgende Fakten ausgedrückt:

- ∅ Eine bestimmte Vorlesung kann von beliebig vielen Dozenten gehalten werden und kann beliebig viele Merkmale zeigen.
- ∅ Zwischen Dozenten und Merkmalen eines Kurses (Vorlesung) bestehen keinerlei Abhängigkeiten
- ∅ Dozenten und Merkmale können mit jedem beliebigen Kurs in Verbindung stehen

Es handelt sich um eine Dreifachbeziehung, die normalisiert folgende Form aufweist:

VO_Dozent	<u>Vorlesung</u>	<u>Dozent</u>	<u>Inhalt</u>
	Datenbanken	Heinrich,	Grundlagen
	Datenbanken	Ernst	Grundlagen
	C++	Maier	Rekursion
	C++	Maier	MFC

Abbildung 4.31.: Beispielrelation, nicht normalisiert

Leicht ist erkennbar, dass die vorliegende Relation redundante Daten enthält, woraus Anomalien entstehen können.

Die vorliegende, normalisierte Relation ist aber sogar in BCNF, denn alle Attributwerte sind Primärschlüssel. Es gibt außer der Kombination der Attribute VORLESUNG, DOZENT-NAME, MERKMAL kein weiteres funktional determinierendes Attribut.

Spaltet man die vorliegende Tabelle in zwei Relationen auf (Projektionen der ursprünglichen Relation), so ergibt sich die Lösung des Problems:

VO_Dozent	Vorlesung	Dozent
	Datenbanken	Heinrich
	Datenbanken	Ernst
	C++	Maier

VO	Vorlesung	Inhalt
	Datenbanken	Grundlagen
	C++	Schleifen
	C++	MFC

Abbildung 4.32 bis 4.33.: Beispielrelation in BCNF und 4. NF

Für jede von einer bestimmten Person gehaltene Vorlesung gibt es eine identische Menge von Merkmalswerten. Man sagt, dass Attribut MERKMAL ist **mehrwertig abhängig** (multi value dependent) vom Attribut VORLESUNG. bzw. DOZENT.

Das führt zu der folgenden Definition einer mehrwertigen Abhängigkeit:

#### *Definition 4.20*

In dem Relationenschema  $R = \{A_i, A_j, A_k\}$  ist das Attribut  $A_k$  vom Attribut  $A_i$  mehrwertig abhängig, falls zu einem  $A_i$ -Wert, für jede Kombination dieses  $A_i$ -Werts mit einem  $A_j$ -Wert, eine identische Menge von  $A_k$ -Werten erscheint. Mehrwertige Abhängigkeiten erscheinen in einer Relation immer paarweise.

Werden diese mehrwertigen Abhängigkeiten beseitigt, dann spricht man von einer Relation in vierter Normalform.

Eine Relation ist in 4. NF, falls sie in 3. NF bzw. BCNF ist und keine mehrwertigen Abhängigkeiten zeigt.

Verstöße gegen die vierte Normalform entstehen häufig bereits in der Entwurfsphase. So hätte man im vorliegenden Beispiel erkennen können, dass die Beziehung zwischen VORLESUNG und DOZENT bzw. VORLESUNG und MERKMAL voneinander unabhängig sind und keine Dreifachbeziehung darstellen.

## 4.2.6 Der Normalisierungsprozess

Nachfolgend soll nun anhand eines Fallbeispiels der vollständige Normalisierungsprozess gezeigt werden:

### Fallbeispiel 4.20.

Problemstellung:

Gesucht ist ein Datenmodell für eine relationale Datenbank, das die Entitätstypen PERSON (Mitarbeiter), ABTEILUNG, PRODUKT berücksichtigt.

Eine Analyse des Sachverhalts führt zu folgenden Fakten:

- ∅ Ein Mitarbeiter hat einen Namen und einen Wohnort.
- ∅ Ein Mitarbeiter ist in einer Abteilung tätig
- ∅ Ein Mitarbeiter arbeitet an mehreren Produkten jeweils für eine vorbestimmte Zeit
- ∅ Jede Abteilung hat einen Namen
- ∅ Jedem Produkt ist ein Name zugeordnet
- ∅ In einer Abteilung sind mehrere Personen tätig
- ∅ An einem Produkt arbeiten in der Regel mehrere Personen

Betrachten wir diese Anforderungen genauer, so können folgende Eigenschaften identifiziert werden:

- ∅ Entitäten: PERSON, ABTEILUNG, PRODUKT
- ∅ Beziehungen zwischen Entitäten
- ∅ Entitätsattribute
- ∅ Beziehungsattribut

Betrachten wir daher als ersten Ansatz eine einzelne Relation:

Person	(Pers#	Name	Wohnort	Abteilung#	AbtName	Produkt#	ProduktName	Arbeitszeit)
	101	Müller	Baden	1	Prod-Bad	10,11	Tisch, Sessel	120,80
	102	Maier	Wien	2	Prod-Wien	20	Trafo	80,60
	103	Rolf	Linz	2	Prod-Wien	10,11,20	Tisch, Sessel, Trafo,	230,430,240
	104	Pauli	Graz	4	Prod-Graz	10,20	Tisch, Trafo	120,280

Abbildung 4.34.: Beispielrelation, nicht normalisiert

Leicht ist ersichtlich, dass es bei der nicht normalisierten Darstellung zu Anomalien kommen kann.

Überführung in die 1. NF:

Hier ist in Verbindung mit einem bestimmten Schlüsselwert immer höchstens ein einziger Wert eines Attributs erlaubt.

Person	(PE#	Name	Wohnort	A#	AbtName	PR#	ProduktName	Arbeitszeit
	101	Müller	Baden	1	Prod-Bad	10	Tisch	120
	101	Müller	Baden	1	Prod-Bad	11	Sessel	80
	102	Maier	Wien	2	Prod-Wien	20	Trafo	80
	103	Rolf	Linz	2	Prod-Wien	10	Tisch	230
	103	Rolf	Linz	2	Prod-Wien	11	Sessel	430
	103	Rolf	Linz	2	Prod-Wien	20	Trafo	240
	104	Pauli	Graz	3	Prod-Graz	10	Tisch	120
	104	Pauli	Graz	3	Prod-Graz	20	Trafo	280

Abbildung 4.35.: Beispielrelation in 1. NF

In Abbildung 4.25 können zahlreiche funktionale Abhängigkeiten erkannt werden:

PE# -> Name, Wohnort, A#, AbtName

PR# -> ProduktName

(PE#, PR#) -> Name, Wohnort, A#, AbtName, PR#, ProduktName, Arbeitszeit

Weiters kann es zu Anomalien beim Einfügen zusätzlicher Datensätze kommen. Eine Person könne beispielsweise mehrere Wohnorte bekommen, oder Abteilungen mehrere Namen ....

Überführung in die 2. Normalform:

Hier muss jedes nicht dem Schlüssel zugehörnde Attribut funktional abhängig vom Gesamtschlüssel (nicht von den einzelnen Schlüsselteilen) sein.

Die Transformation in 2. NF bedeutet ein Aufspalten der Relation:

Person	(PE#)	Name	Wohnort	A#	AbtName)
	101	Müller	Baden	1	Prod-Bad
	102	Maier	Wien	2	Prod-Wien
	103	Rolf	Wien	2	Prod-Wien
	104	Pauli	Graz	3	Prod-Graz

PE-PR	(PE#)	PR#	Zeit)
	101	10	120
	101	11	80
	102	20	80
	103	10	230
	103	11	430
	103	20	240
	104	10	120
	104	20	280

Produkt	(PR#)	PoduktName)
	10	Tisch
	11	Sessel
	20	Trafo

Abbildung 4.36 bis 4.38.: Relationenschema in 2. Normalform

Im vorliegendem Relationenschema kann allerdings immer noch eine Anomalie auftreten (Abteilungsnummer <-> Abteilungsname).

Überführung in die 3. Normalform:

Es ist keine funktionale Abhängigkeit zwischen nicht dem Schlüssel angehörenden Attributen erlaubt.

In der Relation Person existiert jedoch noch eine funktionale Abhängigkeit: A# -> AbtName.

Person	(PE#)	Name	Wohnort	A#)
	101	Müller	Baden	1
	102	Maier	Wien	2
	103	Rolf	Wien	2
	104	Pauli	Graz	3

Abteilung	(A#)	AbtName)
	1	Prod-Baden
	2	Prod-Wien
	3	Prod-Graz

Abbildung 4.39 – 4.40.: Relationenschema in 3. Normalform

Somit ist die 3. Normalform erreicht.