

# LE 4 – IBIS – Datenbanken

**Relationenmodell, funktionale Abhängigkeiten und Normalisierung**

Prof. Dr. Markus Grüne, FB03, Wirtschaftsinformatik

## Lernziel / Fragen

- Was sind Relationen?
- Wie wird ein ER-Modell in ein Relationenmodell übersetzt?
- Was ist ein „gutes“ Datenmodell?
- Wiederholungen, wo notwendig

Modellierung mit ER-Diagrammen bzw. EER-Diagrammen.

# Relationen

- Eine relationale Datenbank ist eine Sammlung von **Relationen**, die als **Tabellen** dargestellt werden.
- Im Sinne der konzeptuellen Modellierung werden Entitäten und (fast alle) Beziehungen in Tabellen (als Relationen) dargestellt.
- Die Spalten (Columns) der Tabelle bezeichnen die Attribute der Relation. Die Zeilen (Rows) heißen auch Datensätze oder Einträge.

...	Titel	Nachname	Vorname	Gebdatum	SchulEintritt	Stufe
1	Dr.	Schmidt	Erika	1949-07-01	1974-08-01	StudienDirektorin
2	NULL	Schön	Helmut	1944-01-07	1971-03-01	StudienDirektor
3	NULL	Glensmann	Jürgen	1964-06-06	1989-05-01	Studienrat
4	NULL	Derwall	Jupp	1954-03-02	1980-06-01	OberStudienrat
5	NULL	Nerz	Otilie	1958-05-06	1986-05-06	OberStudienrat
6	NULL	Lukas	Laura	1975-01-09	2000-08-17	Studienrätin
7	NULL	Meier	Horst	1955-04-04	1985-07-04	Oberstudienrat
8	NULL	Müller	Gerd	1971-02-02	2000-09-14	Studienrat
9	Dr.	Bauer	Renate	1961-05-01	1987-04-02	Oberstudienrätin
10	NULL	Hummel	Heinz	1965-02-02	1990-01-01	Studienrat

*Relation „lehrer“ als Tabelle*

# Relationen

- Eine Relation besitzt
  - Einen Namen
  - Zeilen (Tupel)
  - Attribute mit Domänen (Wertebereich) und jeweils einem Namen

Wertebereiche  
können auch  
NULL-Werte  
erlauben

Column Name	Datatype	NOT NULL	AUTO INC	Default Value	Flags	Commer
PNr	INTEGER	✓		0	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	
Titel	CHAR(5)			NULL	<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Nachname	CHAR(20)	✓			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Vorname	CHAR(20)	✓			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Gebdatum	DATE	✓		0000-00-00		
SchulEintritt	DATE	✓		0000-00-00		
Stufe	CHAR(20)	✓			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	

Wertebereiche

# Wertebereiche – Transact-SQL Datentypen

Exact Numeric	Approximate Numeric	Character	Date/Time	Binary	Other
tinyint	float	char	date	binary	cursor
smallint	real	varchar	time	varbinary	hierarchyid
int		text	datetime	image	sql_variant
bigint		nchar	datetime2		table
bit		nvarchar	smalldatetime		timestamp
decimal/numeric		ntext	datetimeoffset		uniqueidentifier
numeric					xml
money					geography
smallmoney					geometry

Quelle: Microsoft, 2017

# Datentyp-Konvertierung

- Implizite Konvertierung
  - Kompatible Datentypen werden automatisch umgeformt
- Explizite Konvertierung
  - Mittels Funktion
    - CAST / TRY\_CAST
    - CONVERT / TRY\_CONVERT
    - PARSE / TRY\_PARSE
    - STR
    - ...

Quelle: Microsoft, 2017

## Definition „Relation“ - formal

$A_1, \dots, A_n$  seien Attribute mit den Domänen  $D_i = \text{dom}(A_i)$ .

$R(A_1, \dots, A_n) \subseteq D_1 \times D_2 \times \dots \times D_n$  ist eine Relation vom Grad  $n$ .

Ein Tupel aus  $R$  hat dann die Form  $(r_1, \dots, r_n) \in R$  mit  $r_i \in D_i$ .

Eine Relation ist demnach eine (unechte) Teilmenge des kartesischen Produkts der Domänen der Attribute.

### Beispiel

$\text{Lehrer} \subseteq \text{tinyint} \times \text{nvarchar}(50) \times \text{nvarchar}(50) \times \text{nvarchar}(50) \times \text{date} \times \text{date} \times \text{nvarchar}(50)$

$(1, \text{"Dr."}, \text{"Schmidt"}, \dots, \text{"Studiendirektorin"}) \in \text{Lehrer} \leftrightarrow$

$(1, \text{"Dr."}, \text{"Schmidt"}, \dots, \text{"Studiendirektorin"}) \in \text{tinyint} \times \text{nvarchar}(50) \times \text{nvarchar}(50) \times \text{nvarchar}(50) \times \text{date} \times \text{date} \times \text{nvarchar}(50)$

# Das relationale Modell – Elemente

Element-klasse	Name	Deutscher Begriff
Relationale Objekte	Domain	Wertebereich
	Relation	Tabelle
	Degree	Ausdehnungsgrad der Tabelle
	Attribut	Spalte
	Tupel	Datensatz, Rekord
	Candidate Key	Eindeutiger (möglicher) Schlüssel
	Primary Key	Hauptschlüssel
	Alternate Key	Alternativschlüssel
	Foreign Key	Fremdschlüssel

Element-klasse	Name	Deutscher Begriff
Relationale Integritätsregeln	Entity-Integrität	
	Referenzielle Integrität	
Relationale Operationen	Restriction	Zeilenselektion / Selektion
	Projection	Spaltenselektion / Projektion
	Product	Kartesisches Produkt
	Union	Vereinigung
	Intersection	Schnittmenge
	Difference	Differenz
	Join	Verbund / Verbindung
	Division	Division

In Anlehnung an: Sauer, H. (1998): Relationale Datenbanken – Theorie und Praxis, 4. Aufl., Bonn, Addison-Wesley-Longman, S.29-30.



## Schlüssel (Keys) – Schlüsselkandidat / Kandidatenschl.

- Ein Attribut oder eine Kombination von Attributen, die ein Tupel einer Relation eindeutig identifiziert heißt Schlüsselkandidat bzw. Kandidatenschlüssel.
- Ein Schlüsselkandidat ist „minimal“, d.h. durch Entfernen eines Attributs geht die Schlüsseleigenschaft verloren.
- Ein Schlüsselkandidat erhält die Rolle des Primärschlüssels (s. folgende Folie).
- Achtung: Es kann mehrere Schlüsselkandidaten für eine Relation geben!


## Schlüssel (Keys) - Primärschlüssel

- Jede Relation besitzt genau einen Primärschlüssel.
- Der Primärschlüssel erlaubt es, ein Tupel der Relation eindeutig zu identifizieren.
- Der Primärschlüssel wird definiert, indem ein **Kandidatenschlüssel zum Primärschlüssel erklärt** wird **oder** indem ein **neues Attribut** speziell für diesen Zweck der Relation hinzugefügt wird.
- Primärschlüssel dürfen keine NULL-Werte enthalten.
- Primärschlüssel können auch aus mehreren Attributen zusammengesetzt sein.

# Beispiel für einen Primärschlüssel

Der Primärschlüssel der folgenden Relation ist durch das Schlüsselsymbol markiert.  
Da es sich um den Schlüssel handelt, darf dieser nicht „NULL“ sein.

Spalten: + Neu - Entfernen ▲ Auf ▼ Ab

#	Name	Datentyp	Länge/SET	Vorzeic...	Erlaube NULL
 1	<b>PNr</b>	<b>TINYINT</b>	<b>3,0</b>	<input type="checkbox"/>	<input type="checkbox"/>
2	Titel	NVARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	Nachname	NVARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	Vorname	NVARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5	GebDatum	DATE	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	SchulEintritt	DATE	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>
7	Stufe	NVARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>

# Einfügeanomalie (Wiederholung)

PersID	Name	Geb.Dat	Wohnort	AbtNr	Abtname	AbtLtr
1	Theo Retisch	12.3.1986	Wetzlar	1	Vertrieb	Schmitt
2	Heinz Ellmann	23.6.1976	Limburg	1	Vertrieb	Schmitt
3	Wendy Lador	11.11.1995	Koblenz	2	Service	Werner
				3	Entwicklung	Wolf
4	Justin Time	23.2.1987	Gießen	1	Vertrieb	Schmidt
5	Kenny Ned	3.9.1977	Marburg	3	Entwicklung	Wolf

# Änderungsanomalie

PersID	Name	Geb.Dat	Wohnort	AbtNr	Abtname	AbtLtr
1	Theo Retisch	12.3.1986	Wetzlar	1	Vertrieb	Schmitt
2	Heinz Ellmann	23.6.1976	Limburg	1	Vertrieb	<del>Schmitt</del> Geck
3	Wendy Lador	11.11.1995	Koblenz	2	Service	Werner

# Löschanomalie

PersID	Name	Geb.Dat	Wohnort	AbtNr	Abtname	AbtLtr
1	Theo Retisch	12.3.1986	Wetzlar	1	Vertrieb	Schmitt
2	Heinz Ellmann	23.6.1976	Limburg	1	Vertrieb	Schmitt
3	Wendy Lador	11.11.1995	Koblenz	2	Service	werner

Was passiert mit  
der Service-  
Abteilung?

# Anomalien / Defekte

## Einfügedefekt (Einfügeanomalie)

- Beim Anlegen einer neuen Abteilung stehen noch keine Mitarbeiter zur Verfügung.
- Was machen Sie, wenn Sie einen Mitarbeiter ohne Abteilung erfassen müssen?

## Änderungsdefekt (Änderungsanomalie)

- Beim Ändern eines Abteilungsleiters müssen viele Mitarbeiter geändert werden!

## Löschdefekt (Löschanomalie)

- Wird ein Mitarbeiter einer Abteilung mit nur diesem einen Mitarbeiter gelöscht, so verschwindet die Abteilung mit!

**Abhilfe schafft die Reduzierung der Redundanzen!**

# Problem 1

## Mitarbeiter

PersID	Name	Geb.Dat	Wohnort	Abschlüsse	AbtNr	Abtname	AbtLtr
1	Theo Retisch	12.3.1986	Wetzlar	Fachinformatiker 2005, BSc 2010, MSc 2013	1	Vertrieb	Schmitt
2	Heinz Ellmann	23.6.1976	Limburg	Versicherungskaufmann 1990	1	Vertrieb	Schmitt
3	Wendy Lador	11.11.1995	Koblenz	FISI 2010, BSc 2015	2	Service	Werner

Wie sortieren wir  
nach Nachnamen?

Wie viele  
sollen es  
denn sein?



# Erste Normalform

Eine Relation ist in der ersten Normalform (1NF), wenn

- sie einen Primärschlüssel besitzt,
- alle Attribute atomar sind und
- sie keine Wiederholgruppen besitzt.

Nutzen:

- Abfragen werden erleichtert / ermöglicht

Korrekturen unseres Beispiels:

- Trennung von Name und Vorname
- Auslagern der Abschlüsse (Wiederholgruppe) in eigene Relation

# Problem 1 – Lösung (1.NF)

## Mitarbeiter

PersID	Nachname	Vorname	Geb.Dat	Wohnort	AbtNr	Abtname	AbtLtr
1	Retisch	Theo	12.3.1986	Wetzlar	1	Vertrieb	Schmitt
2	Ellmann	Heinz	23.6.1976	Limburg	1	Vertrieb	Schmitt
3	Lador	Wendy	11.11.1995	Koblenz	2	Service	Werner






Fremdschlüssel

PersID	Abschluss	Jahr
1	Fachinformatiker	2005
1	BSc	2010
1	MSc	2013
2	Versicherungskaufmann	1990
3	FISI	2010
3	BSc	2015

## Abschluss

# Problem 2 / Zweite Normalform

## Projektmitarbeiter

 PersID	 Nachname	 ProjNr	 ProjName	 Stunden
1	Retisch	13	Intranet	33
2	Ellmann	13	Intranet	85
3	Lador	21	Webauftritt	121

Redundanz

Funktionale Abhängigkeiten:

- ✓ PersID → Name
- ✓ ProjNr → ProjName
- ✓ PersID, ProjNr → • Stunden

Eine Relation ist in der zweiten Normalform (2NF), wenn

- sie bereits in der ersten Normalform vorliegt und
- alle Nicht-Schlüssel-Attribute vom Primärschlüssel voll funktional abhängig sind.

Nutzen:



- Jede Relation modelliert nur einen Sachverhalt
- Reduktion von Redundanz und somit Inkonsistenzen

Korrekturen unseres Beispiels:

- Auslagerung des Nachnamens
- Auslagerung des Projektnamens

# Problem 2 – Lösung (2.NF)

## Projektmitarbeiter

 PersID	<del>Nachname</del>	 ProjNr	<del>ProjName</del>	<del>Stunden</del>
1	Retisch	13	Intranet	33
2	Ellmann	13	Intranet	85
3	Lador	21	Webauftritt	121

↓ Fremdschlüssel  
Mitarbeiter

PersID	Nachname
1	Retisch
2	Ellmann
3	Lador

↓ Fremdschlüssel  
Projekte

ProjNr	ProjName
13	Intranet
21	Webauftritt

Funktionale Abhängigkeiten:

- ✓ PersID → Name
- ✓ ProjNr → ProjName
- ✓ PersID, ProjNr → • Stunden

# Problem 3

## Mitarbeiter

 PersID	<del>Q</del> Nachname	<del>Q</del> Geb.Dat	<del>Q</del> Wohnort	<del>Q</del> AbtNr	<del>Q</del> Abtname	<del>Q</del> AbtLtr
1	Retisch	12.3.1986	Wetzlar	1	Vertrieb	Schmitt
2	Ellmann	23.6.1976	Limburg	1	Vertrieb	Schmitt
3	Lador	11.11.1995	Koblenz	2	Service	Werner

Redundanz  
Trotz 2. NF

Definition 2. NF: alle NSA sind vom Primärschlüssel voll funktional abhängig.

Funktionale Abhängigkeiten:

PersID → Nachname, Geb.Dat, Wohnort, AbtNr, AbtName, AbtLtr

AbtNr → AbtName, AbtLtr

**Transitive Abhängigkeiten:**

PersID → AbtNr → AbtName

PersID → AbtNr → AbtLtr

# Dritte Normalform

Eine Relation ist in der dritten Normalform (3NF), wenn

- sie in der 2. NF ist und
- kein Nicht-Schlüssel-Attribut von einem anderen Nicht-Schlüssel-Attribut funktional abhängig ist (z.B.: transitive Abhängigkeit)

Nutzen:






- Verbliebene thematische Durchmischungen in der Relation behoben: nach der 3NF sind die Relationen des Schemas zuverlässig monothematisch.

Korrekturen unseres Beispiels:

- Auslagern der Abteilung in eigene Relation

# Problem 3 – Lösung (3. NF)

## Mitarbeiter

 PersID	 Nachname	 Geb.Dat	 Wohnort	 AbtNr	 Abtname	 AbtLtr
1	Retisch	12.3.1986	Wetzlar	1	Vertrieb	Schmitt
2	Ellmann	23.6.1976	Limburg	1	Vertrieb	Schmitt
3	Lador	11.11.1995	Koblenz	2	Service	Werner

Fremdschlüssel

## Abteilung

AbtNr	Abtname	AbtLtr
1	Vertrieb	Schmitt
2	Service	Werner

## Definition – Funktionale Abhängigkeit

$R(A_1, \dots, A_n)$  sei eine Relation sowie  $X$  und  $Y$  Teilmengen der Attributmenge  $\{A_1, \dots, A_n\}$ .

Falls aus der Gleichheit der Tupel in  $X$  stets die Gleichheit der Tupel in  $Y$  folgt, so heißen  $X$  und  $Y$  **funktional abhängig** ( $X \rightarrow Y$ ).

Sprechweise:  $Y$  ist funktional abhängig von  $X$ .

Umgangssprachlich: Man wählt eine Menge von Attributen  $X$  und eine Menge von Attributen  $Y$ . Wenn ich nun zwei beliebige Tupel aus  $R$  auswähle und diese sind in den in  $X$  ausgewählten Attributen gleich, dann folgt, dass sie auch in den Attributen aus  $Y$  gleich sind!

Anders ausgedrückt: Wenn ich in einem Tupel die Attribute aus  $X$  kenne, sind die Attribute aus  $Y$  dieses Tupels festgelegt.



## Definition – volle funktionale Abhängigkeit

Wenn aus X kein Attribut ohne Verlust dieser Eigenschaft entfernt werden kann, so spricht man von **voller funktionaler Abhängigkeit** ( $X \rightarrow \bullet Y$ ).

Falls ( $X \rightarrow Y$ ) und ( $Y \rightarrow Z$ ) gilt, so heißen X und Z **transitiv abhängig**.

Aus  $X \rightarrow Y$  folgt nicht  $Y \rightarrow X$

# Beispiel: funktionale Abhängigkeit

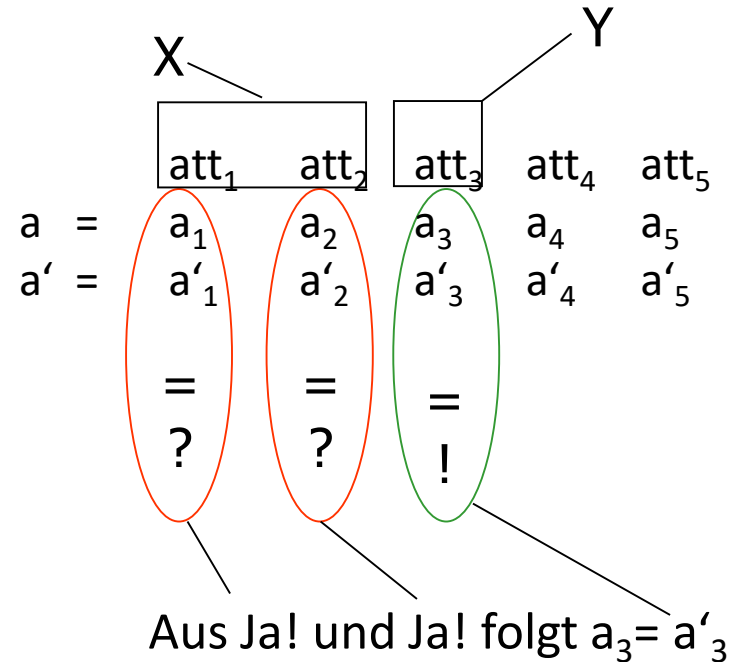
## Beispiel

Seien  $R(\text{att}_1, \dots, \text{att}_5)$  und  $X = (\text{att}_1, \text{att}_2)$  und  $Y = (\text{att}_3)$ .

Falls für alle  $a, a' \in R$  und Teiltupel  $(a_1, a_2, a_3), (a'_1, a'_2, a'_3) \in (\text{att}_1, \text{att}_2, \text{att}_3)$  von  $a$  und  $a'$  gilt

$$(a_1 = a'_1) \wedge (a_2 = a'_2) \Rightarrow (a_3 = a'_3),$$

so gilt  $X \rightarrow Y$ .



# Schlüssel redefiniert

$R(A_1, \dots, A_n)$  sei eine Relation sowie  $S$  eine Teilmenge der Attributmenge  $R = \{A_1, \dots, A_n\}$ .

Falls  $S \rightarrow R$  gilt, so heißt  $S$  ein **Superschlüssel**.

Offenbar gilt auch  $R \rightarrow R$  (**trivialer Superschlüssel**).

Eine Menge  $S$  von  $R$  mit  $(S \rightarrow \bullet R)$  heißt **Schlüsselkandidat**. Achtung: voll fA !

Umgangssprachlich: Wenn ich bei  $S$  nichts mehr wegnehmen kann, ohne dass die volle funktionale Abhängigkeit der ganzen Relation verloren geht, ist  $S$  ein Schlüsselkandidat.

Schlüsselkandidaten sind also minimale Superschlüssel.

## Weitere Normalformen

Neben den genannten Normalformen, existieren weitere Normalformen, die im betrieblichen Alltag keine oder nur sehr wenig Relevanz haben.

Eine zu starke Normalisierung von Daten führt dazu, dass die entstehenden Relationen bei der Anlage von Reports wieder umständlich durch Verbundoperationen (Joins) zusammengefügt werden müssen.

- Boyce-Codd-Normalform: keine transitiven Abhängigkeiten zwischen Schlüsselattributen
- Vierte Normalform: keine Redundanzen in funktional abhängigen Attributen / Auflösung von mehrwertigen Attributmengen
- ...

## Key Takeaways

- Relationen haben Attribute. Attribute haben einen Wertebereich (Domäne).
- Der Wertebereich einer Tabelle / Relation besteht aus dem Kreuzprodukt der Domänen ihrer Attribute.
- Primär-Schlüssel erlauben es, eindeutig Datensätze in einer Tabelle zu identifizieren.
- Ein gutes Design wird durch die Normalformen gewährleistet. Die Normalformen orientieren sich an den so genannten "funktionalen Abhängigkeiten".