Inhalt

Abschnitt 6: Grundlagen der Datenbanktechnologie

Inhalt:

- Dateien vs. Datenbanken
- Erstellung von Datenbank-Anwendungen mit Visual Basic
- Datenbanken: Tabellen, Attribute und Datentyp
- Datenmodellierung mit dem Relationenmodell
- Entwurf von Datenbanken mit dem Entity-Relationship-Modell
- Normalformen einer Datenbank
- Relationenalgebra und Structured Query Language SQL
- Tabellen und Abfragen
- Reports und Formulare

Dateien vs. Datenbanken (1)

Dateien:

- sind benannte Objekte, die eine Speicherung von Anwendungsdaten auf einem linearen Adressraum zulassen
- ***Persistente Speicherung***, d.h. Daten bleiben nach Beendigung der Anwendung und auch nach Abschalten des Systems erhalten
- Dateisystem als Teil des Betriebssystems sorgt für Zugriff mit Dateinamen, Verzeichnisse, Pufferung, Zugriffsschutz
- Datei-Daten können beliebig strukturiert sein:
  - Binärdaten
  - Textdaten (lesbarer Text)
  - XML-Datenformate (Text mit Tags zur Strukturierung)
- es werden keine Regelmäßigkeiten der abgespeicherten Datenstrukturen vorausgesetzt.

Dateien vs. Datenbanken (2)

Dateien (Fortsetzung):

- Wenn Daten eine Struktur bzw. Regelmäßigkeiten aufweisen, so wird dieser Umstand nicht vom Dateisystem gespeichert und nicht verwaltet.
- Hinweis auf Typ/Struktur der Datei über Dateityp, z.B. bild.jpg ist eine Bilddatei im JPEG-Format … aber nicht mehr
- Zugriff auf Daten in vollständiger Kontrolle des Anwendungsprogramms
- Konkurrierende Zugriffe durch mehrere Anwendungen sind nicht berücksichtigt und werden höchstens durch Sperren der gesamten Datei vermieden.

Dateien vs. Datenbanken (3)

Datenbanken:

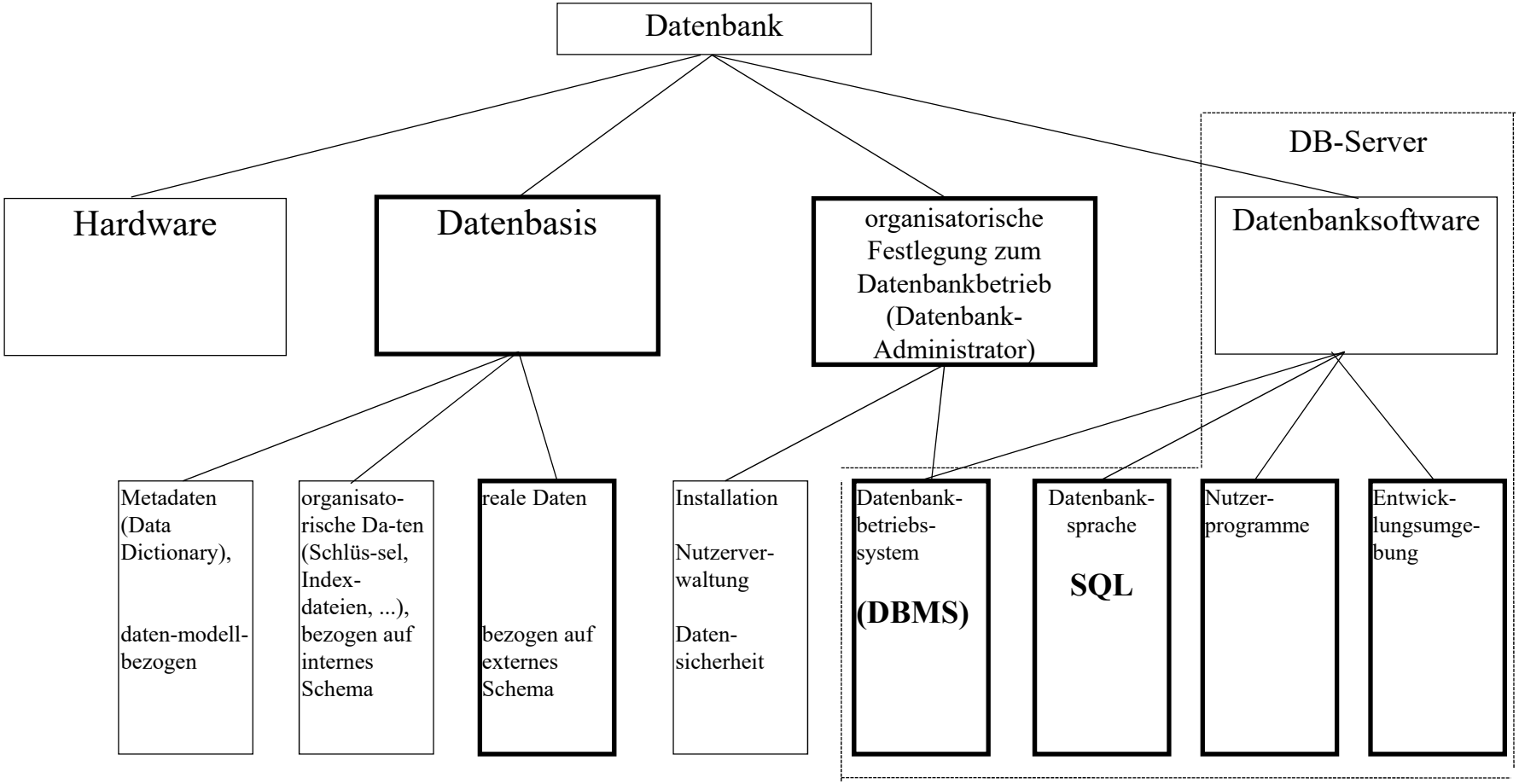
- Tabellen mit gleichartig aufgebauten Datensätzen
- Beziehungen zwischen Tabellen durch geeigneten Datenbankentwurf
- Zugriff entweder durch Satzzugriff auf einzelne Sätze einer Tabelle
- oder Zugriff durch eine Abfragesprache (SQL)
- Die Datenbank-Tabellen werden als Dateien gespeichert, oder als Image auf einem Speichermedium, ein direkter Zugriff über Datei-Operationen ist aber nicht vorgesehen.
- Die regelmäßige Struktur der Daten wird in der Datenbank gespeichert und zur Steuerung der Zugriffe ausgenutzt.

Dateien vs. Datenbanken (4)

Datenbanken (Fortsetzung):

- Ein Datenbank-Management-System dient zur Laufzeit dem Zugriff auf die Datenbank. Das sind i.d.R. Serverprozesse, die im Hintergrund ausgeführt werden.
- ***Persistente Speicherung***, d.h. Daten bleiben nach Beendigung der Anwendung und auch nach Abschalten des Systems erhalten (wie Dateien)
- Konkurrierende Zugriffe erlaubt, Verhalten bei Konflikten durch Transaktionen geregelt

Dateien vs. Datenbanken (5)



Verschiedene DBMS (Auswahl von SQL-fähigen)

MS-Access – Desktop-Datenbank im Office Packet
(für kleine DB, max. 2GB, 1024 geöffnete Tabellen, 255 Benutzer)

MS-SQL Server – Server Datenbank (Windows-Server)

Oracle – Server Datenbank (UNIX und Windows-Server)

IBM DB2 – Mainframe/Server Datenbank (zOS, OS/390, UNIX und Windows)

MySQL- im Linux-Umfeld, auch Windows (XAMPP Packet), mit stark mit Websprache PHP verbunden

Ingres – UNIX, Windows (freie Software, GPL)

Postgres - freie objekt-relationale DB, gute Anbindung an C/C++, Perl, Java, .NET ODBC

und viele weitere …

Datenbanken und Visual Basic (1)

Database Objekte

Erster Schritt: Öffnen einer Datenbank und Speichern der Zugriffsreferenz in Variablen:

```
Imports ADODB
REM ADODB als Verweis setzen
Dim CN = New ADODB.Connection
Dim RS = New ADODB.Recordset
CN.Open("kontaktedb")  ‘ODBC Quelle auf T:\\TEMP\\kontakte.accdb
```

Nun ist die Datenbank geöffnet und Zugriffe sind erlaubt.

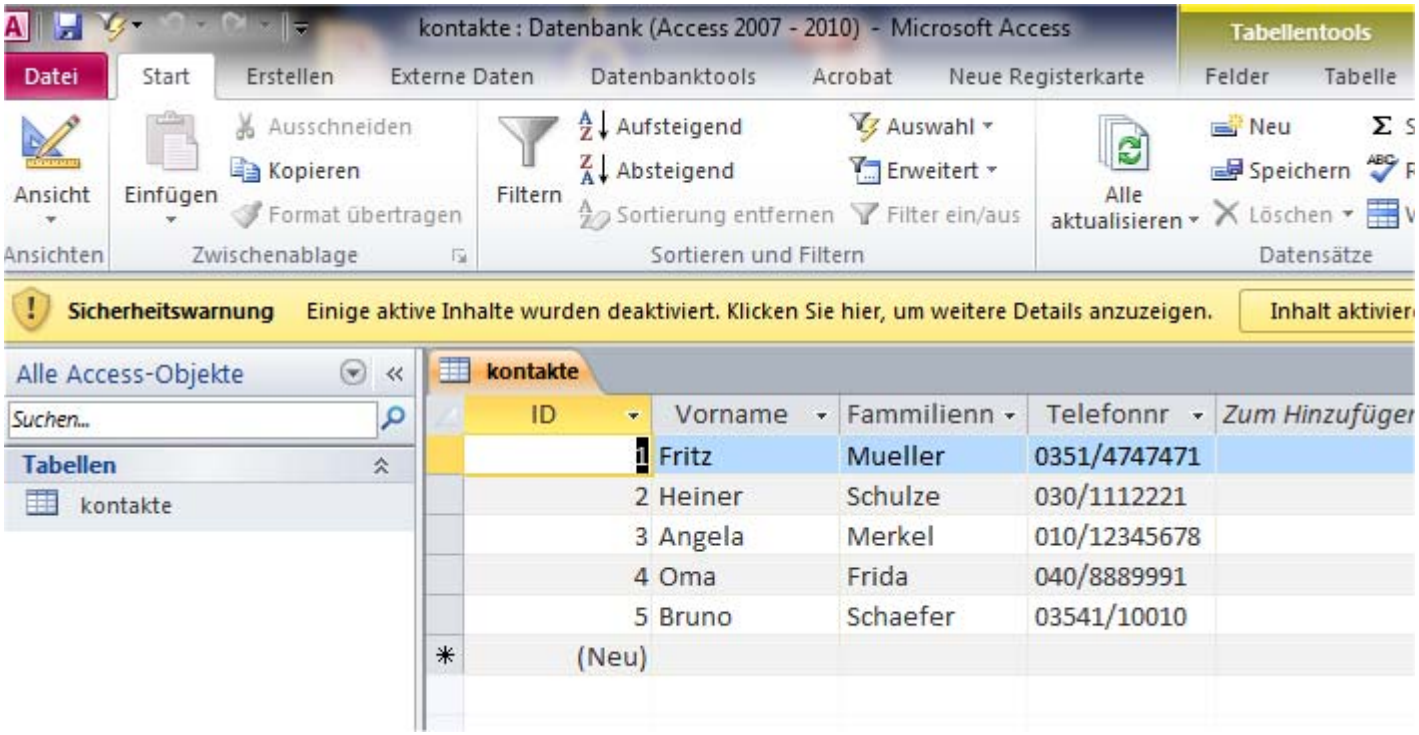
Möglicherweise besteht die Datenbank aus mehreren Tabellen.

```
RS.Open("kontakte",CN) ‘ öffnet Tabelle kontakte
```

DB und Visual Basic (2)

RecordSet Object

Gewährleistet den Zugriff auf die Tabelle ‚kontakte‘,
die in einer Access-Datenbank eingeschlossen ist.



DB und Visual Basic (3)

Zugriff auf Zeilen (Records)

Nach den Öffnen einer Tabelle (RecordSet) kann u.a. mit den Methoden *MoveFirst, MoveNext, MovePrevious, MoveLast* auf die Sätze zugegriffen werden.

Im Beispiel soll dadurch eine Listbox (ListRecords) innerhalb einer Forms-Anwendung mit Daten gefüllt werden.

Quelltext des Unterprogramms *Form_Load()* umseitig.

DB und Visual Basic (4)

Zugriff auf einzelne Zeilen (Records)

```
Dim str As String
Dim anfang As Boolean = True
If Not RS.EOF Then RS.MoveFirst()
ListBox1.BeginUpdate()
Do While Not RS.EOF
      If anfang Then ‘beim ersten mal werden Feldnamen ausgegeben
            str = ""
            For i = 0 To RS.Fields.Count - 1
                  str = str + Strings.Space(15 * i - str.Length) +
                                                RS.Fields(i).Name
            Next
            ListBox1.Items.Add(str)
            anfang = False
       End If
```

DB und Visual Basic (5)

Zugriff auf einzelne Zeilen (Records)

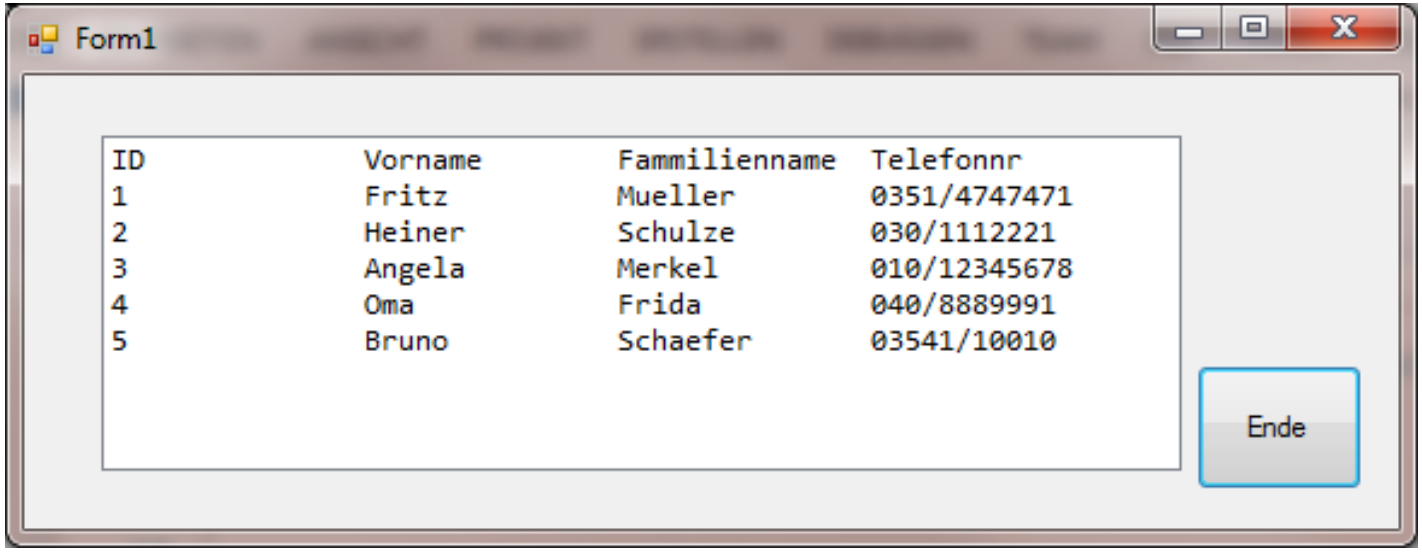
```
  str = ""
    For i = 0 To RS.Fields.Count - 1
       str = str + Strings.Space(15 * i - str.Length)
                                + Format(RS.Fields(i).Value)
    Next
    ListBox1.Items.Add(str)

    RS.MoveNext()
Loop

ListBox1.EndUpdate()
RS.Close()
CN.Close()
```

DB und Visual Basic (6)

Forms-Anwendung mit Datenbankzugriff über das Recordset



Über das Recordset-Objekt (RS) sind weitere Operationen möglich:

- Löschen eines Datensatzes (Tabellenzeile in Datenbank)
- Zufügen eines Datensatzes

Tabellen, Attribute und Datentyp

Tabellenname

Wetter

Attribut

Tupel, auch Datensatz oder Zeile

Ortsnr	**Ortsname**	**Temperatur**	**Luftdruck**
1	Dresden	22	1012
2	Leipzig	20	1010
3	Berlin	25	998
…	…	…	…

Wert mit einem speziellen Datentyp, der je Attribut definiert ist

Datenmodellierung mit dem relationalen Modell

- Definition der mathematischen Relation
- Wertebereiche, Eigenschaften und Relationenformat
- Definition der Relation im Relationalen Datenmodell
- Entität, Entitätsmenge und Primärschlüssel
- Beispiele für Entitäten
- Tabellensicht der Relation
- Definition des Relationstyps

Tabellen als so genannte Relation

Die mathematische Modellierung von Datenbanken und Zugriffsoperationen kann über das Relationale Datenmodell (Codd 1970) erfolgen. Es basiert auf dem mathematischen Relationenbegriff. Durch die Einfachheit (Tabellensicht) und klare mathematische Basis konnte sich das relationale Modell in der Entwicklung der Datenbanktechnologie seit 1970 durchsetzen.

Entity-Relationship-Modell (Chen 1976) dient zur Datenmodellierung, d.h. zur Entscheidung wie Dinge und Beziehungen auf Tabellen abgebildet werden.

Definition der mathematischen Relation (1)

Das Relationale Datenmodell (RM) basiert auf dem mathematischen Relationenbegriff. Durch die Einfachheit (Tabellensicht) und klare mathematische Basis konnte sich das RM in der Entwicklung der Datenbanktechnologie seit 1970 durchsetzen.

(mathematische) Definition der Relation:
Gegeben seien die nichtleeren Mengen W_1, W_2, ... W_n mit

$W_1 x W_2 x ... x W_n = \{(w_1, w_2, ..., w_n) \mid w_i \in W_i \;\; (i=1,2,...,n)\}$

Dann ist jede nichtleere Teilmenge $R \subseteq W_1 x W_2 x ... x W_n$ eine n-stellige **Relation** über W_1, W_2, ..., W_n . Die W_i müssen nicht alle paarweise verschieden sein.

Ein **Tupel** $(w_1, w_2, ..., w_n)$ =
(4989; Möbius; Wilhelm; Ja; 01.01.66; 80686; gesch; 2; DBADM; 4.360,50).
könnte z.B. zu R gehören. Als Trennzeichen zwischen den Tupelwerten wurde das Semikolon verwendet, da das Komma in Dezimalzahlen vorkommen kann.

Definition der mathematischen Relation (2)

(4989; Möbius; Wilhelm; Ja; 01.01.66; 80686; gesch; 2; DBADM; 4.360,50).

Für das obige Beispiel wären die Mengen W_i konkret festgelegt:
W_1 ganze Zahl (int), W_2 Zeichenkette (varchar), W_4 Besonderes (bit), W_5 Datum und Zeit (datetime), …, W_{10} Währung (money).

Wertebereiche, Eigenschaften und Relationenformat (1)

Wir deuten jetzt die Mengen $\mathbf{W_i}$ als **Wertebereiche** (Domänen) und die $\mathbf{w_i} \in \mathbf{W_i}$ als **Werte** und verbinden die $\mathbf{W_i}$, genauer das Kreuzprodukt $W_1 x W_2 x ... x W_n$, mit einer Interpretation I

$$E=\{E_1,E_2,...,E_n\} = I(W_1 x W_2 x ... x W_n) \quad \text{für } i=1,2,...,n$$

Diese $\mathbf{E_i}$ werden als **Eigenschaften** (Attribute) von **Entitäten** gedeutet.

Für das obige Beispiel wären die Eigenschaften E_i wie folgt gedeutet:
E_1 MNR (Mitarbeiternummer), E_2 Name (Familienname),
E_4 Geschlecht , E_5 Geburtsdatum, … ,E_{10} Gehalt.

Mit dieser Semantik würde das Tupel
(4989; Möbius; Wilhelm; Ja; 01.01.66; 80686; gesch; 2; DBADM; 4.360,50)
die Entität „Herr Möbius“ eindeutig charakterisieren.

Wertebereiche, Eigenschaften und Relationenformat (2)

E nennt man **Relationenformat**.
Das Relationenformat bildet den Kern einer Relation im Sinne der Datenbanktheorie. Zusammen mit der Interpretation definiert das Relationenformat eine Relation im Sinne des Relationalen Datenmodells. Manche Autoren betrachten als Relationenformat gleich die Menge der Paare (E_i,W_i) für $i=1,2,...,n$.

An den Begriff Relation werden aber noch einige Forderungen gestellt. Es ist möglich, dass ein Eigenschaftswert fehlt, d.h. undefiniert ist. Dieser Fall wird durch den (Sonder-)Wert **NULL** angezeigt.

Definition der Relation im Relationalen Datenmodell

Eine Relation $R= R(E_1,E_2,...,E_n)$ ist eine Menge von Tupeln, wobei R der Name der Relation, $E=\{E_1,E_2,...,E_n\}$ das Relationenformat und die E_i (i=1,2,...,n) die Eigenschaften sind, deren Werte w_i aus den zugeordneten Wertebereichen W_i stammen müssen.

Die E_i , die zu einer Relation gehören, müssen paarweise verschieden sein.

Ist r ein Tupel der Relation, bezeichnet $w_i = r.E_i$ den Wert der i-ten Eigenschaft des Tupels $r \in R \subseteq (W_1 x W_2 x ... x W_n)$. n heißt Grad der Relation.

Charakteristische Eigenschaften von Datenbank-Tabellen

Eine Tabelle hat folgende **charakteristische Eigenschaften**:

1. Die Zeilen (enthaltene Datensätze) sind eine Menge. Damit gibt keine paarweise identischen Datensätze.
   Die Tabelle hat deshalb einen Primärschlüssel, der jede Zeile eindeutig identifiziert.

2. Die Reihenfolge der Zeilen in der Tabelle ist ohne Belang (wegen Menge). Durch Sortierung und Gruppierung kann beim Zugriff eine gewünschte Reihenfolge erzeugt werden.

3. Die Reihenfolge der Spalten einer Tabelle ist ohne Belang.
   Spalten können bei der Abfrage nach Wunsch angeordnet werden.

4. Die Einträge in ein Tabellenfeld müssen einfach sein,
   d.h. dürfen nicht mengenwertig oder nicht strukturiert sein.
   Das Weglassen eines Eintrags (Wert: NULL) ist aber möglich.

Primärschlüssel einer Tabelle (1)

Primärschlüssel PS :

Ein Teil der Spaltenwerte in einer Tabelle stellt sich oft als identifizierend für die Zeilen heraus.

PS $\subseteq$ Menge über die Spaltenüberschriften

- PS ist die einzelne identifizierende Eigenschaft bzw. eine Eigenschaftskombination, d.h. jeder Wert des PS verweist eindeutig auf einen Datensatz.
- PS sollten minimal sein, d.h. nur die eine oder so wenige Eigenschaften enthalten, damit (gerade so) eine eindeutige Identifikation der Datensätze entsteht.
- Oft werden derartige Schlüssel schon durch organisatorische Maßnahmen unabhängig von der Datenbank eingeführt. Das sind zum Beispiel Matrikelnummern, Kundennummern, Artikelnummern oder Zugnummern.

Primärschlüssel einer Tabelle (2)

PS heißt **Primärschlüssel** der Relation $R=R(E_1,E_2,...,E_n)$, falls gilt:

1. PS $\subseteq$ E und
2. PS ist in E **identifizierende Eigenschaft** bzw. Eigenschaftskombination, d.h. jedes Tupel r $\in$ R hat einen eindeutigen PS-Wert ps_r=r.PS und $ps_r \neq$ NULL.

Ein Primärschlüssel PS heißt **minimal**, falls es keine echte Teilmenge PS‘ $\subset$PS gibt, für die die obigen Beziehungen 1. und 2. gelten.

Entität, Entitätsmenge

Eine **Relation** R im RM beschreibt eine **Entitätsmenge**. Jedes Tupel r ϵ R charakterisiert damit eindeutig eine **Entität** im zu modellierenden Diskursbereich.

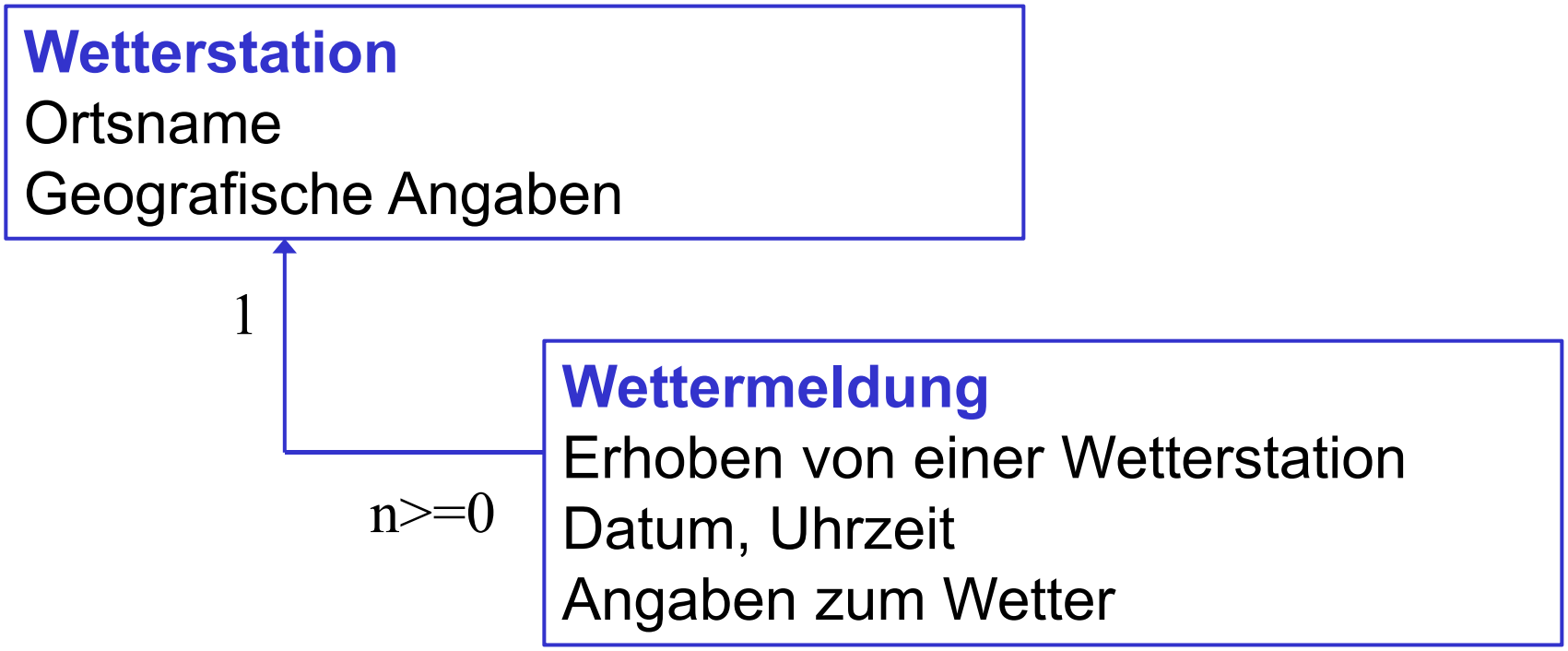
Eine **Entität** ist eine Person, Sache oder gedankliches Konstrukt, das im zu modellierenden Diskursbereich real existiert und eindeutig identifizierbar ist. Eine Entität ist Träger von Eigenschaften $E_1,E_2,...,E_n$.

Eine **Entitätsmenge** ist die Zusammenfassung aller Entitäten mit der gleichen Menge von Eigenschaften $\{E_1,E_2,...,E_n\}$, die **zu einer bestimmten Zeit** im Diskursbereich existieren. Eine Relation R mit dem Relationenformat E eignet sich zur Beschreibung einer Entitätsmenge.

Man beachte, dass damit eine Relation während der Lebensdauer einer Datenbank **zeitabhängig** ist.

Ein Beispiel zum Entitätsbegriff (1)

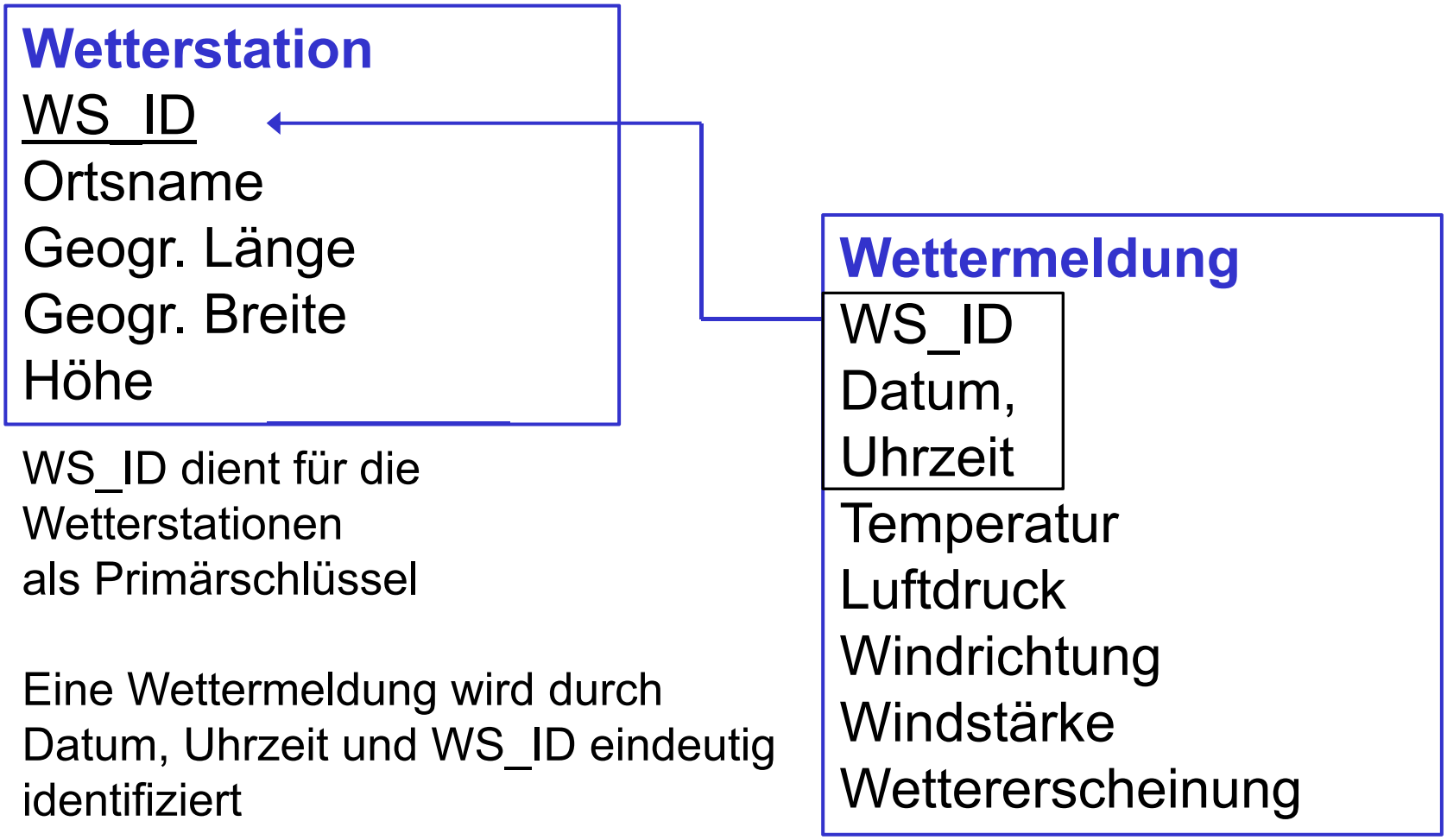
Beispiel eines **Diskursbereiches Wetterdatenerfassung**



Zusatzwissen: Eine Wetterstation wird typischerweise mehrere Wettermeldungen abgeben

Ein Beispiel zum Entitätsbegriff

Beispiel eines **Diskursbereiches Wetterdatenerfassung**



WS_ID dient für die Wetterstationen als Primärschlüssel

Eine Wettermeldung wird durch Datum, Uhrzeit und WS_ID eindeutig identifiziert

Ein Beispiel zum Entitätsbegriff (3)

wetterstationen

WS_ID	Ortsname	GeogrLaenge	GeogrBreite	Hoehe
1	Arkona	13.437	54.676	43
2	Brocken	10.615	51.799	1141
3	Fichtelberg	12.954	50.429	1214
4	Heiligendamm	11.843	54.142	10
	…			

meldungen

WS_ ID	Datum	Uhrzeit	Temp	Druck	Windr	Windst	Erscheinung
3	24-12-14	12:00	-4	998	West	7	Schneefall
1	1-5-15	15:00	17	1011	Nordwest	4	
3	12-5-15	8:00	12	978	Sued	5	Regen
…	…						

Definition des Relationentyps

Da eine **Relation R** im RM eine **Entitätsmenge** beschreibt und damit zeitabhängig ist, besteht die Notwendigkeit, eine zeitunabhängige Beschreibung von R vorzunehmen. Mit dem Begriff des Relationstyps wird diese zeitunabhängige, formale Beschreibung einer Relation im RM realisiert.

Definition des Relationstyps:

R(E | IB) heißt **Relationstyp** der Relation R wenn gilt:

- **R** ist der Name der Relation
- **E** = $\{E_1,E_2,...,E_n\}$ ist das Relationenformat von R
- E_i sind die Eigenschaften von R, deren Werte aus den zugeordneten Wertebereichen W_i stammen müssen
- **IB** ist eine Menge von <u>semantischen</u> Integritätsbedingungen (constraints)

In der Datenbanktechnologie sind neben der semantischen Integrität noch die operationale und physische Integrität von Bedeutung.

Entwurf von Relationalen Datenbanken (1) (mit dem Entity-Relationship-Modell)

In der Regel werden Diskursbereiche durch mehrere Relationen (Tabellen) abgebildet.

Ziele:

- Vermeiden von Redundanz in Relationen
- Vermeiden von Inkonsistenzen
- Effizienzsteigerung, wenn nur wirklich benötigte Daten gelesen werden, bzw. Informationen nur in wenigen Tupeln aktualisiert werden müssen.

Beispiel: eine Datenbank über Verkäufe

Verkaufs-Nr.	Produkt	Einzelpreis	Anzahl	Käufer	Käufer-Adresse
…	…	…	…	…	…

Entwurf von Relationalen Datenbanken (2)

Gesamt

Verk.-Nr.	Produkt	Einzelpreis	Anzahl	Käufer	Käufer-Adresse
1	Tomate	0.20	4	Heidi	33333 Wiesenhain, Am Waldrand 1
2	Apfelsaft	1.40	2	Heidi	33333 Wiesenhain, Am Waldrand 1
3	Katzenfutter	2.30	1	Kurt	33221 Bachhagen, Hauptstr. 23
4	Apfelsaft	1.40	3	Kurt	33221 Bachhagen, Hauptstr. 23
5	Gurke	0.55	1	Peter	33200 Feldstadt Am Markt 5
6	Tomate	0.20	8	Peter	33200 Feldstadt Am Markt 5
7	Bier	0.78	4	Kurt	22331 Bachhagen, Hauptstr. 23
…	…	…	…	…	…

Redundanzen und Inkonsistenz

Entwurf von Relationalen Datenbanken (2)

Verkaeufe

Verk.-Nr.	Produkt-ID	Anzahl	Käufer-ID
1	T	4	42
2	A	2	42
3	K	1	20
4	A	3	20
5	G	1	17
6	T	8	17
7	B	4	20
…	…	…	…

Produkte

Produkt-ID	Name	Preis
T	Tomate	0.20
A	Apfelsaft	1.40
K	Katzenfutter	2.30
G	Gurke	0.55
B	Bier	0.78
…		…

Kunden

Käufer-ID	Name	Adresse
17	Peter	33200 Feldstadt Am Markt 5
20	Kurt	33221 Bachhagen, Hauptstr. 23
42	Heidi	33333 Wiesenhain, Am Waldrand 1
…	…	…

- Redundanz ist jetzt beseitigt
- Inkonsistenzen können jetzt nicht mehr auftreten

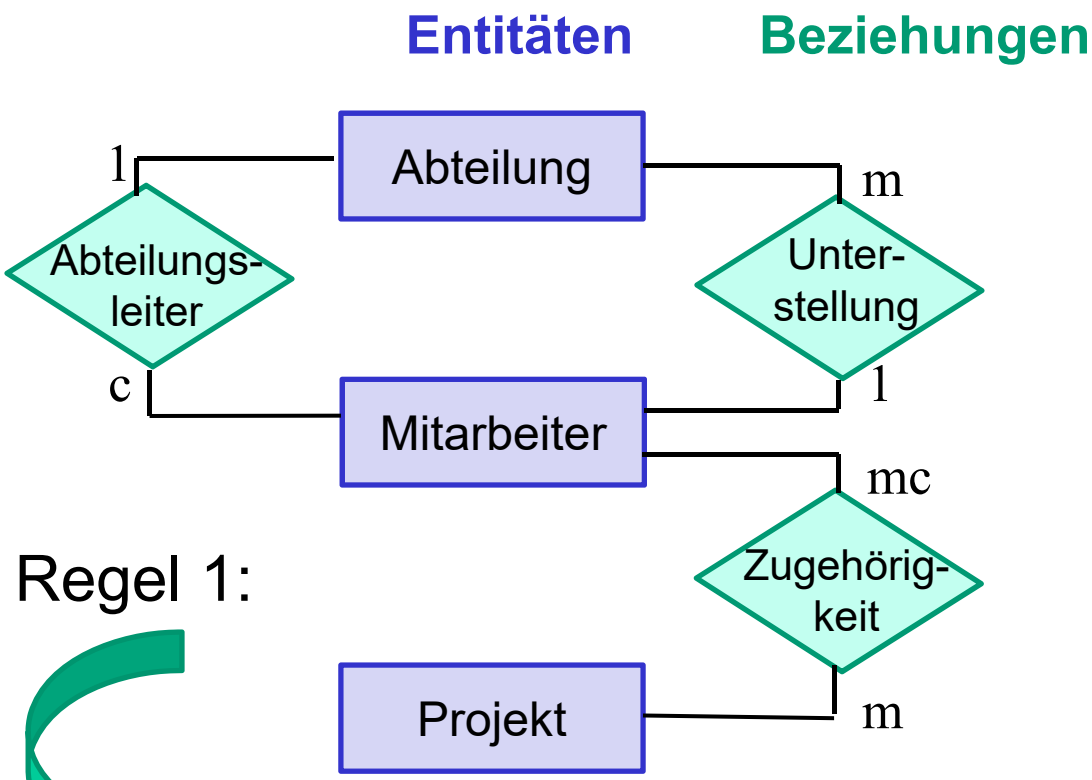
Entwurfsregeln (1)

Identifikation der Entitätsmengen und der Beziehungsmengen

Regel 1: Jede Entitätsenge muss erst einmal als eigenständige Tabelle mit einem eindeutigen Primärschlüssel definiert werden. Merkmale gehen als Attribute in die Tabellen ein.

Regel 2: Beziehungsmengen können als eigenständige Tabellen definiert werden. Die Merkmale der Beziehungen erscheinen in den Attributen der Tabelle. Der Verweis aus die Entitätsmengen geschieht durch die Aufnahme der Schlüssel in die Tabelle (Fremdschlüssel).

Entwurfsregeln (2)



Beispiel aus:

A. Meier:
Relationale Datenbanken – Leitfaden für die Praxis.
5. Auflage, 2004 Springer Verlag

Regel 1:

Einzelne Tabellen für Entitäten

Abteilung

Anr	Bezeichnung
…	…

Mitarbeiter

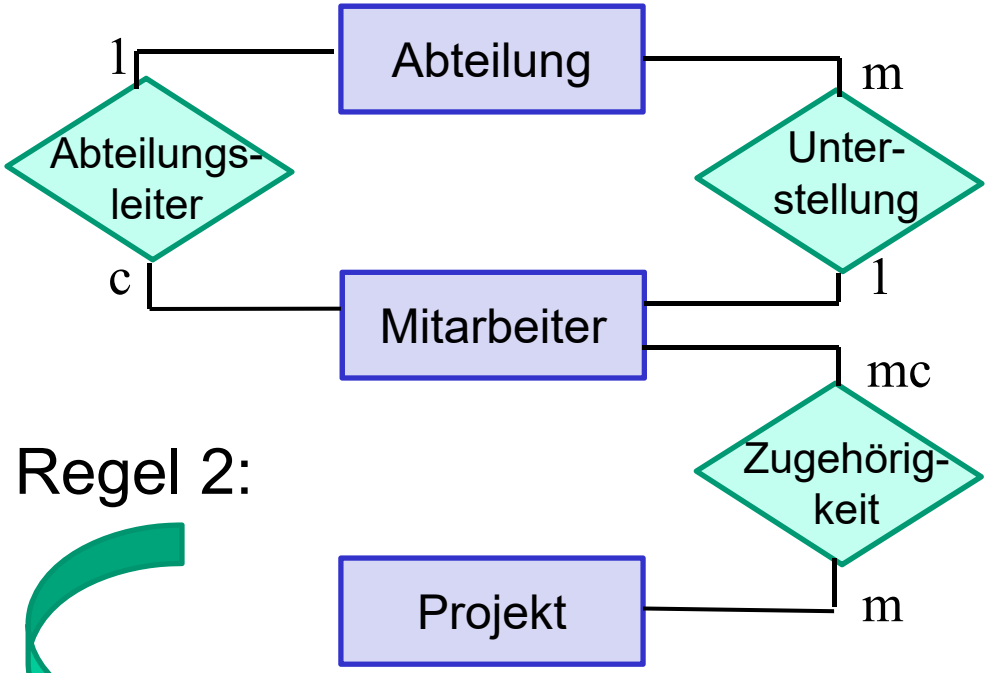
Mnr	Name	Vorname
…	…	…

Projekt

Pnr	Bezeichnung
…	…

Entwurfsregeln (3)

Entitäten **Beziehungen**



Regel 2:

Tabellen oder
Fremdschlüsselbeziehungen
für Beziehungen

Assoziationstypen:

1 – genau ein
c – kein oder ein
m – ein oder mehrere
mc – kein, ein oder mehrere

Entwurfsregeln (4)

Beziehungs-Komplexität:

Die Mächtigkeit der in Beziehung stehenden Mengen definiert verschiedene Klassen:

von/zu	1	c	m	mc
1	(1,1)	(1,c)	(1,m)	(1,mc)
c	(c,1)	(c,c)	(c,m)	(c,mc)
m	(m,1)	(m,c)	(m,m)	(m,mc)
mc	(mc,1)	(mc,c)	(mc,m)	(mc,mc)

B1

B2

B2

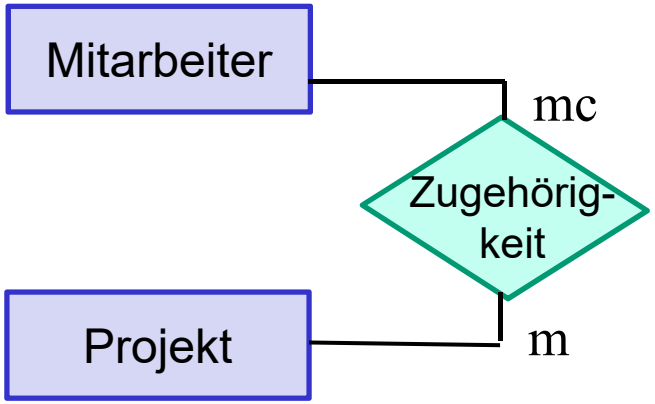
B3

B1: einfach-einfache Beziehungen
B2: einfach-komplexe Beziehungen
B3: komplex-komplexe Beziehungen

Entwurfsregeln (5)

Regel 3:

Komplex-komplexe Beziehungen (m zu m, m zu mc, mc zu m, mc zu mc) müssen als eigenständige Tabelle definiert werden.



Mitarbeiter

Mnr	Name	Vorname
...	...	...

Zugehoerigkeit

Mnr	Pnr	Prozent
...	...	...

Projekt

Pnr	Bezeichnung
...	...

Entwurfsregeln (6)

Regel 4:
einfach-komplexe Beziehungen (m zu 1, 1 zu m und weitere) können ohne eigenständige Tabelle durch die beiden zugeordneten Tabellen der Entitätsmengen ausgedrückt werden.

Fremdschlüssel-Beziehung

Abteilung

Anr	Bezeichnung
…	…

Mitarbeiter

Mnr	Name	Vorname	Abt_Unterstell
…	…	…	

Entwurfsregeln (7)

Regel 5:
einfach-einfache Beziehungen (1 zu 1, 1 zu c und weitere) können ohne eigenständige Tabelle durch die beiden zugeordneten Tabellen der Entitätsmengen ausgedrückt werden.

Fremdschlüssel-Beziehung

Abteilung

Anr	Bezeichnung	Abtleiter
…	…	

Mitarbeiter

Mnr	Name	Vorname	Abt_Unterstell
…	…	…	

Fremdschlüssel-Beziehung

Entwurfsregeln (8)

Regel 6:
Generalisierung – Eigenständige Tabellen für übergeordnete Entitätsmenge und für jede Spezialisierung

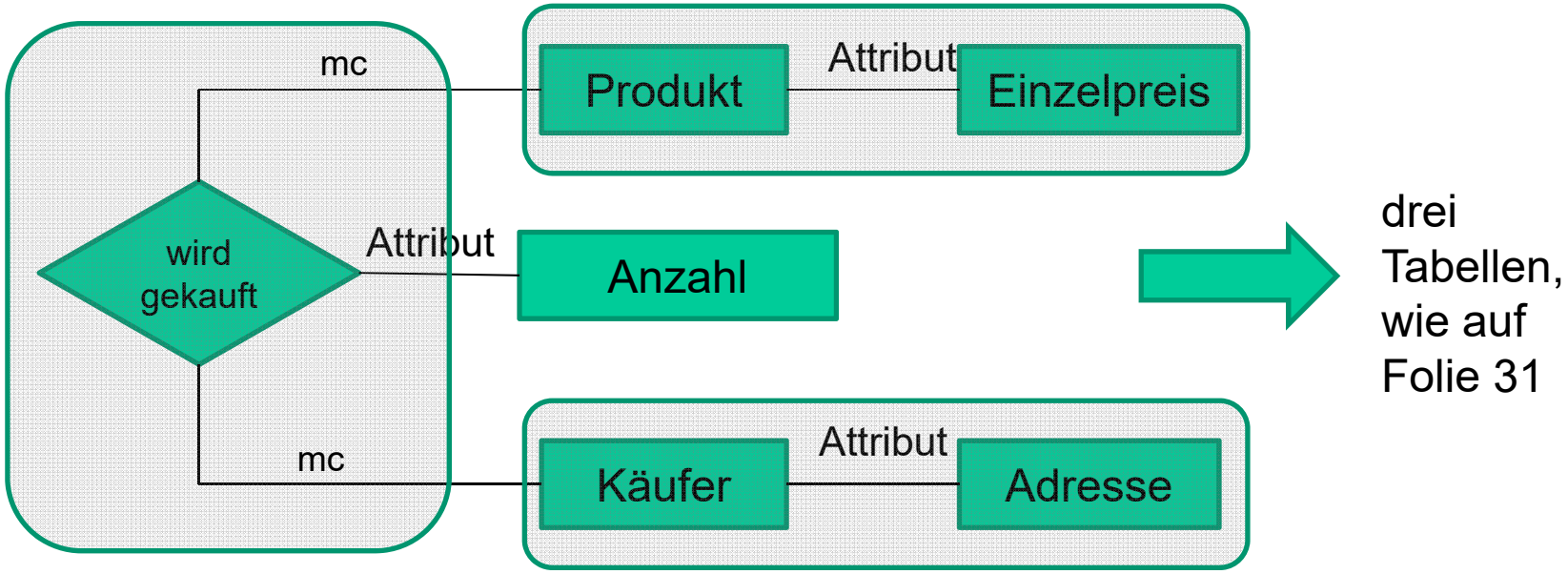
Regel 7:
Part-Of-Beziehung - Eigenständige Tabellen für Entität und Beziehungsmenge, falls Beziehungstyp komplex-komplex ist

… hier nur zur Vollständigkeit erwähnt.

Entwurfsregeln (9)

Wie wäre ein regelbasierter Entwurf für das eingangs benutzte Beispiel ausgegangen?

Verk.-Nr.	Produkt	Einzelpreis	Anzahl	Käufer	Käufer-Adresse
…	…	…	…	…	…



Normalformen (1)

Ziele:

- **Vermeiden von Redundanz in Relationen** – bei ungünstiger Struktur der Tabellen würde gleiche Information u.U. mehrfach gespeichert. Eine Strukturierung entsprechend der dritten Normalform verhindert das.
- **Vermeiden von Anomalien** – bei ungünstigen Entwurf können Informationen u.U. ungewollt verloren gehen.

Normalformen (2)

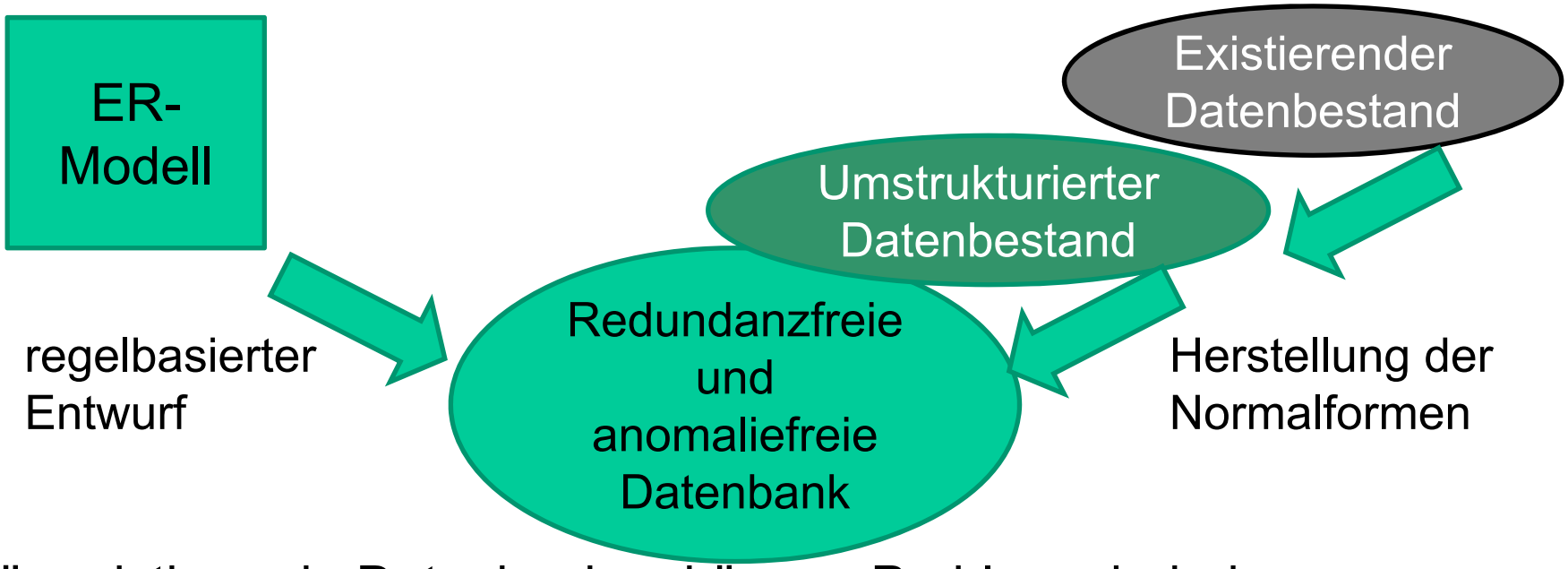
Verk.-Nr.	Produkt	Einzelpreis	Anzahl	Käufer	Käufer-Adresse
…	…	…	…	…	…

Anomalien:

- Einfügeanomalie – Neue Sachverhalte können nicht eingefügt werden. Im Beispiel oben kann keine neues Produkt mit einem Preis eingefügt werden, ohne einen Kaufvorgang zu beschreiben.
- Löschanomalie – Das Löschen von Kaufvorgängen bedingt u.U. den Verlust von Produktinformationen und Käuferdaten.
- Änderungsanomalie – Das Abändern eines Sachverhalts verlang oft das Ändern vieler Datensätze (bei Redundanz von z.B. Käufer-Adressen)

Zweck der Normalformen

Ein Datenbankentwurf nach dem Entity-Relationship-Modell bringt die Tabellen automatisch in eine redundanzfreie und anomaliefreie Struktur.



Für existierende Datenbanken können Probleme behoben werden, indem das Vorhandensein der Normalformen geprüft wird und diese durch Umstrukturierung der Tabellenstruktur hergestellt werden.

Erste Normalform (1)

Eine Tabelle ist in der ersten Normalform (1NF), falls die Wertebereiche der Merkmale atomar sind.

1NF verlangt, dass jedes Merkmal Werte aus einem unstrukturierten Wertebereich aufnimmt. Damit dürfen keine Mengen, Aufzählungstypen oder Wiederholungsgruppen in den einzelnen Merkmalen vorkommen.

Beispiel für eine nicht erfüllte 1NF:

Matrnr	Name	Vorname	Studiengänge
11	Schmidt	Robert	{Mathematik, Physik}
47	Schulze	Klaus	{Sport, Erziehungswissenschaften}
78	Peters	Karoline	{Informatik}

Erste Normalform (2)

Beispiel mit atomarem Wertebereich für Studiengang

Matrnr	Name	Vorname	Studiengang
11	Schmidt	Robert	Mathematik
11	Schmidt	Robert	Physik
47	Schulze	Klaus	Sport
47	Schulze	Klaus	Erziehungswissenschaften
78	Peters	Karoline	Informatik

Zweite Normalform (1)

Eine Tabelle ist in der zweiten Normalform (2NF), wenn sie in der 1NF ist und wenn jedes Nichtschlüsselmerkmal von jedem Schlüssel ***voll funktional abhängig*** ist.

Voll funktional abhängig bezieht sich aus zusammengesetzte Schlüssel (S1,S2) von denen alle Nichtschlüsselmerkmale abhängig sind: (S1,S2) -> N
Sie dürfen dabei aber nicht funktional abhängig von einem Teilschlüssel sein, z.B. S1->N

Im Beispiel sind Name und Vorname Nichtschlüsselmerkmale, der Schlüssel setzt sich aus Matrnr und Studiengang zusammen.
(Matrnr, Studiengang) -> Name
aber auch Matrnr -> Name … damit keine 2NF

Zweite Normalform (2)

Herstellung der zweiten Normalform (2NF): Tabellenteilung
Alle Merkmale, die von einem Teilschlüssel abhängig sind werden in eine eigene Tabelle ausgegliedert.

Matrnr	Name	Vorname
11	Schmidt	Robert
47	Schulze	Klaus
78	Peters	Karoline

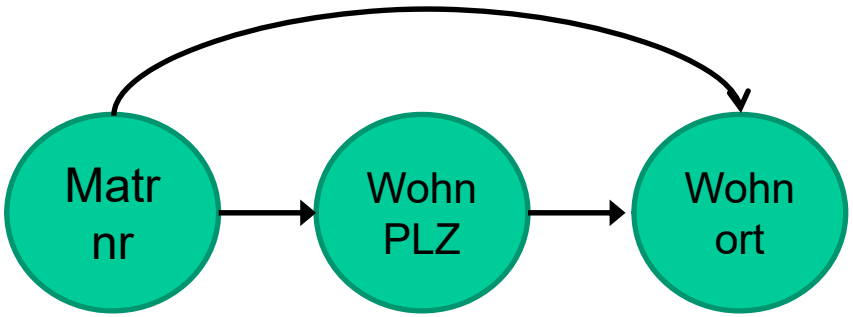
Matrnr	Studiengang
11	Mathematik
11	Physik
47	Sport
47	Erziehungswissenschaften
78	Informatik

Dritte Normalform (1)

Eine Tabelle ist in der dritten Normalform (3NF), wenn sie in der 2NF ist und kein Nichtschlüsselmerkmal von irgendeinem Schlüssel transitiv abhängig ist.

Ein Beispiel für eine nicht erfüllte 3NF:

Matrnr	Wohn-PLZ	Wohnort
11	01217	Dresden
47	01219	Dresden
78	14197	Berlin
89	14197	Berlin
95	01217	Dresden



Wohnort tritt redundant auf.

Dritte Normalform (2)

Überführung in dritte Normalform (3NF), indem ein transitiv abhängiges Merkmal in eine neue Tabelle ausgegliedert wird. Das Merkmal von dem die letzte Stufe der Abhängigkeit ausgegangen ist, wird als Fremdschlüssel in die neue Tabelle aufgenommen.

Matrnr	Wohn-PLZ
11	01217
47	01219
78	14197
89	14197
95	01217

PLZ	Ort
01217	Dresden
14197	Berlin
01219	Dresden