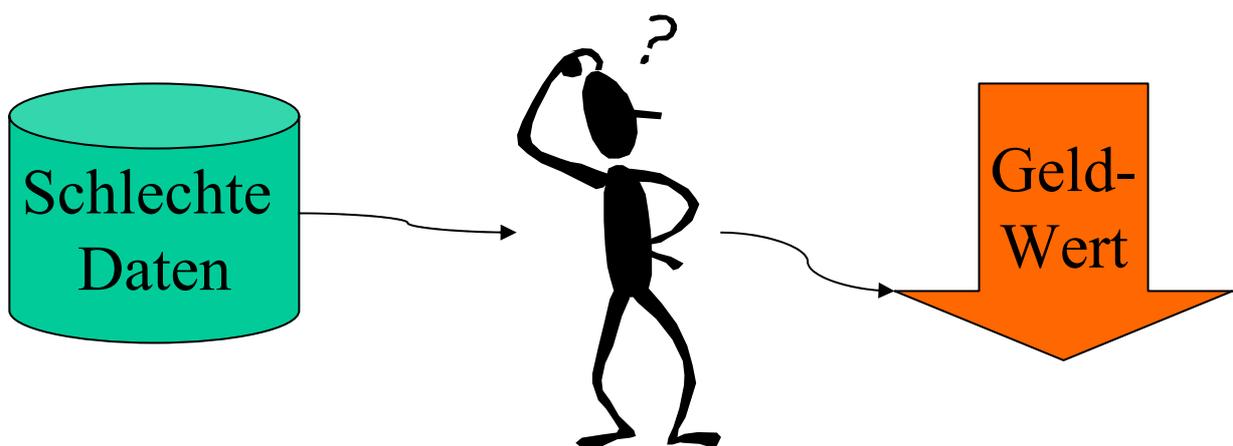


Daten <-> Informationen

- *Daten*
„...sind eine Repräsentation von Fakten, Konzepten oder auch Instruktionen in einer formalisierten Art und Weise, die sie für die Kommunikation, Interpretation und Verarbeitung durch Maschinen geeignet machen“
- *Informationen*
„beziehen sich auf die Bedeutung, die Menschen durch vereinbarte Konventionen diesen Daten zuordnen“

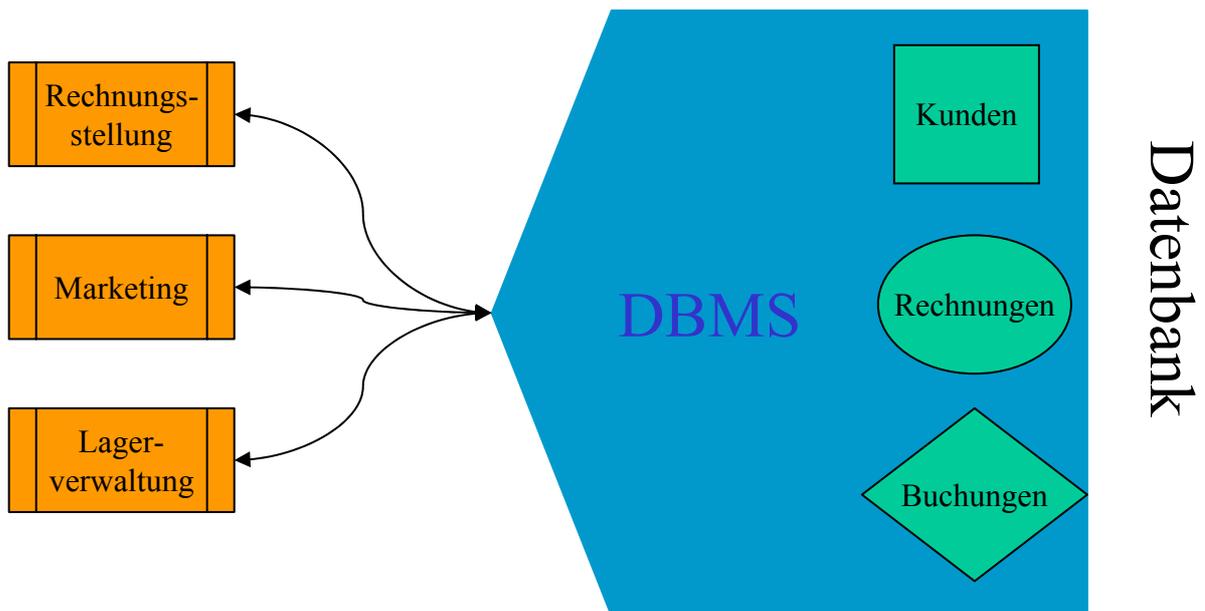
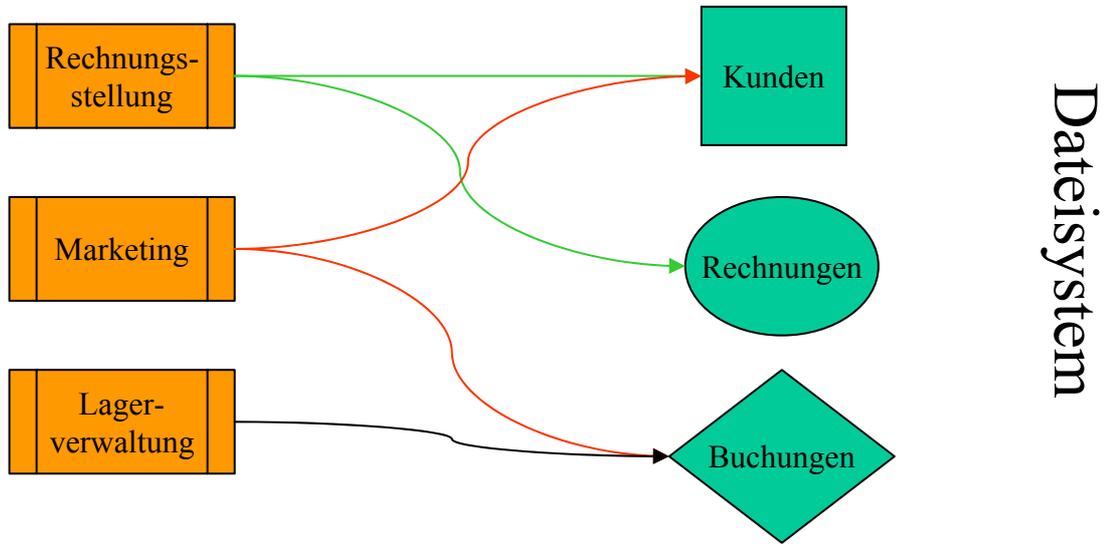
=> Informationen sind interpretierte Daten

- Wert von Informationen:
 - Informationen bilden die Grundlage für menschliche Entscheidungen
 - Informationen bringen Nutzen => Monetärer Wert



=> Hoher Stellenwert der Datenbasis

Funktionsweise einer Datenbank



Motivation für den Einsatz einer Datenbank

- **Konsistenz und Redundanz**

Datenhaltung in isolierten Dateien verursacht oft eine mehrfache Abspeicherung derselben Sache in verschiedenen Dateien.

- Erhöhter Arbeitsaufwand
- bei Änderung der Daten Gefahr von Inkonsistenzen

- **Zugriffsmöglichkeiten**

Es ist schwierig, Daten in isolierten Dateien miteinander zu verknüpfen
-> Analyse wird erschwert

- **Mehrbenutzerbetrieb**

- *Datenbank*

Mehrere Benutzer können gleichzeitig an einer Datenbank arbeiten.
Datensätze werden automatisch gesperrt.

- *Dateisystem*

Eine Datei kann nur einmal geöffnet sein

- **Verhalten in Fehlersituationen**

Bei isolierten Dateien wird die Wiederherstellung eines konsistenten Datenbestandes nach einem Crash schwierig bis unmöglich

- **Integrität**

Oft existieren Konsistenzbedingungen, die dringend eingehalten werden müssen, weil ohne sie keine Integrität der Daten gewährleistet werden kann. Bei Datenbanken werden Transaktionen nur dann vollzogen, wenn sie die Datenbasis in einen konsistenten Zustand überführen.

- **Sicherheit**

Nicht jeder Anwender soll alles machen können. Oft ist diese Unterscheidung sehr detailliert zu treffen, was ein Dateisystem in der Regel nicht leisten kann.



Motivation für den Einsatz einer Datenbank

- **Datenunabhängigkeit**

- *Datenbank*

- Das Datensystem macht die Nutzer unabhängig von der internen Speicherung der Daten.

- => Daten können von verschiedenen Programmen über genormte Schnittstellen genutzt werden

- *Dateisystem*

- Dateien sind immer vom Betriebssystem abhängig

- Stichwort: Dos2Unix

- **Transaktionskonzept**

- Zusammenhängende Aktionen dürfen nur komplett oder gar nicht ausgeführt werden.

- *Datenbank*

- „ACID“-Transaktionskonzept

- *Dateisystem*

- (kein Konzept)



Stichwort: Datenunabhängigkeit

- *Datenunabhängigkeit* meint die weitgehende Unabhängigkeit der Daten von den Benutzern und Programmen
- Mittel zum Erreichen der Datenunabhängigkeit:
3-Abstraktionsebenen-Konzept
 - internes Schema
behandelt die interne (physikalische) Abspeicherung der Datensätze. Hier sind Art und Aufbau der verwendeten Datensätze festgelegt. Interessiert nur den Administrator der Datenbank, meist bei Performance-Maßnahmen.
 - konzeptionelles Schema
logische Gesamtsicht der Daten und ihrer Beziehungen zueinander. „Datenmodell“.
Beachte: Hier nur Definition des Informationsgehaltes, keine Aussage über Speicherstruktur!
 - externes Schema
Definition darüber, wie die Benutzer die Daten präsentiert bekommen (Zugriffsrechte, Sichten, Berichte, ...)
- Beispiel: Adressverwaltung in Access
 - internes Schema: Aufbau der Bits und Bytes der MDB-Datei
 - konzeptionelles Schema: Definition der Tabellen (Hausnummer, PLZ, etc.)
 - externes Schema: Erscheinungsbild von Access, Darstellung von Tabellen, benutzerspezifische Formulare, Berichte, ...



Stichwort: Konsistenzerhaltung

- *Konsistenzerhaltung* bedeutet die widerspruchsfreie Speicherung der Inhalte in der Datenbank

- Beispiel:

Bei einer Bank wird eine Überweisung veranlaßt:

Kunde A will Kunde B einen bestimmten Betrag zahlen.

Konkret sind von der Bank zwei Aktionen auszuführen:

1. Abbuchung des Betrages von Konto A
2. Buchung des Betrages auf Konto B

Nehmen wir an, daß es zwischen diesen beiden Aktionen zu einem Ereignis (Zugriffsrechte, Plattencrash, Stromausfall, ...) kommt, so daß die zweite Aktion nicht mehr ausgeführt werden kann, so muß das System in der Lage sein, die erste Aktion rückgängig zu machen.

Dies wird erreicht durch das Transaktionskonzept, bei dem die beiden Aktionen untrennbar zu einer Transaktion zusammengefaßt werden. Eine Transaktion kann dabei immer nur ganz oder gar nicht ausgeführt werden.



Datenbankmodelle

- „Auslaufmodelle“:
 - Hierarchische DBMS:
 - eignen sich für die Darstellung von „1 zu viele“-Beziehungen
 - z.B.: Vater -> Sohn -> Kind
 - Netzwerk- DBMS:
 - am besten für „viele zu viele“ -Beziehungen
- aktuelles Modell: Relationale Datenbanken
 - Vorteile:
 - kann verschiedene Arten von Beziehungen darstellen
 - universell durch Relationenalgebra
 - sehr weit verbreitet
 - Schwachpunkte:
 - Objektidentität nicht vorhanden
 - Trennung von Eigenschaften und Operationen
- im Kommen:
 - Objektrelationale DBMS
 - Relationale DBMS erweitert um objektorientierte Ansätze. Das heißt die Daten werden nach wie vor in der bekannten Tabellenstruktur abgespeichert, durch umfangreiche Änderung der Anfragesprachen hat man aber den Eindruck, als würde man mit Objekten arbeiten.
 - Objektorientierte DBMS
 - jedes Objekt hat Identität
 - Objekt enthält Eigenschaften und Operationen !
 - Modellierung einfacher, weil der menschlichen Denkweise besser angepaßt
 - Zur Zeit noch wenig im Gebrauch, weil wenig standardisiert



Basis einer relationalen Datenbank: Relationenalgebra

- Dinge der Realwelt (Entities) und ihre Beziehungen untereinander werden durch das Konzept der Relation dargestellt
- Relation = rein mathematisches Konzept
- Relation definiert als Teilmenge eines kartesischen Produktes mehrerer endlicher Mengen
- ein Element einer Relation heißt Tupel
- Beispiel:

Menge 1	A	B
	1	2
	2	2

Menge 2	C	D	E
	1	2	3
	4	5	6
	7	8	9

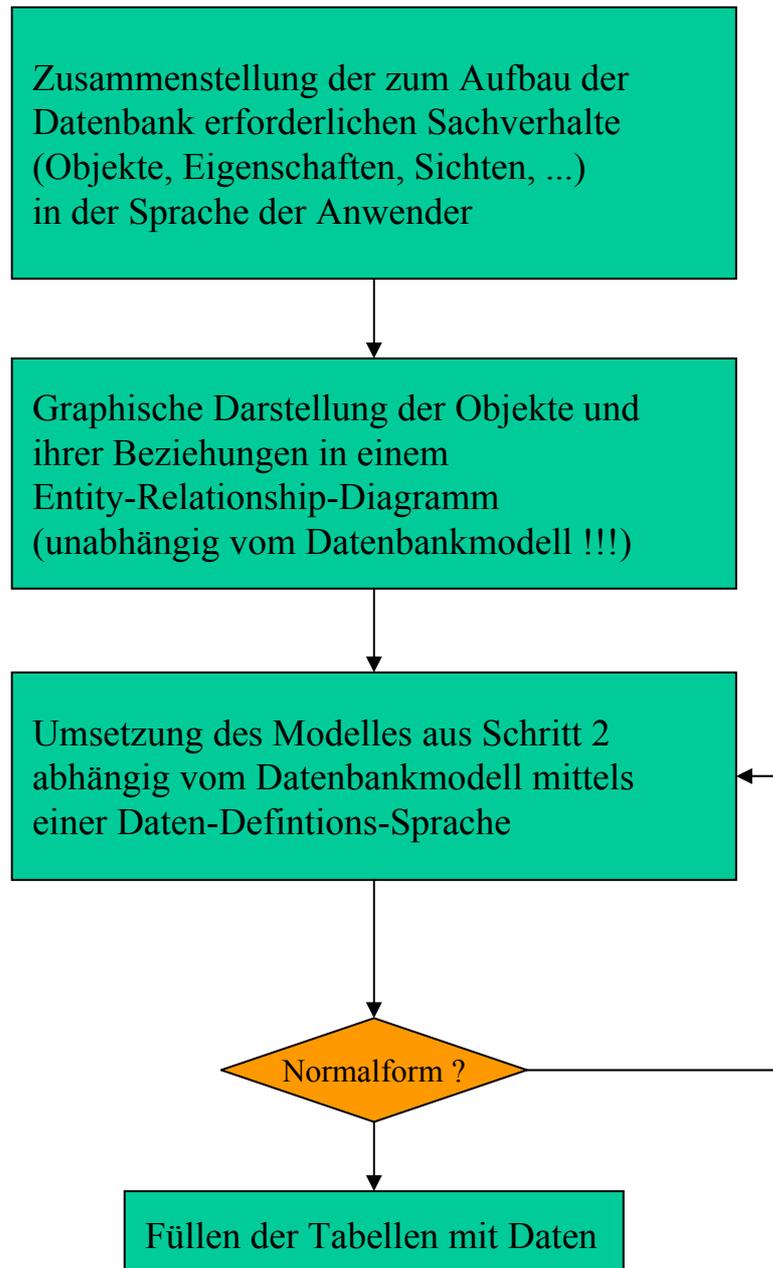
Menge 1 X Menge 2	A	B	C	D	E
	1	2	1	2	3
	1	2	4	5	6
	1	2	7	8	9
	2	2	1	2	3
	2	2	4	5	6
	2	2	7	8	9

- Man sieht:
 - Verarbeitung der Werte durch rein mathematische Regeln
 - Relation ist eine Menge von Werten, keine Liste; Es gibt keine Reihenfolge !
 - Objekt (Tupel) definiert sich durch seine Attribute, hat aber á priori keine Identität, die man direkt ansprechen könnte



Anlegen einer Datenbank

Das Anlegen einer Datenbank geschieht in vier Schritten:



Entity-Relationship-Diagramm „ER-Diagramm“

- Elemente eines Entity-Relationship-Diagrammes:
 - *Entität*:
Wohlunterscheidbares Objekt der realen Welt oder unserer Vorstellung
Beispiele für Entitäten:
Personen, Begriffe, Gegenstände, Ereignisse, ...
Entitäten gleichen Typs bilden Entitätsmengen.
 - *Beziehung*:
Stellt die Beziehung zwischen Entitäten dar
Beziehungen gleichen Typs bilden ebenfalls Beziehungsmengen
und können zusätzliche Merkmale tragen.
Beispiele:
„hat Anteil an“, „gehört zu“, „ist Teil von“, ...
 - *Attribut*:
Dient als Charakterisierung von Gegenständen oder Beziehungen
Beispiel:
„Vorname“ als Attribut für Entität „Person“
„Note“ als Attribut für die Beziehung „prüfen“ zwischen den beiden Entitäten „Professoren“ und „Studenten“



Entity-Relationship-Diagramm „ER-Diagramm“

- Graphische Darstellung:

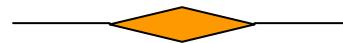
- Entität



- Attribut



- Beziehung



Beziehungen in ER-Diagrammen

- Einfache graphische Darstellung von Beziehungen :

- 1:1 - Beziehung (one-to-one)



- 1:n - Beziehung (one to many)



- m:n - Beziehung (many to many)



- Graphische Darstellung von Beziehungen mit Angabe der Funktionalität:

- 1:1 - Beziehung (one-to-one)



- 1:n - Beziehung (one to many)



- m:n - Beziehung (many to many)



- Graphische Darstellung von Beziehungen in (min,max)-Notation:

- 1:1 - Beziehung (one-to-one)



- 1:n - Beziehung (one to many)



- m:n - Beziehung (many to many)



Datendefinition

- Umsetzung des Entitäten-Beziehungs-Modells in Form von Tabellen => Datendefinition
- Unter Datendefinition versteht man die Festlegung eines Datenbankschemas, das die Struktur einer Datenbank beschreibt
 - Namen der Tabellen und ihrer Felder
 - Eigenschaften der Tabellen, Wertebereiche, Indizes, Integritätsbedingungen
- Bemerkung: Der Inhalt der Datenbank ist nicht Gegenstand der Datendefinition

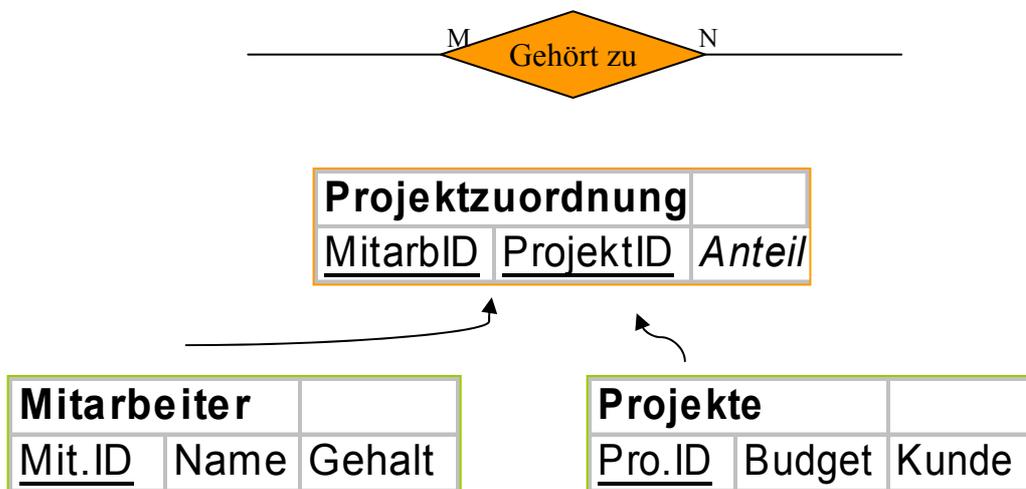
Regeln zur Überführung von Entitätsmengen in relationale Datenbankschemas:

- Regel 1:
 - Jede Entitätsmenge muß durch eine eigenständige Tabelle dargestellt werden.
 - Die Merkmale der Entitätsmenge werden dabei zu Feldern in der Tabelle. Identifikationseigenschaft der Entitätsmenge bestimmt das Primärschlüsselfeld.
- Regel 2:
 - Jede Beziehungsmenge kann durch eine Tabelle dargestellt werden, muß aber nicht. Das hängt vom Typ der Beziehung ab.
 - Auch hier werden Merkmale der Beziehungsmenge zu den Feldern der Tabelle
 - In der Regel setzt sich der Primärschlüssel der Beziehungsmenge aus den Fremdschlüssel der zugehörigen Entitätsmengen zusammen.



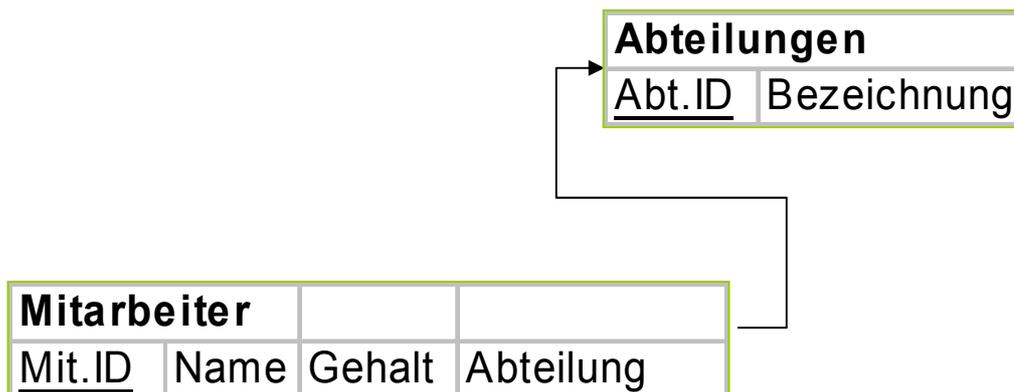
Definitionsarten für Beziehungen: M:N-Beziehungen

- Regel 3:**
 Jede M:N-Beziehung (komplex-komplex) muß über eine eigene Tabelle dargestellt werden.
 Im Allgemeinen setzt sich der Primärschlüssel aus der **Kombination** der beiden Fremdschlüssel der zugehörigen Entitätsmengen zusammen



Definitionsarten für Beziehungen: 1:N - Beziehung

- Regel 4:**
 Jede 1:N-Beziehung (einfach-komplex) kann ohne eigene Tabelle dargestellt werden.
 Dazu wird der Tabelle auf der N-Seite ein Fremdschlüssel auf die Entitätsmenge auf der 1-Seite hinzugefügt. Zusätzliche Attribute kann man hierzu passend in weiteren Spalte einfügen



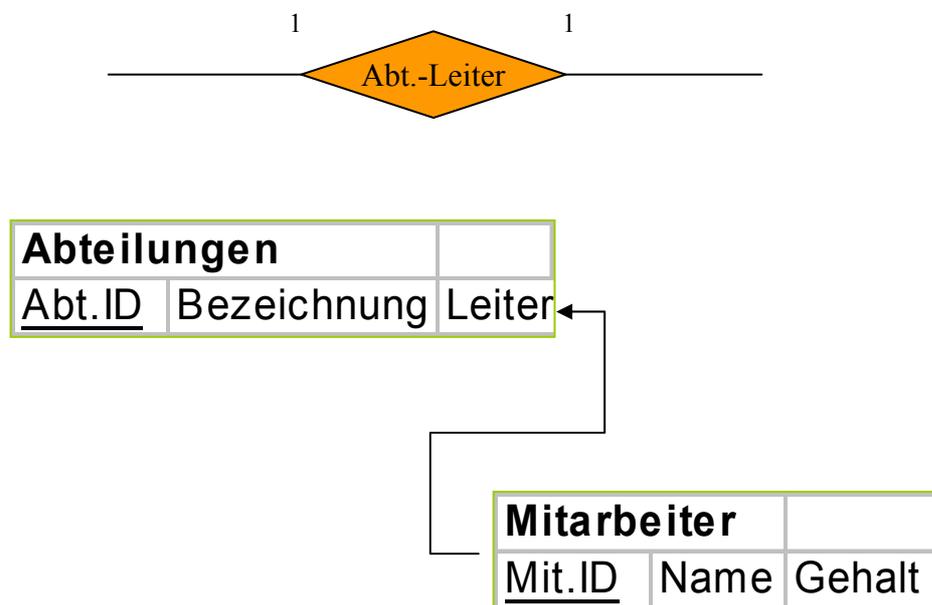
Definitionsarten für Beziehungen: 1:1 - Beziehung

- Regel 5:

Jede 1:1-Beziehung (einfach- einfach) kann ohne eigene Tabelle dargestellt werden.

Dazu wird einer Tabelle ein Fremdschlüssel auf die Entitätsmenge der anderen Seite hinzugefügt. Welche Tabelle man dafür wählt, ist letztendlich egal.

Speichertechnisch am besten: Fremdschlüssel in der Tabelle einfügen, die voraussichtlich weniger Datensätze hat.



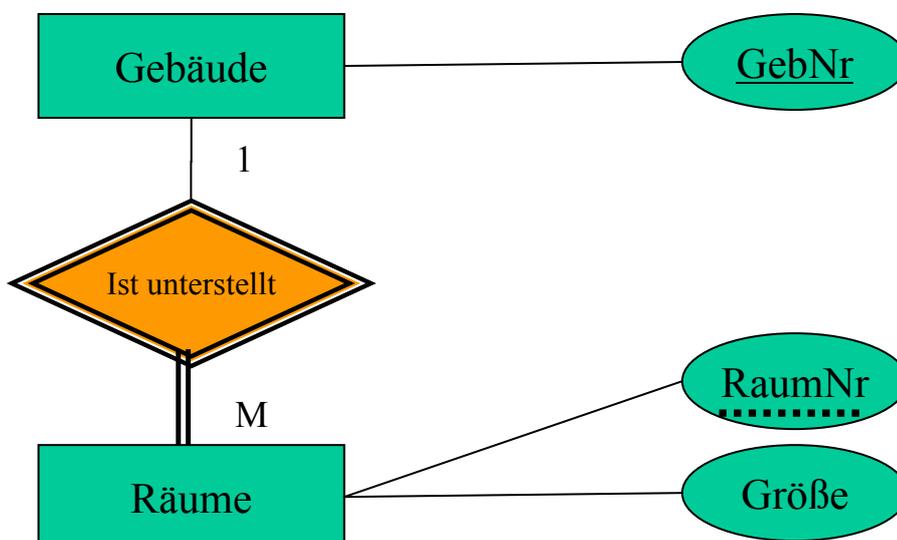
Weitere Verfeinerungen: Existenzabhängige Typen

Entities müssen nicht zwangsweise unabhängig voneinander existieren und sie müssen auch nicht über ein Schlüsselattribut eindeutig identifizierbar sein -> „schwache Entities“

Schwache Entities sind

- in ihrer Existenz von einem anderen, übergeordneten Entity abhängig
- oft nur in Kombination mit dem Schlüssel des übergeordneten Entities eindeutig identifizierbar

Beispiel: Gebäude und Räume



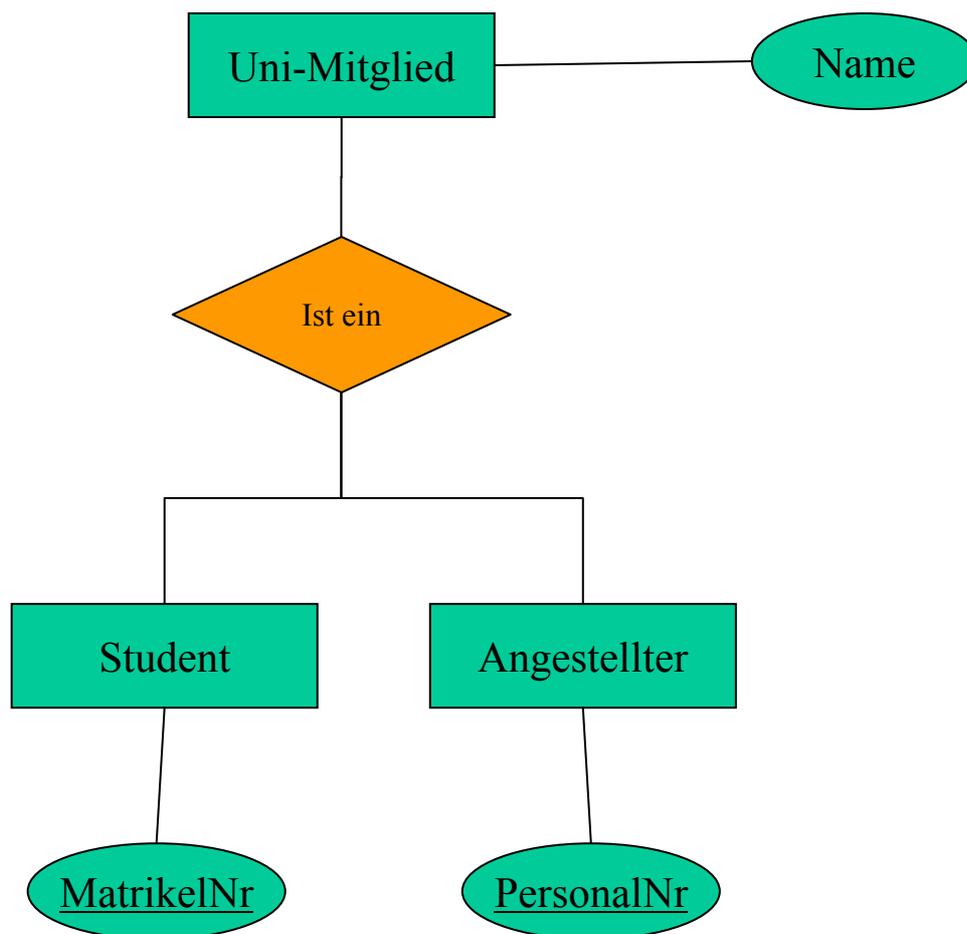
Weitere Verfeinerungen: Generalisierung

Generalisierung wird eingesetzt, um eine bessere (natürlichere und übersichtlichere) Strukturierung der Entitytypen zu erhalten

Bei der Generalisierung werden gleichartige Entitytypen strukturiert und dabei Ober- und Untertypen herausgeschält

Generalisierungen erkennt man vor allem daran, daß die Beziehung zwischen zwei Entities als eine „ist ein“-Beziehung charakterisieren läßt

Beispiel: Universitätsverwaltung:



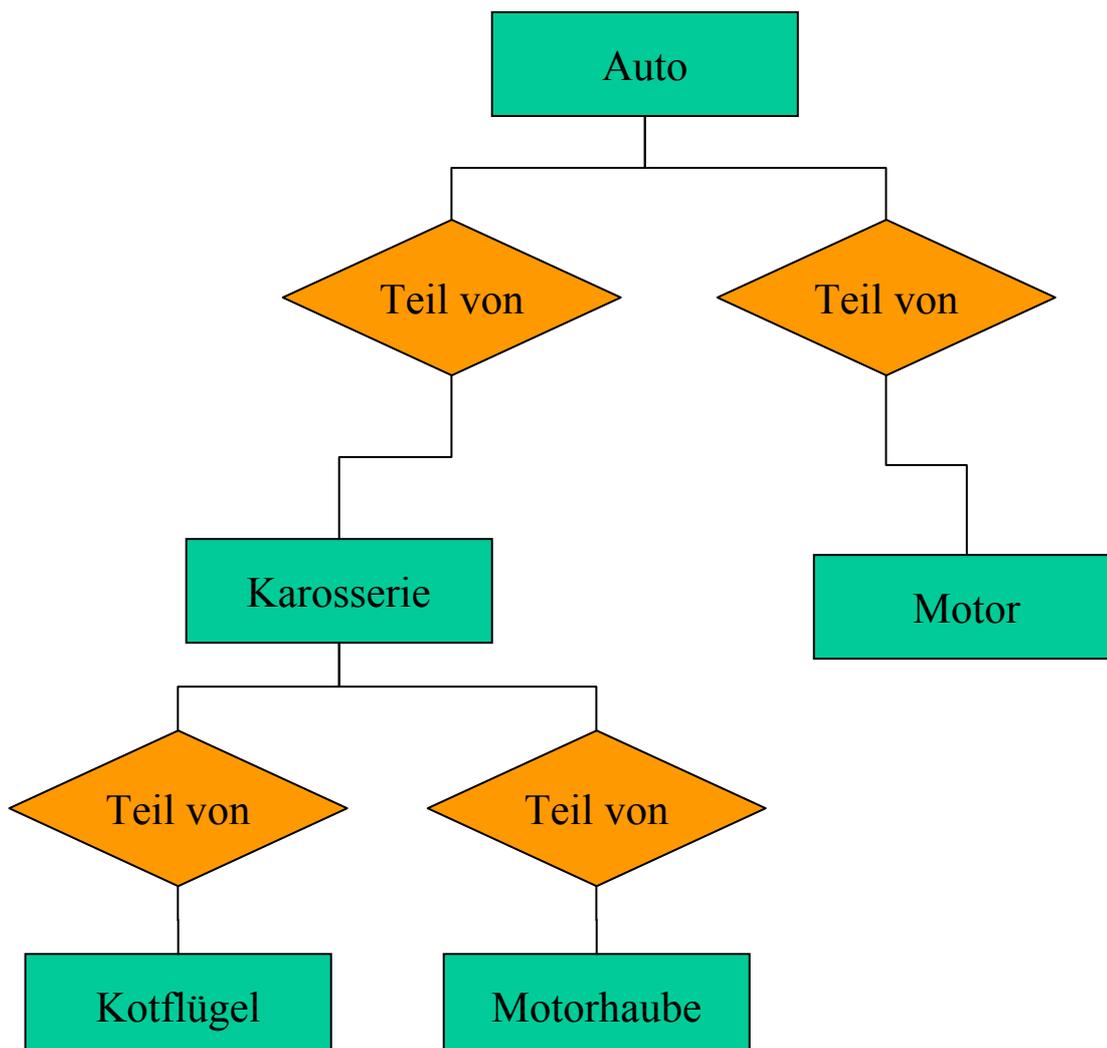
Weitere Verfeinerungen: Aggregation

Aggregation ist ein weiteres Hilfsmittel, um eine bessere (natürlichere und übersichtlichere) Strukturierung der Entitytypen zu erhalten

Bei der Aggregation werden unterschiedliche Entitytypen strukturiert, die in Ihrer Gesamtheit ein strukturiertes Objekt ergeben, einander zugeordnet

Aggregationen erkennt man vor allem daran, daß die Beziehung zwischen zwei Entities als eine „Teil von“-Beziehung charakterisieren läßt

Beispiel: Automobil:



Qualität des Datenbankschemas

Tabelle Student				
Nachname	Vorname	MatrikeINR	Nummer des Studienfachs	Name des Studienfachs
Meiser	Hans	0815	1	Dummschwätzen
Clinton	Bill	007	23	(zensiert)
Muster	Gerd	4711	4	Psychologie
...

- Redundanzen in Tabellen erschweren die korrekte Pflege und Präsentation der Daten
- Die Einhaltung der Normalformen verhindert Redundanzen
=> wichtig für den Entwurf
- Bemerkung 1:
Ein Merkmal einer Spalte ist redundant, wenn man es ohne Informationsverlust weglassen kann
- Bemerkung 2:
In der Praxis sind die ersten drei Formen einzuhalten, um einen brauchbaren Entwurf zu erhalten



Fallbeispiel

Mitarbeiterkonto		
Bankverbindung	MitarbID	
123465 Sparkasse Karlsruhe (66050101)	1	
235236 Citibank (11435334)	2	
985762 Volksbank Hintertupfingen (12376512)	3	
...	...	



Mitarbeiterkonto			
KontoNR	Bank	BLZ	MitarbID
123465	Sparkasse Karlsruhe	66050101	1
235236	Citibank	11435334	2
985762	Volksbank Hintertupfingen	12376512	3
...			...

1. Normalform:

Keine zusammengesetzten Werte in den Feldern, wie in den Feldern „Bankverbindung“ vor der Normalisierung !



Fallbeispiel 2

Mitarbeiterkonto			
KontoNR	Bank	BLZ	MitarbID
123465	Sparkasse Karlsruhe	66050101	1
235236	Citibank	11435334	2
985762	Volksbank Hintertupfingen	12376512	3
...			...



Banken	
Bank	BLZ
Sparkasse Karlsruhe	66050101
Citibank	11435334
Volksbank Hintertupfingen	12376512
...	

&

Mitarbeiterkonto		
KontoNR	BLZ	MitarbID
123465	66050101	1
235236	11435334	2
985762	12376512	3
...		...

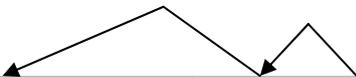
2. Normalform:

Alle Nichtschlüsselwerte sind vollständig vom Schlüssel abhängig.

In diesem Fall war „Bank“ nur von „BLZ“ abhängig, nicht aber von der Kontonummer !



Fallbeispiel 3



Mitarbeiterkonto			
KontoNR	BLZ	MitarbID	Name
123465	66050101	1	Mayer
235236	11435334	2	Müller
985762	12376512	3	Schulze
...		...	



Mitarbeiterkonto		
KontoNR	BLZ	MitarbID
123465	66050101	1
235236	11435334	2
985762	12376512	3
...		...

&

Mitarbeiter	
MitarbID	Name
1	Mayer
2	Müller
3	Schulze
...	

3. Normalform:

keine transitiven Abhängigkeiten, d.h. keine Abhängigkeiten über Umwege. Anders ausgedrückt sollte normale Attribute untereinander nicht abhängig sein, sondern nur direkt vom Schlüsselattribut!



Fallbeispiel 4

Person	Sprachen	Hobbies
Hans	Englisch	Schwimmen
Hans	Englisch	Radfahren
Hans	Russisch	Schwimmen
Hans	Russisch	Radfahren
Peter



Person	Sprachen
Hans	Englisch
Hans	Russisch
Peter	...

&

Person	Hobbies
Hans	Schwimmen
Hans	Radfahren
Peter	...

4. Normalform:

keine mehrwertigen Abhängigkeiten

Überlegungen zur Abfrage nach den Personen, die Englisch **und** Schwimmen können, führt zu dem Zwang, alle Kombinationen in der ersten Tabelle zu führen. Ansonsten könnte es sein, daß Hans nicht ausgegeben wird, wenn man nur zwei Datensätze mit Englisch/Radfahren und Russisch/Schwimmen angelegt hat!



Zusammenfassung der Normalformen

- 1. Normalform:
„Ein Relationenschema ist in Normalform, falls jeder Wertebereich eines Attributs nur aus atomaren Wertebereichen besteht“
 - 2. Normalform:
1. Normalform + „Nichtschlüsselattribute sind von allen Schlüsselmerkmalen voll funktional abhängig“
 - 3. Normalform:
2. Normalform + „kein Nichtschlüsselmerkmal ist Attribut eines anderen Nichtschlüsselmerkmals“
 - 4. Normalform:
3. Normalform + Berücksichtigung von mehrwertigen Abhängigkeiten
-
- Sinn der Normalformen: Redundanzen verhindern, sauberen Entwurf gewährleisten, Daten pflegbar machen
 - Vorgehensweise: Untersuchung von Abhängigkeiten innerhalb von Tabellenstrukturen und gegebenenfalls Änderung nach bestimmten Regeln
 - Bemerkung: Ist das ER-Schema einigermaßen gewissenhaft durchgeführt worden, dann sind die Normalformen 1-3 schon erfüllt !

Eine wichtige Faustregel: Ein Fehler, dessen Behebung in der **Entwurfsphase 1 Euro** beanspruchen würde, kostet in der **Implementierungsphase 10 Euro** und im **laufenden Betrieb 100 Euro ...**



Ausblick: Objektorientierte DMBS

- Objekt ist das zugrundeliegende Konzept:
 - Objekte gehören einer Klasse an
 - Jedes Objekt hat eine eigene Identität
 - Objekte besitzen Merkmale und Merkmalswerte
 - Objekte können sich aus anderen Objekten zusammensetzen, z.B. Buch besteht aus Kapiteln
 - Generalisierung: Es können Ober- und Unterklassen gebildet werden
 - Objekte besitzen Methoden (bei Buch: „entleihen“, „zurückgeben“)
- Vorteile dieses Konzeptes:
 - Modellierung entspricht besser der menschlichen Vorstellung von Dingen als beim relationalen Modell

