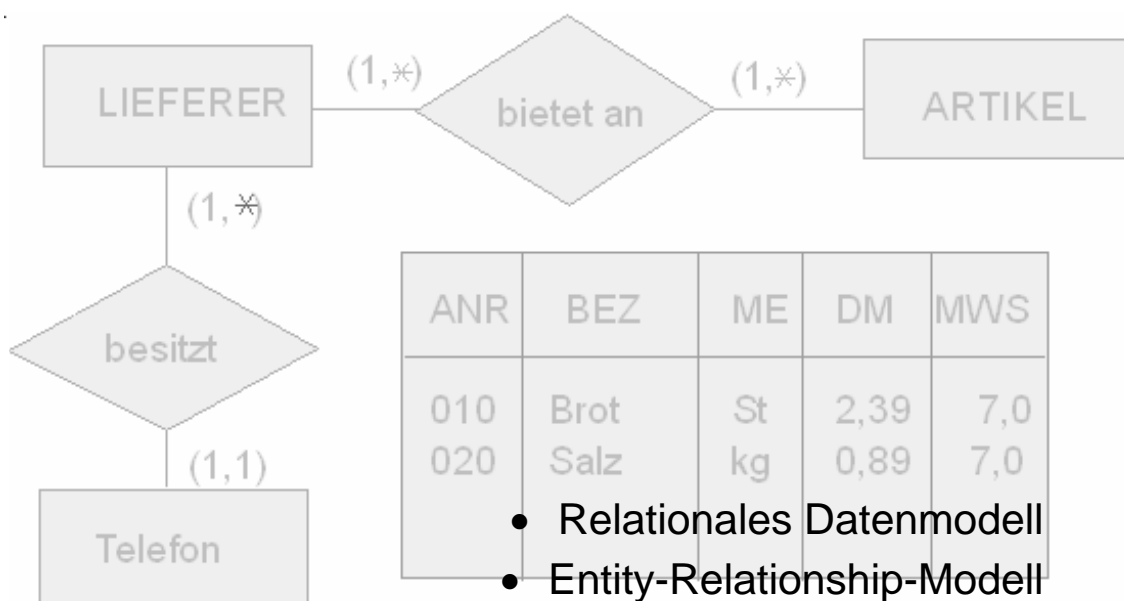


Betriebliche Datenmodellierung und Datenbanktechnologie (DB)



- Relationales Datenmodell
- Entity-Relationship-Modell
- Datenmodellierungstool Data Architect
 - Datenbeziehungen in Formularen
 - Begriffe

Lehr- und Übungsmaterial

Anmerkung: Dieses Skript wird seit dem Wintersemester 2004/2005 nicht mehr aktualisiert. Verwenden Sie ab 2005 bitte die Powerpoint-Foliensätze zur Vorlesung.

Inhaltsübersicht

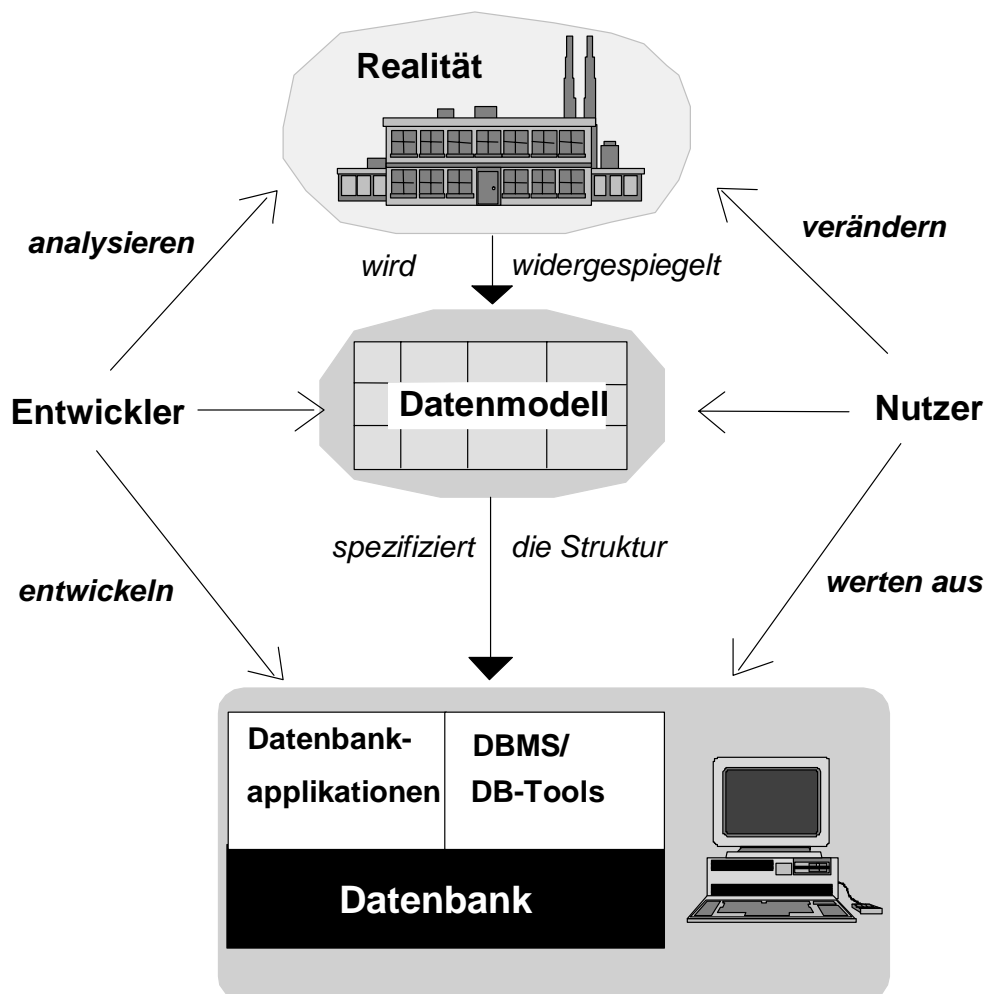
Problemstellung	Seite 1
Teil 1: Das relationale Datenmodell	Seite 4
1.1 Grundform der gebräuchlichsten Datenmodelle	Seite 4
1.2 Die Tabelle (Relation) - Grundlage des relationalen Datenmodells	Seite 5
1.3 Semantisch verbundene Tabellen im relationalen Datenmodell	Seite 6
1.4 Normalformen des relationalen Datenmodells (1.- 4.NF)	Seite 7
1.5 Grundzüge der Relationenalgebra	Seite 8
1.6 Begriffe zum relationalen Datenmodell	Seite 9
Kontrollfragen	Seite 10
Teil 2: Das Entity-Relationship-Modell	Seite 11
2.1 Definition	Seite 11
2.2 Einordnung in die Methodik der Datenmodellierung	Seite 11
2.3 Darstellung des Modells	Seite 11
2.4 Konstruktionsoperatoren zur Erzeugung neuer semantischer Objekte	Seite 14
2.5 Auflösung nichtelementarer Attribute in ERM	Seite 15
2.6 Umsetzung des ERM in Relationen des relationalen Datenmodells	Seite 15
2.7 Beispiel	Seite 17
2.8 Methodik der Entity-Relationship-Modellierung	Seite 20
2.9 Unternehmensdatenmodelle (UDM) / Data Warehouse	Seite 21
Kontrollfragen	Seite 23
Teil 3: Datenmodellierung mit dem PowerDesigner Data Architect	Seite 24
Teil 4: Widerspiegelung formaler Datenbeziehungen in Formularen (Beispiel MS-Access)	Seite 26
Teil 5: Sachwörter Datenmodellierung und Datenbanken	Seite 27
Teil 6: Literaturhinweise	Seite 31

Problemstellung

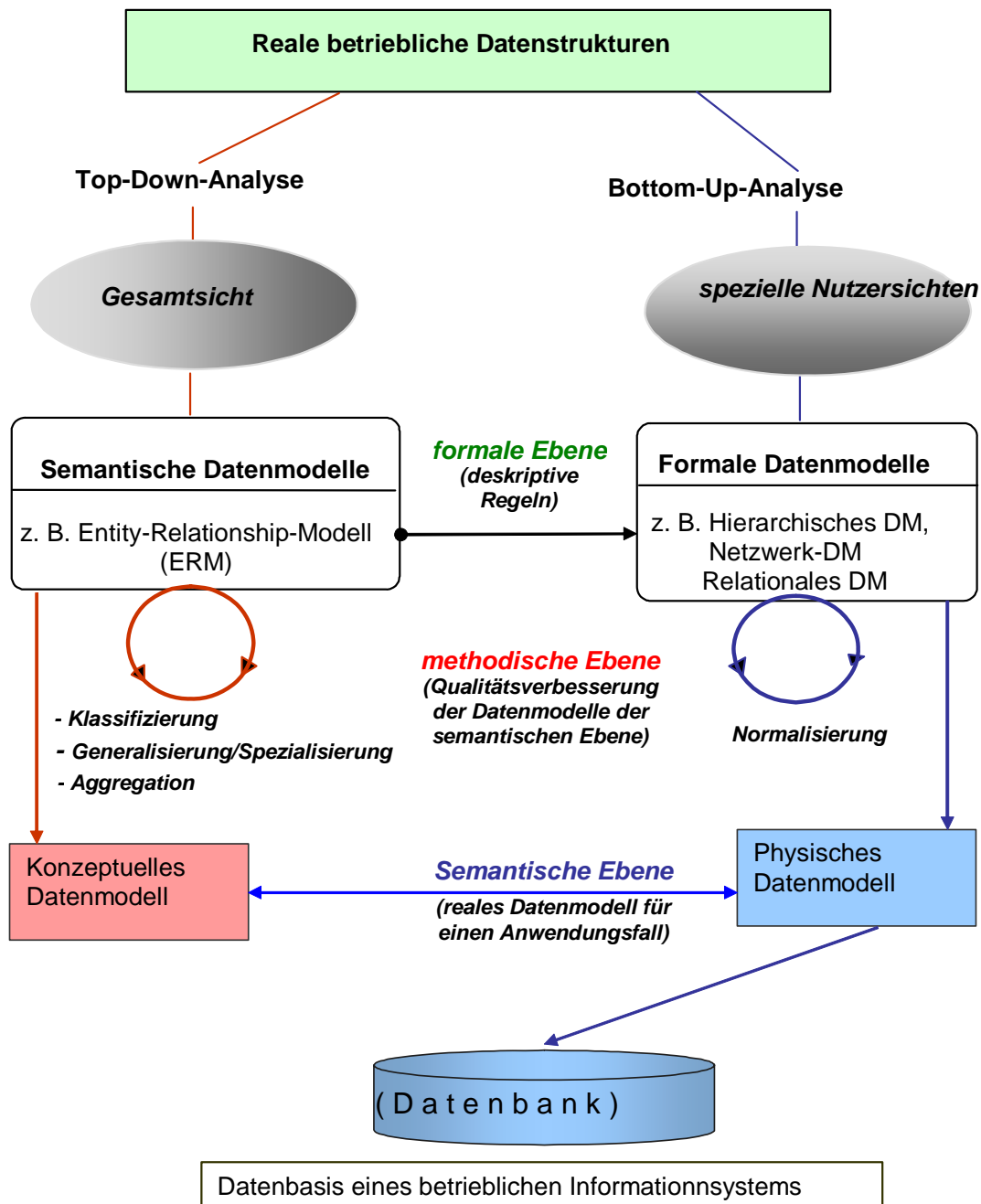
Datenmodelle und Datenbanken dienen aus informationstechnologischer (it) Sicht zur Bewältigung strukturierbarer komplexer Datenmengen. Bezogen auf den Gegenstand der Wirtschaftsinformatik befasst sich die Lehrveranstaltung "Betriebliche Datenmodellierung und Datenbanktechnologie" insbesondere mit der Lösung informationeller Problemstellungen innerhalb betrieblicher Informations- und Kommunikationssysteme. Die Verbindung it- und anwendungsbezogener (semantischer) Aspekte ist unabdingbar, weil jegliches Datenmodell und jegliche Datenbankapplikation nur dann effektiv und betriebswirtschaftlich sinnvoll sein kann, wenn sie in ausreichendem Maße der Realität entsprechen und in der Nutzung der Veränderung der Realität dienen.

Folglich bilden

- die betriebswirtschaftliche Realität als Anwendungsgegenstand,
 - das Datenmodell als formalisierte Widerspiegelung des durch Daten erfassbaren Teils der betriebswirtschaftlichen Realität und
 - die von Anwendersoftware genutzte und durch ein Datenbankmanagementsystem (DBMS) verwaltete Datenbank als Informationsspeicher
- eine untrennbare Einheit.



Innerhalb dieser Einheit fungieren Datenmodelle als Ordnungsschemata zur strukturierten Speicherung von Daten und ihren wechselseitigen Beziehungen. Durch den Prozess der **Datenmodellierung** sollen die in der betriebswirtschaftlichen Realität existenten Informationen mit ihrer semantischen Bedeutung, ihrer syntaktischen Definition und ihren semantischen Beziehungen modellhaft so abgebildet werden, dass sie einer rechnergestützten Speicherung und Bearbeitung zugänglich werden. Dafür gibt es folgende prinzipielle methodischen Ansätze:



Semantische Datenmodelle spiegeln die Realität vornehmlich aus betriebswirtschaftlicher Sicht wider. Konkrete rechnerbezogene Speicherungsformen werden auf dieser Ebene noch nicht dargestellt. Aus der Sicht des Datenmodellie-

rungsprozesses werden diese Modelle deshalb auch als **konzeptuelle Datenmodelle** (seltener: konzeptionelle Datenmodelle) bezeichnet. Die konkrete Struktur einer physikalisch existierenden Datenbank bildet dazu im Gegensatz das **physische Datenmodell** (seltener: physikalisches Datenmodell). Um letztlich eine Datenbanktechnologie praxisreif zu entwickeln, muss folglich das ihr zugrunde liegende semantische Datenmodell in ein adäquates physisches Datenmodell überführt werden. Eine wichtige Mittlerrolle spielen dabei formale Datenmodelle.

Formale Datenmodelle sind theoretische Konstrukte, die eine formale Struktur der Datenobjekte und ihre möglichen Beziehungen **ohne** jeglichen semantischen Inhalt beschreiben. Sie sind faktisch der formale Konstruktionsplan nach dem spezielle anwendungsbezogene physische Datenmodelle entwickelt werden. Erst dadurch wird die Entwicklung allgemeingültiger **Datenbankmanagementsysteme** (DBMS) als softwareseitige Grundlage des Betriebs von **Datenbanken** möglich. DBMS fußen bezüglich der von ihnen unterstützten Speicherstrukturen für Daten immer auf einem formalen Datenmodell. Die für ein betriebliches Informationssystem insgesamt oder auch nur für eine spezielle Softwareapplikation benötigten Daten werden innerhalb einer Datenbank mit einem konkreten physischen Datenmodell beschrieben, das den Konventionen des dem DBMS zugrunde liegenden formalen Datenmodells entspricht. Das physische Datenmodell kann im Bottom-Up-Verfahren intuitiv direkt entwickelt werden. Speziell für komplexe Anwendungen hat sich in der Praxis jedoch die auf dem Top-Down-Verfahren beruhende Entwicklung semantischer Datenmodelle mit ihrem vorwiegend inhaltlichen von der konkreten Struktur der Datenbank abstrahierenden Bezug durchgesetzt. Die nachfolgend notwendige Überführung des semantischen Datenmodells in ein physikalisches Datenmodell kann in Abhängigkeit von dem gewählten formalen Datenmodell nach verbindlichen Regeln - und damit rechnerunterstützt mittels Datenmodellierungstools - erfolgen.

Den Ablauf der Datenmodellierung als Phase des Datenbankentwicklungsprozesses in einem Unternehmen zeigt die nachfolgende Abbildung (frei nach Bleimann u. a. : Betriebsinformatik, München Wien 1990):

Arbeitsetappe

- ❶ Konzeptuelle (konzeptionelle) Datenmodellierung
(Gesamt-Modell der Datenobjekte und ihrer Verbindungen)
- ❷ Modellierung der Verteilungsstruktur 1)
(Verteilung der Daten innerhalb räumlich verteilter Knoten eines Rechnernetzes)
- ❸ Logische Datenmodellierung
(Strukturierung der Daten auf Basis eines konkreten formalen Datenmodells)
- ❹ Physische Datenmodellierung
(Anpassung an konkrete Hardware und Datenbanksoftware)
- ❺ Modellierung externer Sichten (Views)
(Datenauswahl für konkrete Teilanwendungen)

1) nur für verteilte Datenbanken.

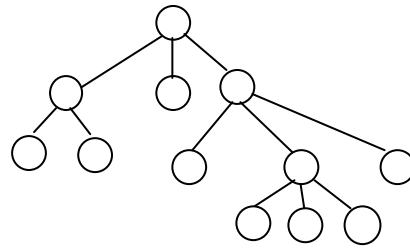
Mitwirkung



Teil 1: Das relationale Datenmodell

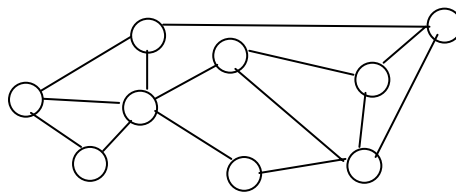
1.1 Grundform der gebräuchlichsten Datenmodelle

1. Hierarchisches Modell



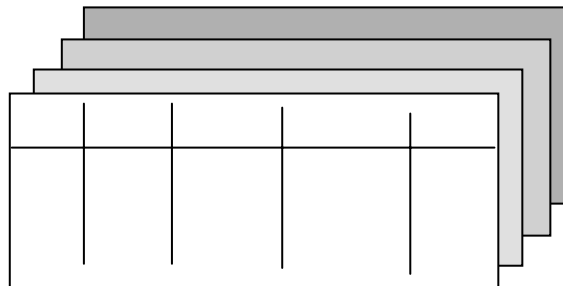
Beispiel: Directory des Betriebssystems MS-DOS

2. Netzwerkmodell (vermaschte Modelle)



Beispiel: Internet WWW (HTML-Seitenstruktur)

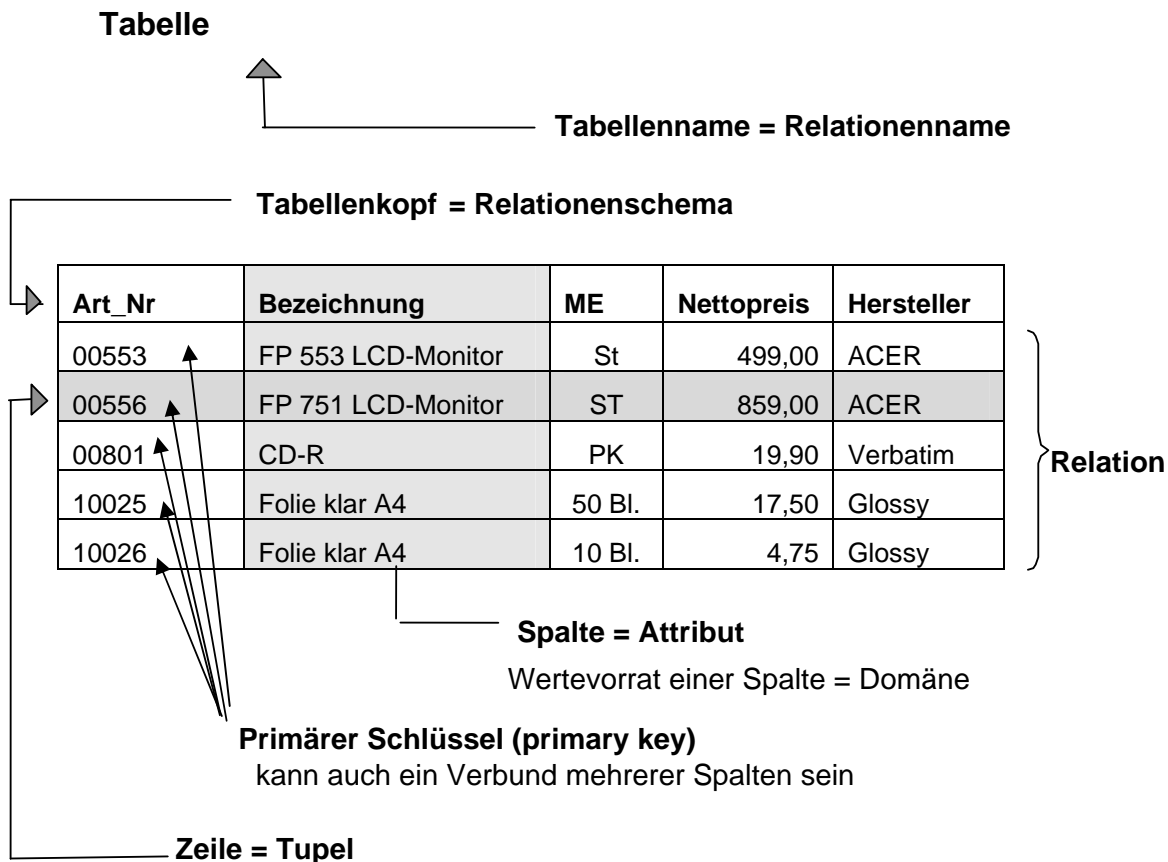
3. Relationales Modell (Tabellen)



Beispiel: Mit MS-ACCESS verwaltete Datenbank

Fazit: Formale Datenmodelle beschreiben Datenobjekte und ihre Beziehungen.

1.2 Die Tabelle (Relation) - Grundlage des relationalen Datenmodells



Schreibweise als Relation: ARTIKEL(Art_Nr, Bezeichnung, ME, Nettopreis, Lager)

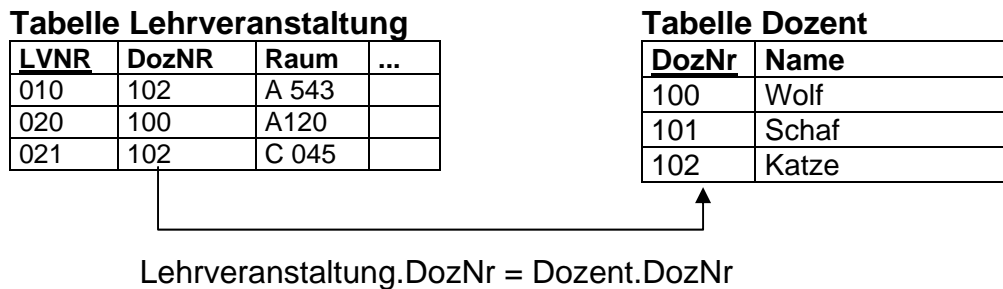
Charakteristische Relationeneigenschaften nach Codd sind:

1. Keine zwei Tupel einer Relation sind identisch, d. h. es existieren keine identischen Zeilen innerhalb einer Tabelle (Ausnahme: Anonymisierung).
2. Die Reihenfolge der Tupel einer Relation ist ohne Belang, d. h. die Sortierfolge der Zeilen einer Tabelle ist ohne Bedeutung.
3. Die Reihenfolge der Attribute einer Relation ist ohne Belang, d. h. die Spalten einer Tabelle werden mit Namen und nicht mit ihrer Position bezeichnet.
4. Jeder Attributswert in der Relation ist elementar. Wertemengen innerhalb einer Spalte der Tabelle sind folglich nicht zulässig (siehe 1.NF).

Als Primärschlüssel (primary key) dient die minimal notwendige Anzahl von Attributen, die erforderlich ist, um jedes Tupel eindeutig zu identifizieren. Können mehrere Attribute diese Funktion alternativ erfüllen, spricht man von Schlüsselkandidaten.

1.3 Semantisch verbundene Tabellen im relationalen Datenmodell

In der Regel besteht eine Datenbankapplikation aus mehreren Relationen, zwischen denen sinnvollerweise semantische **Beziehungen** (Relationship) existieren (vgl. Teil 2). Bedingung für die Definition dieser Beziehungen im Datenmodell ist das Vorhandensein von Schlüsselattributen zur Identifikation der Tupel in den beteiligten Relationen, deren Domänen mengenmäßig identisch sind.



Deutung:

Die Tabelle "Lehrveranstaltung" ist bezüglich der eingesetzten Dozenten abhängig von der Tabelle "Dozent". Das bedeutet semantisch, dass in einer Lehrveranstaltung nur ein Dozent (identifiziert durch das Attribut DozNr) eingesetzt werden kann, der in der Tabelle Dozent enthalten ist. Im formalen Datenmodell wird diese Beziehung als **referentielle Integrität** bezeichnet. Die referentielle Integrität wird über Schlüsselattribute realisiert. Diesen Schlüsselattributen muss die gleiche Domäne (der gleiche Wertebereich) zugeordnet sein. Der Schlüssel in der abhängigen Tabelle wird als **Fremdschlüssel** (foreign key) bezeichnet. In der unabhängigen Tabelle ist in der Regel der referenzierte Schlüssel mit dem primary key identisch.

Im Beispiel besteht die Domäne des Attributes DozNr (Dozentenummer) in der Tabelle "Dozent" aus der Gesamtheit der Dozentenummern aller z. B. an der FHTW tätigen Dozenten. Dieser Schlüssel ist in der Tabelle "Dozent" zugleich primary key. In der Tabelle "Lehrveranstaltung" ist das Attribut DozNr foreign key. Der foreign key kann nur Werte enthalten, die auch im Attribut DozNr der Tabelle "Dozent" enthalten sind.

1.4 Normalformen des relationalen Datenmodells (1. - 4. NF)

Unter Normalisierung versteht man schlechthin den Prozess der verlustfreien Zerlegung von Relationen mit dem Ziel, dass die durch Zerlegung gewonnenen Relationen definierten Anforderungen bezüglich Redundanzfreiheit und Vermeidung von Anomalien genügen. Die Anforderungen werden durch die zu erreichende Normalform (NF) bestimmt. Von besonderer praktischer Bedeutung für den Entwurf von Datenbanken sind die 1. bis 3. Normalform.

Ausgangspunkt der Normalisierung ist die Bestimmung der sogenannten funktionalen Abhängigkeit (engl. functional dependency, abgekürzt häufig FD). Die Werte der Attribute β innerhalb einer Relation sind **funktional abhängig** von den Attributen α , wenn alle Werte β eindeutig durch die Werte der Attribute α bestimmt werden ($FD \alpha \rightarrow \beta$). Die Werte der Attribute β heißen **voll funktional abhängig**, wenn alle Werte β funktional abhängig von α , nicht aber funktional abhängig von einem beliebigen Teilattribut aus α sind ($FD \alpha \twoheadrightarrow \beta$). Von großer praktischer Bedeutung ist fernerhin die sogenannte transitive Abhängigkeit. Hierunter versteht man eine Beziehung von funktionalen Abhängigkeiten, für welche gilt: Ist $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$, dann ist auch $\alpha \rightarrow \gamma$.

1. Normalform (1NF): Attribute sind elementar (auch atomar), d. h. es gibt **keine** anderen Attribute $B_1 \dots B_n$, aus denen ein Attribut A_0 ($W(A_0) = W(B_1) * W(B_2) * \dots * W(B_n)$) abgeleitet werden kann. Alle Tupel (Zeilen) einer Relation (Tabelle) weisen die gleiche Anzahl von Attributen (Spalten) auf.

Beispiel: Relation STUDENT(ImmNr, Name, Vorname, Fach1, Note, Fach2, Note,...)

Negativbeispiel: Relation STUDENT(ImmNr, Name_und_Vorname, Fächer_mit_Noten,...)
→ Fächer und Noten sind eine mengenwertige Liste, die in Einzelattribute Fach_i und Note_i zerlegbar und folglich nicht elementar ist.

2. Normalform (2NF): Volle funktionale Abhängigkeit der Nichtschlüsselattribute von **allen** aus mehreren Attributen zusammengesetzten Schlüsseln. Folglich darf kein Attribut des Schlüssels entfernt werden, ohne dass dabei die funktionale Abhängigkeit jedes Nichtschlüsselattributs vom Schlüssel verloren geht.

Beispiel: Relation NOTEN(ImmNr, Fach, Note, Bezeichnung)

Negativbeispiel: Relation BESTELLUNG(LiefNr, ArtNr, Name, Anschrift, Menge, Preis)
→ Abhängigkeit von LiefNr, ArtNr gilt nicht für alle Attribute; Name und Anschrift hängen nur von LiefNr ab.

3. Normalform (3NF): Ausschaltung transitiver funktionaler Abhängigkeiten zwischen einem Nichtschlüsselattribut und einem Schlüsselattribut, d. h. ein Nichtschlüsselattribut darf funktional nur vom Schlüssel und nicht von einem weiteren Nichtschlüsselattribut abhängig sein.

Beispiel: Relation NOTEN(ImmNr, Fach, Note)

Negativbeispiel: Relation ARTIKEL(ArtikelNr, Bezeichnung, VK_Preis, Bestand, LagerNr, LagerOrt) → Transitive funktionale Abhängigkeit zwischen ArtikelNr ↔ LagerNr ↔ LagerOrt

4. Normalform (4NF): Beseitigung voneinander unabhängiger mehrwertiger Abhängigkeiten im Schlüssel einer Relation. Ziel ist die Minimierung der Schlüsselattribute.

Negativbeispiel: Relation ARTIKEL_LIEFERER_LAGER(ArtNr, LiefNr, LagerNr) → Es existieren mehrwertige Abhängigkeiten zwischen Schlüsselattributen, nämlich $\text{ArtNr} \leftrightarrow \text{LiefNr}$ und $\text{ArtNr} \leftrightarrow \text{LagerNr}$. Das gilt allerdings nur unter der Voraussetzung, dass ein Artikel unabhängig vom Lieferer immer nur in einem Lager aufbewahrt wird.

Merke: Die Erzielung der 3NF und 4NF ist abhängig von der Semantik der Daten. Teilt sich beispielsweise ein Lager auf mehrere Orte auf (Negativbeispiel zur 3NF), befindet sich die Relation in der 3NF.

1.5 Grundzüge der Relationenalgebra

Die Relationenalgebra beschreibt (neben dem deklarativen Relationenkalkül) in prozedural orientierter Form, wie die in (auf einer Datenbank abgebildeten) Relationen enthaltenen Daten abgefragt werden können. Prinzipiell können alle Operationen der formalen Logik auf Relationen angewandt werden. Von praktischer Bedeutung für die Datenauswertung mittels eines Datenbankmanagementsystems (DBMS) sind jedoch insbesondere die drei folgenden Operationen:

- **Selektion:** Durch Selektion werden die Daten ausgewählt, die einer *Selektionsbedingung* genügen. Die Selektion bezieht sich also auf den *Dateninhalt*. Die Selektionsbedingung besteht aus Operanden (Attributnamen der selektierten Relation, Konstante), Vergleichsoperatoren (=, <, >, <=, >=, <>) und logischen Operatoren (\wedge , \vee , \neg).
- **Projektion:** Durch die Projektion werden die Daten aus den angegebenen Spalten ausgewählt. Die Projektion bezieht sich also auf die *Struktur* der Relation.
- **Joining:** Durch das Joining (relationaler Verbund) werden Relationen verknüpft. Je nach der Art des Joins werden die aus dem kartesischen Produkt gebildeten Daten gefiltert. Nach der Art der Filterung unterscheidet man in der Praxis vor allem den Equi-Join, natürlichen Join (Equi-Join mit Elimination gleicher Werte) und Outer-Join (auch Semi-Join).

Das Ergebnis einer Datenanfrage entsprechend der relationalen Algebra ist immer eine (neue) Relation.

Die relationale Algebra ist eine Grundlage der standardisierten Datenbanksprache SQL. SQL ist vom Sprachkonzept und von der Leistungsfähigkeit her jedoch wesentlich komplexer.

1.6 Begriffe zum relationalen Datenmodell

Attributsmenge:	$A = \{A_i \mid i=1..n\}$
Boyce Codd Normalform (BCNF):	"Eine Relation befindet sich in der BCNF, wenn jede Determinante der Relation ein Kandidatenschlüssel ist. (Quelle: Jackson, Entwurf relationaler Datenbanken, Hanser Verlag München Wien 1990)
Determinante:	"Gilt $A \rightarrow B$ und B ist nicht funktional abhängig von irgendeinem Teil von A, ist A die Determinante von B." (Quelle: a. a. O.)
Fremdschlüssel (foreign key)	Schlüsselattribute (1-n) in einer abhängigen Tabelle (Relation), die sich auf gleichartige (gemeinsame Domänen) Schlüsselattribute in einer semantisch verbundenen unabhängigen Tabelle beziehen.
funktionale Abhängigkeit von Attributen:	"Bei zwei gegebenen Attributen A und B ist B funktional abhängig von A, wenn für jeden Wert von A jederzeit genau ein Wert von B existiert, der mit dem Wert von A verbunden ist. A und B können zusammengesetzt sein, d. h. sie können auch Gruppen von mehreren Attributen sein." (Quelle: a. a. O.)
Kandidatenschlüssel:	"Ein Kandidatenschlüssel ist ein Attribut, oder eine Reihe von Attributen, die als Primärschlüssel für die gegebene Relation verwendet werden könnten. Der Primärschlüssel ist immer ein Kandidatenschlüssel, aber es kann andere Kandidatenschlüssel geben, die als Primärschlüssel hätten verwendet werden können, es aber nicht wurden." (Quelle: a. a. O.)
Kartesisches Produkt:	Neue Menge = Menge ₁ * Menge ₂ * ... Menge _n
Normalisierung:	Teilung einer Relation in n Relationen (n >= 2) zur Verringerung der Redundanz bzw. Vermeidung von Anomalien.
Potenzmenge:	$PM = P(M)$ Menge aller Teilmengen der Menge M.
Primärer Schlüssel (primary key)	eines Relationstyps RT(A) sind die Attribute (a ₁ , a ₂ ... a _n), a _i ∈ W(A _i), die zur eindeutigen Identifizierung eines Tupels ausreichen.
Referentielle Integrität:	Abhängigkeit einer Relation R ₁ von einer Relation R ₀ dergestalt, dass ein als Fremdschlüssel definiertes Attribut A _{R1} nur Werte enthalten kann, die auch tatsächlich in der Wertemenge des referenzierten Attributs A _{Rn} enthalten sind.
Relationstyp:	$RT = \{RT\text{-Name}, A_1, A_2, \dots, A_n\}$
Tupel:	t heißt ein Element aus W(A). t; entspricht einer Zeile einer Tabelle (Relation).
Wertebereich (Domäne):	W(A _i) eines Attributs A _i umfasst alle Elemente a _{ij} (Attributswerte), die die Menge A _i beinhalten kann. $a_{ij} \in W(A_i)$
Wertebereich der Attributsmenge:	A ist definiert durch das kartesische Produkt über die Wertebereiche der Attributsmengen A _i . $W(A) = W(A_1) * W(A_2) * \dots * W(A_n)$
Wertebereich W(RT) eines Relationstyps:	RT ist die Potenzmenge über W(A). $W(RT) = P(W(A))$

Kontrollfragen:

1. Erklären Sie die wichtigsten Eigenschaften des hierarchischen Datenmodells am Beispiel eines Windows-NT-Directories?
2. Lässt sich das hierarchische Datenmodell ggf. mit Einschränkungen der Effektivität auch auf dem relationalen Datenmodell abbilden?
3. Inwiefern bildet das hierarchische Datenmodell nur einen Sonderfall des Netzwerkmodells?
4. Nach welchem Prinzip arbeiten sogenannte Suchmaschinen im WWW?
5. Gegeben sei folgende Tabelle (Relation):
PERSONAL(Personalnummer, Name, Vorname, Geburtsdatum, Akademischer_Grad, Fach, Abschlussjahr).
Es gilt als vorausgesetzt, dass ein(e) Mitarbeiter(in) mehrere akademische Abschlüsse besitzen kann.
 - 5.1 Schlagen Sie einen geeigneten primary key für die Tabelle vor.
 - 5.2 In welcher Normalform befindet sich die Tabelle?
 - 5.3 Überführen Sie die Tabelle in die 3NF.
 - 5.4 Würde sich an der Beantwortung der Fragen 5.1 bis 5.3 etwas ändern, wenn prinzipiell nur ein akademischer Abschluss möglich wäre?
6. Gegeben seien die folgenden Tabellen:

TabA

Nr	Wert
1	A
5	E
7	G

TabB

Nr	Text
1	Anna
3	Christa

Bilden Sie das kartesische Produkt über beide Relationen.

7. Gegeben sind die semantisch abhängigen Tabellen
 - PERSONAL(Personalnummer, Name, Vorname, Geburtsdatum, ...)
 - und
 - KINDER(Name, Vorname, Geburtsdatum).Die semantische Abhängigkeit besteht darin, dass in der Tabelle KINDER nur die Daten der Kinder des Personals erfasst werden sollen.
 - 7.1 Definieren Sie die referentielle Integrität zwischen diesen Tabellen.
 - 7.2 Ergänzen Sie die Tabellen um den primary key und um den foreign key.

Teil 2: Das Entity-Relationship-Modell

2.1 Definition

Von P.P.-S. Chen (The Entity-Relationship Model - Toward a Unified View of Data, in ACM Transactions on Database Systems, Vol.1, No.1, March 1976, S. 9-36) vorgeschlagenes Modell zur Erfassung semantischer Abhängigkeiten zwischen Daten. Das Entity-Relationship-Modell (ERM) setzt voraus, dass die zu modellierende informationelle Realität aus einer Vielzahl abgrenzbarer Objekte (Entities) besteht, die miteinander in Beziehung stehen (Relationship). Entities und Relationships sind durch Attribute beschreibbar.

2.2 Einordnung in die Methodik der Datenmodellierung

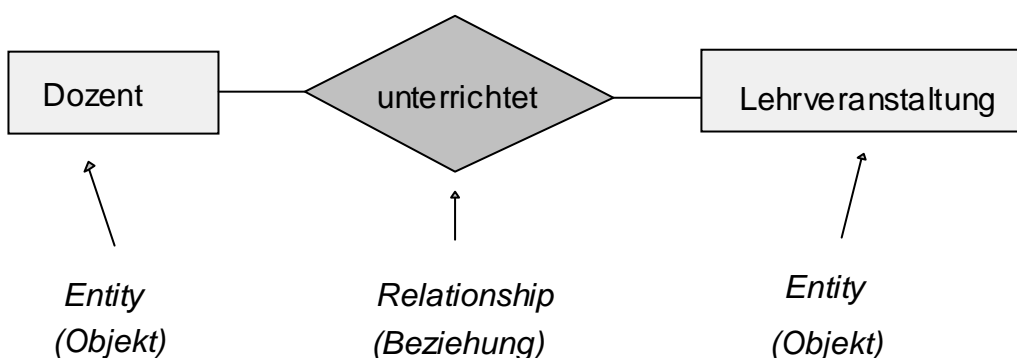
Das ERM - als Verfahren auch mit Entity-Relationship-Methode bezeichnet - wird methodisch

- zum Entwurf von (in der Regel relationalen) Datenbanken,
- zur Entwicklung von Datenstrukturen als Bestandteil des Softwareentwicklungsprozesses (häufig im Rahmen von CASE-Software) sowie
- zum Aufbau und zur Beschreibung von Unternehmensdatenmodellen und/oder Data Warehouse-Datenmodellen.

genutzt. Fernerhin bildet es die methodische Grundlage zur Nutzung unterschiedlicher Datenmodellierungssoftware. Im Datenbanklabor des FB 4 der FHTW Berlin können Sie zu diesem Zweck die Softwareprodukte Power Designer Data Architect von Sybase bzw. S-Designor Professional (Vorversion) nutzen (siehe Abschnitt 3).

2.3 Darstellung des Modells

2.3.1 Grundelemente



Jedes Entity wird durch 1 bis n Attribute beschrieben. **Jedes Attribut ist elementar.**

Folgende Definitionen gelten:

Element	Definition	Beispiel(e)
Entity e	abgrenzbares Objekt	Dozent, Lehrveranstaltung
Relationship rs	Beziehung zwischen den Objekten	unterrichtet
Attribut A_i	elementarer Wert zur Beschreibung eines Entitys oder Relationships	Name, Fachgebiet, Anschrift eines Dozenten
Schlüssel	ein oder mehrere verbundene Attribute, die zur eindeutigen Identifikation eines Entitys e_i oder eines Relationship r_s ausreichen	Lehrveranstaltungsnummer; Personalnummer des Lehrers
Domäne	Wertebereich für ein Attribut A_i	Fachgebietsschlüssel
Entity Set E	Menge gleichartiger Entitys e	alle Lehrer einer Hochschule
Entity-Typ ET	durch die Potenzmenge aller gleichartigen Entitys e definierte Beschreibung der Entitystruktur	Strukturbeschreibung "Dozent"
Relationship-Set RS	Menge gleichartiger Relationships rs	alle Festlegungen einer Hochschule über den Dozenteneinsatz
Relationship-Typ RST	durch die Potenzmenge aller gleichartigen Relationships rs definierte Beschreibung der Relationshipstruktur	Strukturbeschreibung "unterrichtet"

2.3.2 Grad von Beziehungen

Der **Grad (auch Komplexität)** der Beziehung gibt zu jedem Relationship-Typ an, mit wie vielen Entitys des Typs ET_i ein Entity des Typs ET_k in einem konkreten Relationship Set in Beziehung stehen kann. Es gibt folgende Beziehungstypen:

- 1:1 bijektive funktionale Beziehung:

Jedes Entity e_i steht in Beziehung zu genau einem anderen Entity e_j (z. B. jeder Lehrer unterrichtet genau einen Kurs).

- 1:n funktionale Beziehung (auch in Umkehrung m:1):

Jedes Entity e_i kann zu mehreren anderen Entitys e_j in Beziehung stehen, aber jedes andere Entity e_j nur zu einem Entity e_i (z. B. ein Lehrer unterrichtet mehrere Kurse, aber jeder Kurs wird nur von einem Lehrer unterrichtet).

- m:n nichtfunktionale Beziehung

Mehrere Entitys e_i können in Beziehung zu mehreren Entitys e_j stehen (z. B. mehrere Lehrer können einen Kurs unterrichten und jeder Lehrer kann mehrere Kurse unterrichten).

2.3.3 Ausprägung von Beziehungen

Die **Ausprägung (auch Optionalität)** einer Beziehung beschreibt die Verbindlichkeit der Teilnahme jedes Entitys e_i an der Beziehung eines Relationship-Typs RST. Die Ausprägung von Beziehungen kann in ER-Ausprägungs-Diagrammen beschrieben werden (siehe Punkt 2.7 - Beispiel). Die Ausprägung einer Beziehung

ist für beide an der Beziehung beteiligten Entities e_j , e_k gegeben. Folgende graduell unterschiedliche Ausprägungen sind möglich:

nicht obligatorisch: ("kann"-Beziehung) Data Architect: <i>leer</i>	Es kann Entities eines Entity-Sets geben, die durch die Beziehung nicht angesprochen werden (z. B. innerhalb eines Semesters braucht nicht jeder Dozent zu unterrichten).
obligatorisch: ("muss"-Beziehung) Data Architect: <i>Mandatory</i>	Jedes Entity eines Entity-Sets (jeder Dozent bzw. jede Lehrveranstaltung) muss durch die Beziehung angesprochen werden.
obligatorisch-abhängig: Data Architect: <i>Is Dependent</i>	Die Entities eines Entity-Sets können nur in Abhängigkeit von einem anderen (unabhängigen) Entity-Set existieren (z. B. können die Bankverbindungen von Kunden nur existieren, wenn Kunden existieren).
obligatorisch-existenziell abhängig:	Die Entities beider Entity-Sets können nur in Abhängigkeit voneinander existieren (z.B. kann ein Entity Bestellkopf nur existieren, wenn mindestens ein zugehöriges Entity bestellter_Artikel existiert; gleichzeitig kann es kein Entity bestellter_Artikel geben, wenn es keinen zugehörigen Bestellkopf gibt.

Der Grad und die Ausprägung einer Beziehung zwischen Entities werden zusammenfassend als **Kardinalität** bezeichnet.

2.3.4 Darstellung der Kardinalität von Beziehungen im ERM durch Veränderung der Notationsform

Die Kardinalität einer Beziehung kann wie folgt sowohl den Grad als auch die Verbindlichkeit der Teilnahme eines Entitys an der Beziehung zum Entity-Set der anderen Seite ausdrücken:

Grad	obligatorisch	nicht obligatorisch
1 : 1	[1,1]	[0,1]
1 : n	[1,*]	[0,*]

(auch als Minimum-Maximum-Notation bezeichnet)

Beispiel:



Diese Notationsform ermöglicht die Betrachtung einer Beziehung jeweils nur aus der Sicht eines Entity-Sets. Die Datenmodellierungssoftware Data Architect benutzt dabei für die Bezeichnung des Grades einer Beziehung die Begriffe One (für 1) und Many (für n).

Neben den oben genannten im engeren Sinne konzeptuellen Objekten werden mit Vorausblick auf die Generierung physischer Datenmodelle in Modellierungstools auch Angaben zur (standardisierten) physischen Datenmodellierung bereits bei der Entwicklung des semantischen Modells abgefragt. Neben (sinnvollerweise kurzen) physischen Namen für die Relationen und Attribute sind das in der Regel

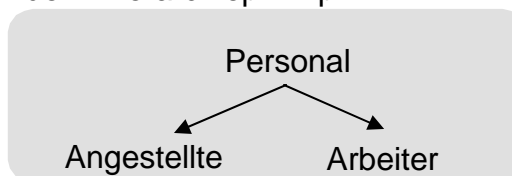
- der Datentyp des Attributs,
- ein Kandidaten-Vorschlag für den primary key,
- die Angabe, ob für das Attribut in der Datenbank ein Wert angegeben werden muss (oder ob er leer bleiben kann),
- die (statische) Domäne (soweit eine Prüfung erfolgen soll) und
- eine Datenprüfbedingung (soweit eine Prüfung erfolgen soll).

2.4 Konstruktionsoperatoren zur Erzeugung neuer semantischer Objekte

Konstruktionsoperatoren dienen zur Systematisierung semantischer Informationen auf dem Wege der Erzeugung neuer Begriffe aus vorhandenen Basisdefinitionen. Folgende Konstruktionsoperatoren werden am häufigsten angewandt:

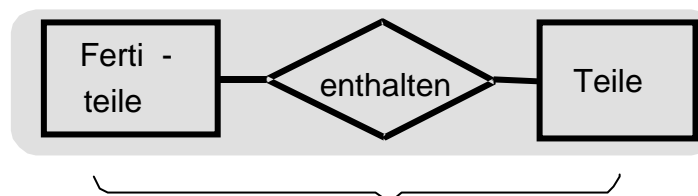
1. Klassifizierung Entities mit gleichen Attributen werden eine Klasse (Bildung von Objekttypen)

2. Generalisierung/ Spezialisierung Zusammenfassen/Unterteilen von Objekttypen nach dem Hierarchieprinzip



Die Eigenschaften von Personal werden auf Angestellte und Arbeiter vererbt.

3. Zusammenfassung von Beziehungen zu einem höher-klassigen Objekt



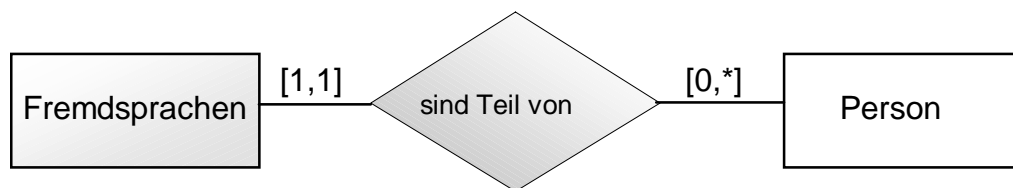
Definition als neues Objekt (Entity)

Fertigteilestückliste

2.5 Auflösung nichtelementarer Attribute in ERM

Bei der Verfeinerung von ERM tritt häufig das Problem auf, dass Attribute nicht elementar sind. Insbesondere sind bei betriebswirtschaftlich orientierten semantischen Datenmodellen Attributsgruppen (Attribute zur Beschreibung eines Entity, die mehrfach auftreten) typisch. Dies verstößt gegen die syntaktischen Grundregeln der Datenmodellierung mittels ERM (siehe Abschnitt 2.3.1). Problematisch wird das allerdings nur dann, wenn das ERM auf die physische Ebene des relationalen Datenmodells transformiert wird, da innerhalb einer Relation auch nur elementare Attribute zulässig sind (anders ist das z. B. bei objektorientierten und objektrelationalen Datenmodellen). Die Lösung des Problems besteht in der Aufteilung des Entitys mit Attributsgruppen in mehrere (aus betriebswirtschaftlicher Sicht gegebenenfalls "künstliche" Entities), die untereinander mit einem "sind-Teil-von"-Relationship verbunden werden.

Beispiel: Ein Entity Personal wird unter anderem durch das Attribut Fremdsprachenkenntnisse beschrieben. Da eine Person mehrere Fremdsprachen beherrschen kann, kann folglich auch das Attribut Fremdsprachenkenntnisse innerhalb eines jeden beliebigen Entitys des Entityset Personal mehrfach (0 bis n-mal) auftreten. Also muss die Attributsgruppe Fremdsprachenkenntnisse als eigenständiges Entity aus dem Entity Personal herausgelöst werden, wodurch folgende Beziehung entsteht:



2.6 Umsetzung des ERM in Relationen des relationalen Datenmodells

Durch die Erfassung der betriebswirtschaftlichen Realität mittels der ERM entsteht ein semantisch orientiertes Datenmodell. Die Umsetzung in datenbankspezifische formale Datenmodelle erfolgt nahezu ausschließlich über Konstruktionsregeln zum Entwurf der Relationen gemäß dem relationalen Datenmodell. Begriffliche Beziehungen zwischen dem ERM und dem relationalen Datenmodell ergeben sich aus den Transformationen:

Entity-Relationship-Modell		Relationales DM
Entity	Relationship	Tupel (Zeile der Tabelle)
Entity-Set	Relationship-Set	Relation (Tabelle)
Entity-Typ	Relationship-Typ	Relationenschema
Attribut	Attribut	Attribut (Spalte der Tabelle)
Entityschlüssel	Relationshipschlüssel	Primärschlüssel (primary key)

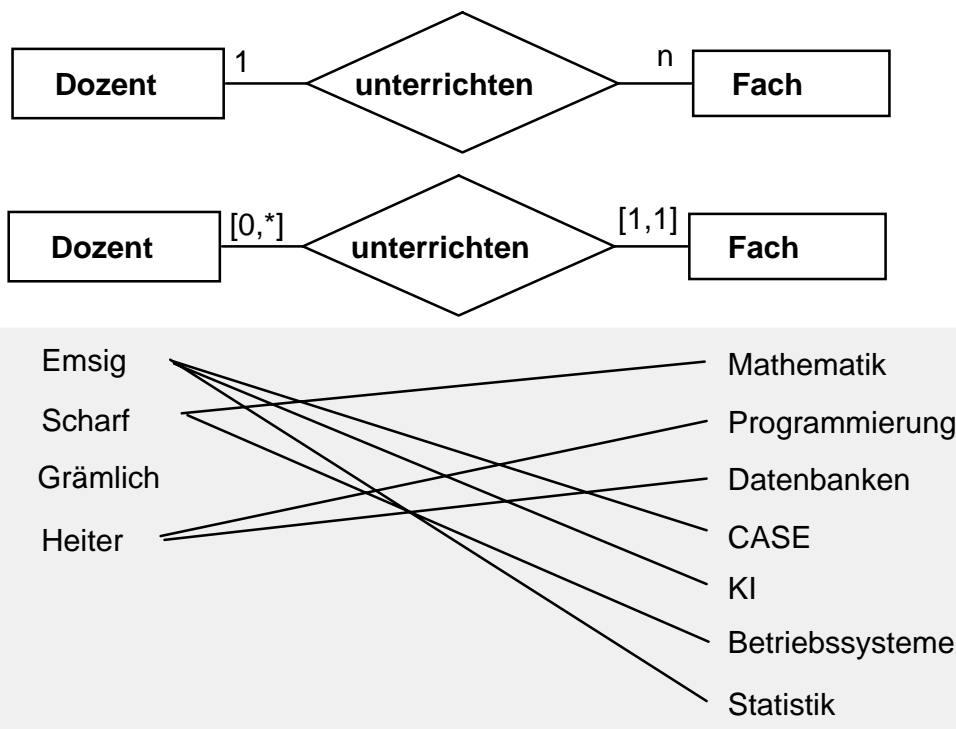
Die Regeln der Ableitung von relationalen Datenmodellen aus dem ERM sind abhängig sowohl vom Grad als auch von der Ausprägung der Beziehungen. Es existieren folgende Regeln (vgl. Jackson: Entwurf relationaler Datenbanken, Carl Hanser Verlag München Wien, 1990):

- REGEL 1:** Ist der Grad der Beziehungen 1:1 und die Teilnahme der Ausprägungen beider Entities obligatorisch, wird nur eine Relation benötigt. Als Primärschlüssel der Relation eignet sich der Entityschlüssel jeder der beiden Relationen.
- REGEL 2:** Ist der Grad der Beziehungen 1:1 und die Teilnahme der Ausprägungen nur einer der beiden Entities obligatorisch, werden zwei Relationen benötigt. Jedes Entity wird durch eine Relation dargestellt. Der Entityschlüssel wird zum Primärschlüssel der jeweiligen Relation. Der Entityschlüssel der nichtobligatorischen Seite ist zugleich ein Attribut (Fremdschlüssel) der Relation der obligatorischen Seite.
- REGEL 3:** Ist der Grad der Beziehungen 1:1 und die Teilnahme der Ausprägungen von keiner der beiden Entities obligatorisch, werden drei Relationen benötigt. Es gibt jeweils eine Relation für beide Entities und die Beziehung. In der Beziehungsrelation müssen beide Entityschlüssel als Attribut (Fremdschlüssel) enthalten sein.
- REGEL 4:** Ist der Grad der Beziehungen 1:n und die Teilnahme der Ausprägungen der n-Seite obligatorisch, sind zwei Relationen erforderlich. Jedes Entity wird durch eine Relation dargestellt. Der Entityschlüssel wird zum Primärschlüssel der jeweiligen Relation. Der Entityschlüssel der 1-Seite ist zugleich ein Attribut der Relation der n-Seite (Fremdschlüssel).
- REGEL 5:** Ist der Grad der Beziehungen 1:n und die Teilnahme der Ausprägungen der n-Seite nichtobligatorisch, sind drei Relationen erforderlich. Es gibt jeweils eine Relation für beide Entities und für die Beziehung. In der Beziehungsrelation müssen beide Entityschlüssel als Attribut (Fremdschlüssel) enthalten sein.
- REGEL 6:** Ist der Grad der Beziehungen m:n, sind drei Relationen erforderlich. Es gibt jeweils eine Relation für beide Entities und die Beziehung. In der Beziehungsrelation müssen beide Entityschlüssel als Attribut (Fremdschlüssel) enthalten sein.

2.7 Beispiel

2.7.1 Elementares Beispiel "Unterricht"

In einem Weiterbildungsinstitut unterrichten die Dozenten bestimmte Fächer. Im Regelfall unterrichtet ein Dozent mehrere Fächer. Jedes Fach muss unbedingt genau einen Dozenten haben. Diese Konstellation kann formal mit einer 1:n (Dozent:Fach) Beziehung bei obligatorischer Teilnahme der Ausprägungen der n-Seite beschrieben werden. Eine bildhafte Beschreibung zur weiteren Verfeinerung des Entity-Relationship-Modells ist durch das nachfolgende **ER-Ausprägungsdiagramm** möglich:



Nach Regel 4 ergeben sich daraus folgende (Beispiel-)Relationen:

DOZENT

DozNr	Name	Titel
001	Emsig	Prof. Dr.
002	Scharf	Dr.
003	Grämlich	Dr.
004	Heiter	Dipl. WI

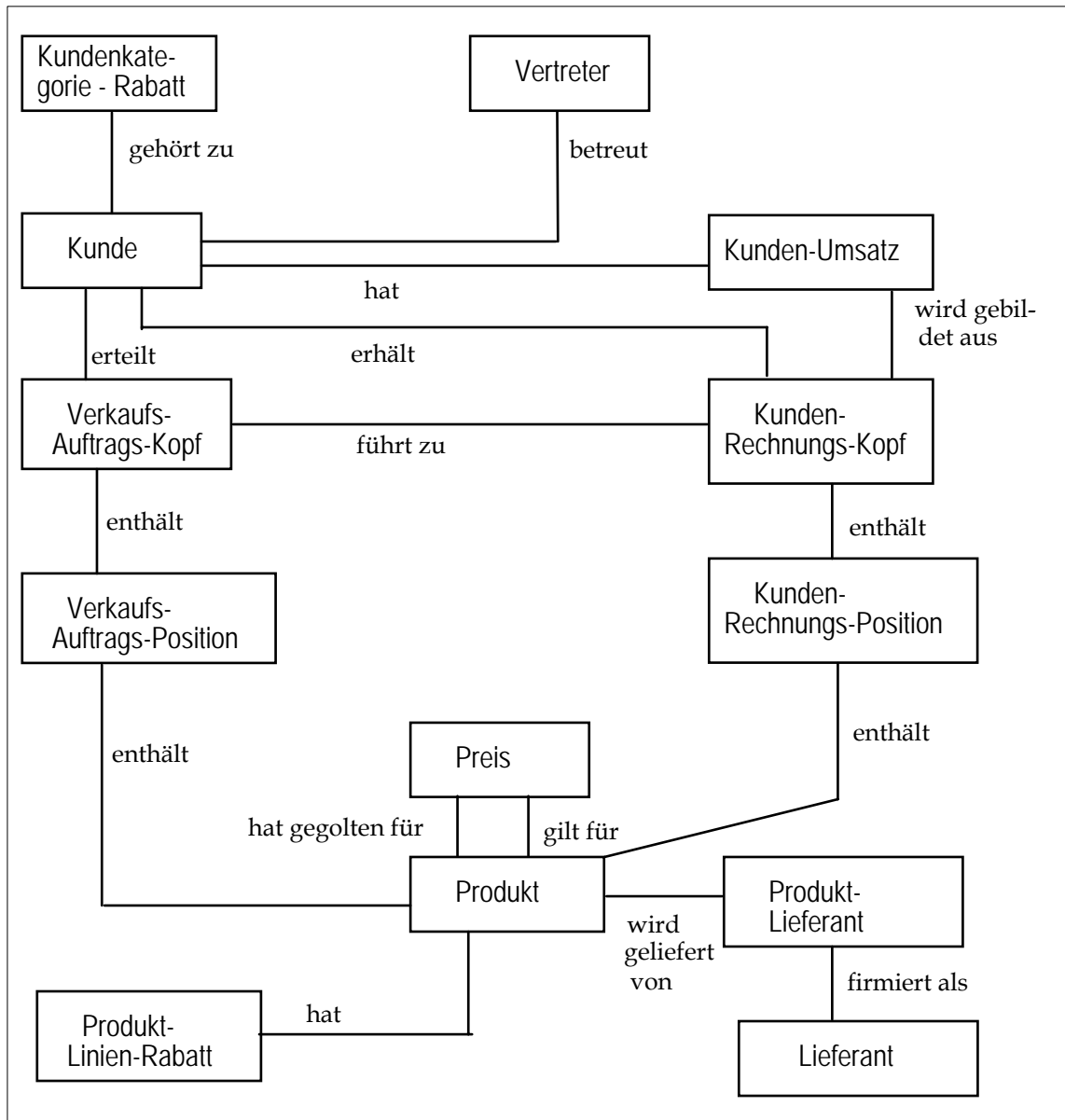
FACH

Fach	SWS	DozNr
MATH	4	002
PROG	4	004
DABA	6	004
CASE	6	001
KI	4	001
BS	2	002
STAT	2	001

Ein Joining der semantisch verbundenen Relationen ist über das in der Relation DOZENT als *Primärschlüssel* (primary key) und in der Relation FACH als *Fremdschlüssel* (foreign key) enthaltene Attribut **DozNr** möglich. Zwischen beiden Tabellen existiert eine referentielle Integrität (Fach ist abhängig von Dozent).

2.7.2 Komplexes Beispiel "Konzeptuelles Modell des Auftragswesens"

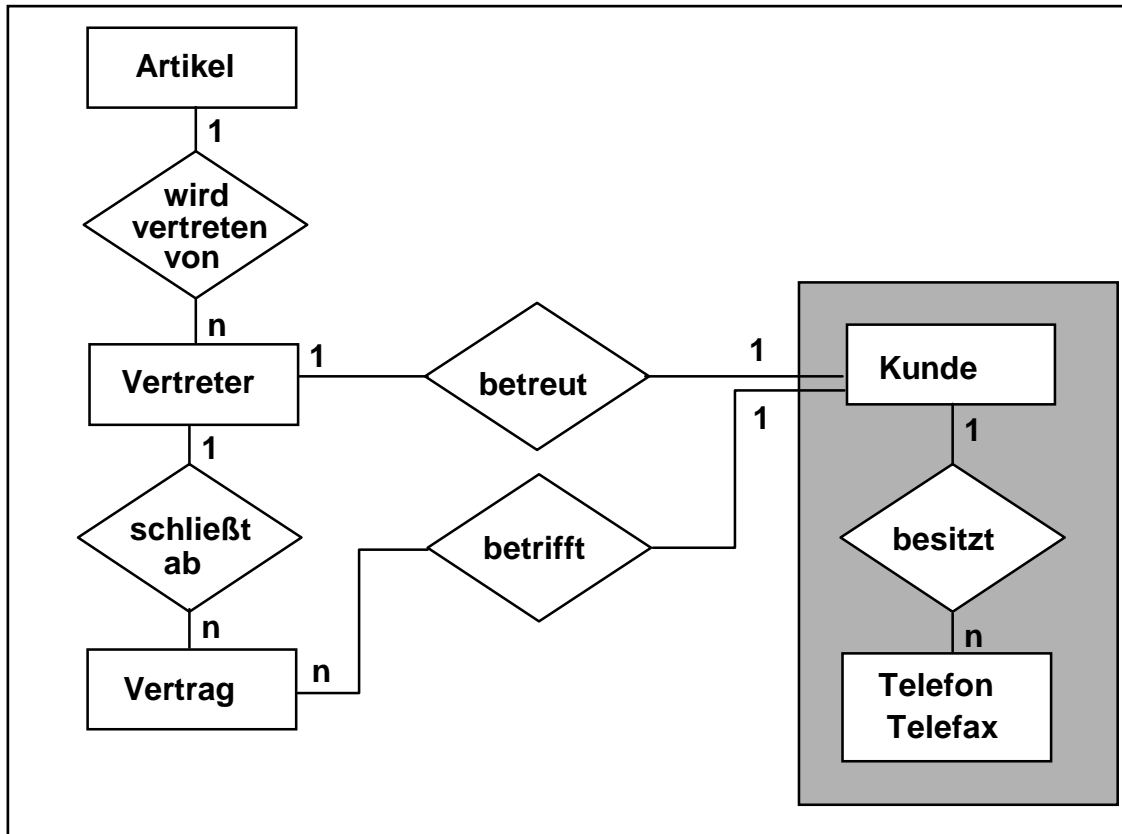
(Quelle: Biethahn, Muksch, Ruf: Ganzheitliches Informationsmanagement, Bd. II, Oldenbourg 1991)



Anmerkung: Aus Gründen der Übersichtlichkeit sind die Relationships neben die Verbindungslinien und ohne Rautenumrahmung geschrieben.

2.7.3 Übungsbeispiel ERM "Vertretertätigkeit" (Basisvariante)

Das Beispiel dient zur Übung der schrittweisen Vervollkommnung eines ERM bei Änderungen der semantischen Voraussetzungen für den untersuchten Gegenstand. Als Voraussetzung wird deshalb folgendes grob vereinfachtes Modell vorausgesetzt, das zunächst in seiner semantischen Aussage verbal interpretiert und später modifiziert werden soll.



Das **Umfeld** des Modells ist ein dienstleistendes Unternehmen, welches seine Produkte/Leistungen den Kunden über Vertreter anbietet.

Eigenständige nur durch eindimensionale Attribute beschriebene **Entities** sind Artikel, Vertreter und Vertrag. Das Entity Kunde besitzt hingegen **Gruppenattribute** (ein Kunde kann mehrere Telefon- und/oder Telefaxanschlüsse besitzen). Die hier gegebene zusätzliche Dimension der Attribute wird formal durch die Bildung eines eigenständigen Entitys "Telefon Telefax" aufgelöst.

2.8 Methodik der Entity-Relationship-Modellierung

2.8.1 Grenzen der ER-Modellierung

Durch die Ausrichtung des ERM auf das relationale DM als formales Zielmodell werden Grenzen für die realitätsnahe Modellierung gesetzt. Insbesondere betrifft das die nachfolgenden zwei Probleme:

Ursache	Lösung	Beispiel / Bemerkung
mehrwertige Attribute (für ein Objekt mehrfach auftretende Attribute) verletzen die 3NF/4NF	eigener Entity-Typ für die Attributsgruppe	Entity-Set Posten = Zeilen einer Rechnung, die mehrere Positionen beinhaltet. Beziehungstyp <u>immer</u> Master-Detail-Beziehung
große bzw. sich häufig ändernde Domänen überfordern die deskriptive Definition im DBMS	eigener Entity-Typ für die Domäne	Typische Attribute: Schlüssel, Text Beziehungstyp <u>aus Domänensicht</u> : 1:n, dependent

Fernerhin verbieten Datenmodellierungstools gegebenenfalls die Attributierung von Relationships, was folgende Konsequenz hat:

Ursache	Lösung	Beispiel / Bemerkung
Tool kann keine Attribute für RS-Typ erfassen	eigener Entity-Typ für Beziehung	Entity-Set Angebot zur Aufnahme des Attributs Preis für eine Beziehung "Produkt wird angeboten".

2.8.2 Methodische Arbeitsschritte für die praktische Er-Modellierung

1. Umfeld bestimmen (z. B. Abteilung Beschaffung)
2. Wesentliche Objekte als Entity-Kandidaten wählen (z. B. Lieferanten, Artikel, Bestellungen)
3. Prüfen, ob gewählte Objekte Entity-Sets sind.
Beachten Sie: Ein Entity-Set mit nur einem möglichen Exemplar ist ein Modellierungsfehler!
4. Attribute zur Beschreibung des Entity ermitteln.
5. Treten Attribute nicht als Einzelwert auf, sondern als Attributsgruppe (z. B. Konten eines Lieferanten), wird die Attributsgruppe ein eigenes Entity (z. B. Bankverbindung)
6. Sollen Attributswerte eines Entity-Sets aus einer Systematik gewählt werden (dynamische Domäne), wird die Systematik ein Entity (z. B. Bankleitungsverzeichnis).
7. Beziehungen (Relationship) zwischen den Entities nach Grad (Cardinality) und Verbindlichkeit (Mandatory, Dependent) bestimmen.
8. Sonderfall Vererbung (Inheritance) erkennen.
9. Abschließende Plausibilitätskontrollen durchführen (Kein Attribut darf im ERM doppelt vorkommen, Beziehungen (Relationships) mit Attributen müssen ggf. in ein Entity umgewandelt werden).

Nicht alle semantischen Spezifikationen des Modells können eindeutig in physikalische Spezifikationen des Modells transformiert werden. Deshalb sind ggf. Modellkorrekturen im physischen Datenmodell erforderlich.

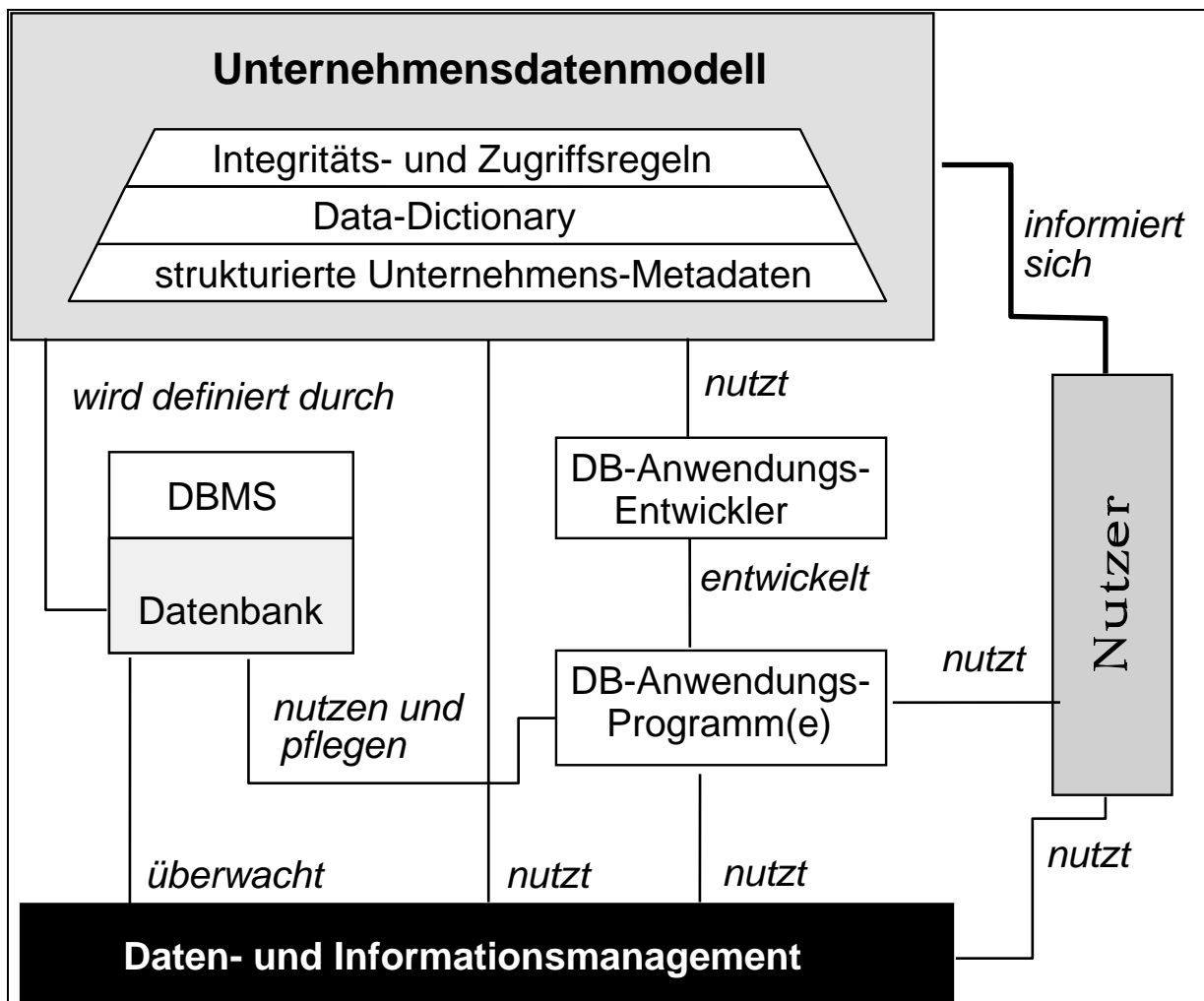
2.9 Unternehmensdatenmodelle (UDM) / Data Warehouse

2.9.1 Begriffsdefinition UDM

Unter einem Unternehmensdatenmodell versteht man das konzeptuelle Modell der Unternehmensdaten, d. h. aller zentral und/oder dezentral zu verwaltenden informationellen Objekte und Beziehungen der "realen Welt" des Unternehmens. Unternehmensdatenmodelle können durch Analyse und/oder Struktursynthese betrieblicher Subdatenbasen gewonnen werden. Das Entity-Relationship-Modell eignet sich - ergänzt durch Konstruktionsoperatoren zur Erzeugung neuer semantischer Objekte - insbesondere zur Synthese und Deskription von Unternehmensdatenmodellen.

Eine spezielle Form der Unternehmensdatenmodellierung sind die spezifisch für die Unterstützung von Informations- und Entscheidungsprozessen definierten Modelle von Data-Warehouse- bzw. Data-Mart-Applikationen.

2.9.2 Integration und Systemdenken als Ziel des UDM



2.9.3 Hauptziele von Unternehmensdatenmodellen

1. Dokumentation der Unternehmensressource 'Information'.
2. Entwurfsgrundlage für neue Informationssysteme.
3. Entwurfsgrundlage für ein Data Warehouse oder Data Marts.
4. Nachdokumentation vorhandener Informationssysteme.
5. Grundlage für Auswahl und Anpassung von Standardsoftware.
6. Grundlage für die Einarbeitung und Schulung neuer Mitarbeiter.
7. Grundlage für die Einordnung und Fortschrittsbeurteilung von DV-Projekten.
8. Sichtbarmachen von Überdeckungen und Schnittstellen zwischen Teilprojekten.
9. Grundlage der Ablaufoptimierung von Geschäftsprozessen.

2.9.4 Schritte zu einem detaillierten semantischen Datenmodell

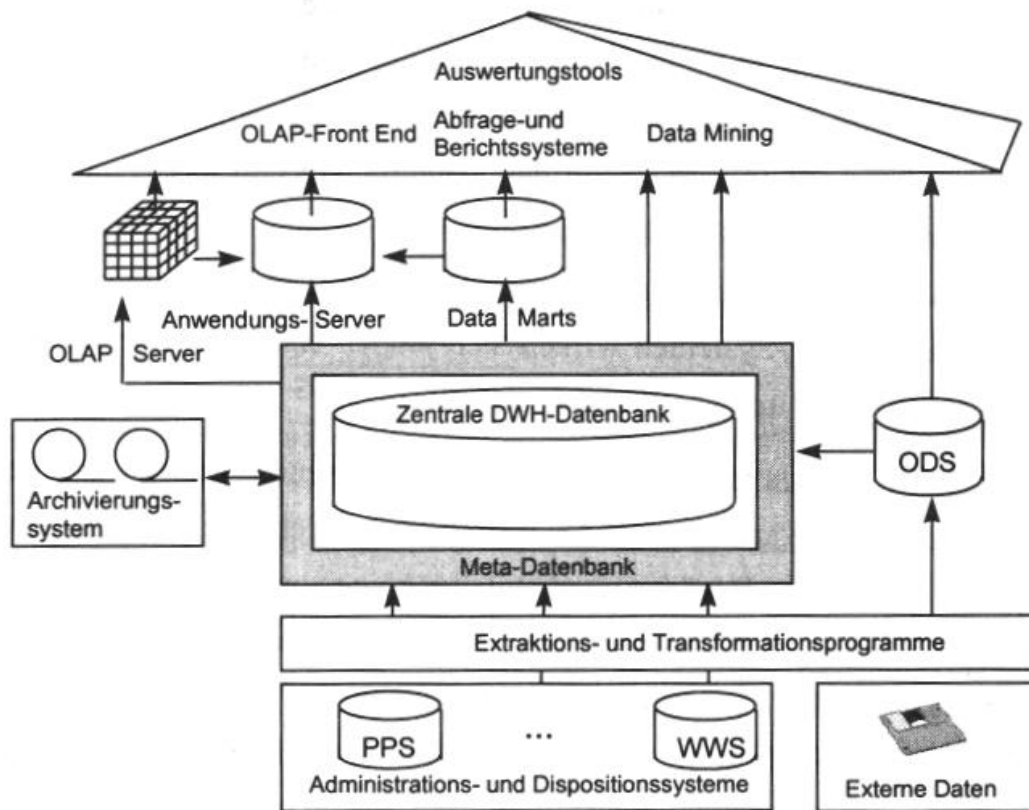
- (1) Analyse bestehender Informationselemente
- (2) Analyse und Beschreibung der relevanten Objekttypen
- (3) Analyse und Beschreibung der identifizierenden Attribute
- (4) Analyse und Beschreibung der relevanten Beziehungstypen sowie der Art dieser Beziehungstypen
- (5) Überprüfung des vorliegenden groben semantischen Datenmodells (Entity-Relationship-Darstellung (ERM))
- (6) Auflösung von (M:N)-Beziehungstypen und von bedingten Beziehungstypen
- (7) Bereinigung von Beziehungs-Inkonsistenz und -Redundanz
- (8) Analyse und Beschreibung der relevanten Attribute inklusive ihrer Zuordnung zu den ermittelten Objekt-Typen
- (9) Analyse und Beschreibung von Integritätsbedingungen
- (10) Überprüfung des detaillierten Datenmodells

Quelle: Münzenberger: "Pragmatische Datenmodellierung", ONLINE 1/90

2.9.5 Data Warehouse / Data Marts

Definition: Konsistente unternehmensweite (Data Warehouse) oder unternehmensbereichsweite (Data Mart) Datenbasis mit dauerhaftem systematisierenden Charakter. Das Data Warehouse wird getrennt von operativen Datenbanken betrieben und dient der Aufbereitung von Informationen zum Zwecke der Unternehmensführung. Typisch ist die sogenannte Multidimensionalität der Daten. Man versteht darunter die Speicherung primärer Fakten-Informationen (z. B. Umsatzdaten) indiziert und damit systematisiert nach mehrfachen Dimensionen (Zeit, Warengruppe, Territorium usw.).

Komponenten eines Data Warehouses (vereinfacht):



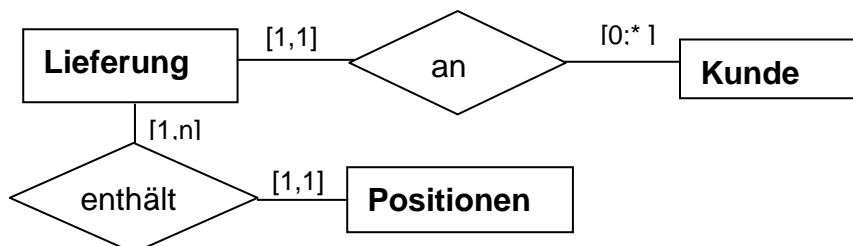
Quelle : Mucksch H./ Behme W., Das Data Warehouse-Konzept, 2000, S. 14

Eine Besonderheit von Data Warehouse-/Data Mart-Datenmodellen besteht darin, dass sie aus Gründen der Zugriffseffektivität häufig **nicht normalisiert** werden.

Eine Vertiefung dieser Problematik erfolgt in der Lehrveranstaltung "Komplexe Datenbankanwendungen" des Masterstudiums!

Kontrollfragen:

1. Geben Sie im Beispiel 2.7.2 den Grad und die Verbindlichkeit der Beziehungen zwischen den Entities in der Notationsform [...,...] an.
2. Wie würde sich das Datenmodell im Beispiel 2.7.3 verändern, wenn jeweils nur genau eine Telefonnummer und eine Telefaxnummer für jeden Kunden erfasst wird?
3. Überführen Sie das folgende ERM - ergänzt um selbstgewählte sinnvolle Attribute - in ein relationales Datenmodell der 3NF.



4. In welcher Beziehung stehen die Begriffe "referentielle Integrität" und "Master-Detail-Beziehung"?

Teil 3: Datenmodellierung mit PowerDesigner Data Architect

Die Software **PowerDesigner Data Architect** von Sybase (auch in der Version 9.0 aufwärtskompatibel vorliegend) ist ein Beispiel für die Softwaregruppe "Datenmodellierungstools", die den rechnerunterstützten professionellen Datenbankentwurf über die Stufen "Semantische Datenmodellierung" → "Physische Datenmodellierung" → "Generierung von SQL-Scrips" unterstützen.

Für Übungszwecke können Sie diese Software zur Erstellung und/oder Nachbearbeitung von Datenbankentwürfen bzw. zum Reverse Engineering von Datenbanken und physischen Datenmodellen nutzen. Im Datenbanklabor ist die Software unter Windows-2000 im Fenster "Datenbanken Programme" installiert. Nach dem Start der Software wird zunächst die Arbeitsfläche für das Neuerstellen/Laden eines konzeptuellen Datenmodells (CDM) angeboten (siehe nachfolgende Abbildung).

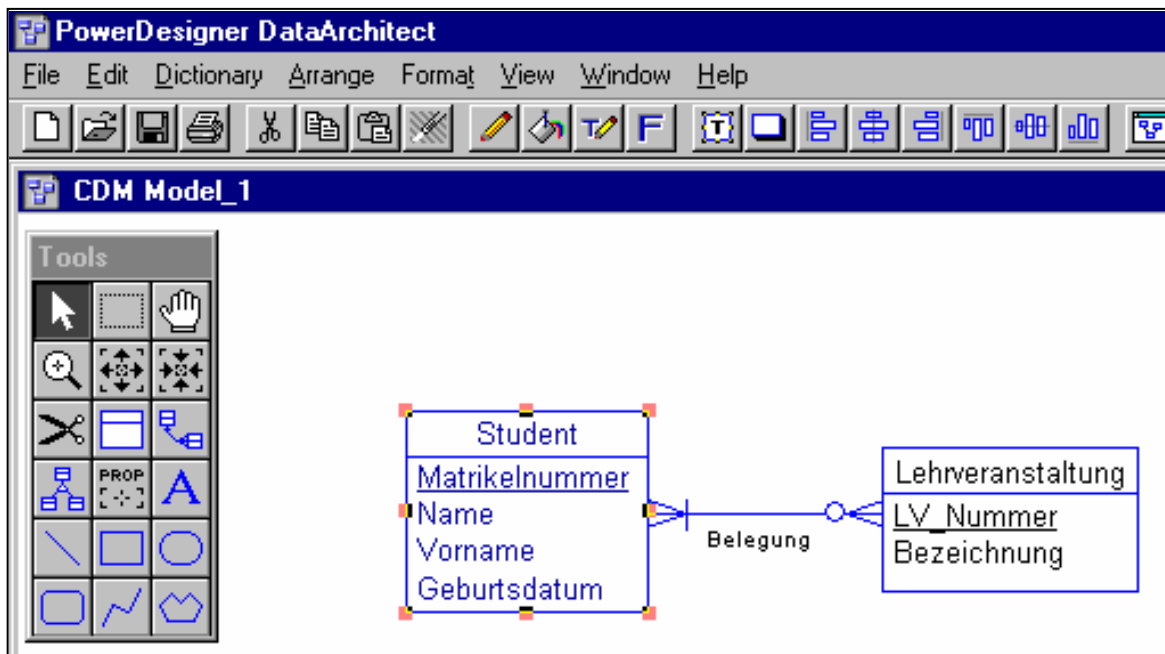
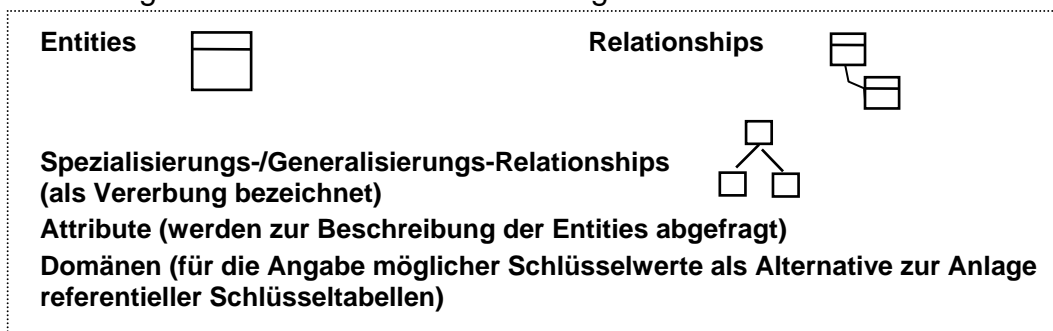


Abbildung: Arbeitsschema DataArchitect "Konzeptuelles Datenmodell" (CDM)



Die wichtigsten Elemente der Modellierung auf der Ebene des CDM sind



Die Kardinalität (Grad und Verbindlichkeit) der Relationships wird symbolisch dargestellt.

Die Bedienung erfolgt jeweils alternativ über Menü oder Wahl der angegebenen Symbole aus der Toolbox. Durch Doppelklick auf ein Entity wird eine Box zur Beschreibung (unter anderem Angabe des Namens, der Attribute und der Primärschlüssel) des betreffenden Entitys geöffnet. Durch Doppelklick auf die Verbindungslinie zwischen zwei Entities wird eine Box zur Beschreibung (unter anderem Angabe der Kardinalität nach der in Abschnitt 2.3.4 beschriebenen Methode) des betreffenden Relationships geöffnet. Ein beschreibender Titel des Modells kann über die Menüpunkte "Options" → "Add Title" hinzugefügt werden.

Nach Speicherung des CDM wird über die Menüpunkte "Dictionary" → "Generate Physical Model ..." das physische Datenbankmodell (PDM) generiert. Das PDM ist auf ein über Menü wählbares Datenbankmanagementsystem (für die Grundausbildung INFORMIX) abgestimmt und beinhaltet die Elemente

- Tabelle (mit Spaltenangabe und primary key)  und
- Referenz (mit Angabe des foreign key) 

Im Interesse der Performance der Datenbankapplikation können am physischen Modell Veränderungen (Denormalisierungen, Veränderung des primary key) vorgenommen werden, die bei späteren Veränderungen des CDM erhalten bleiben.

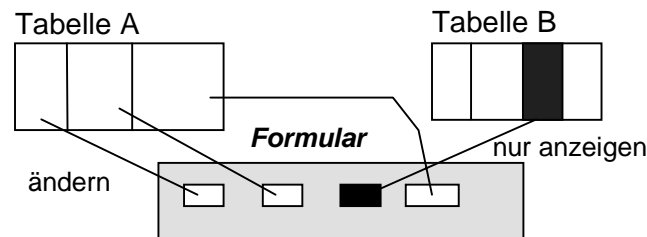
Als dritter Schritt des Entwurf einer Datenbank wird über die Menüpunkte "Interface" → "Generate Database ..." der konkrete Datenbankentwurf insgesamt oder für ausgewählte Teile des Modells in Form eines SQL-Scrips generiert. Als Ausgabemedium sollte im Übungsbetrieb dabei immer die Ausgabeform "Generate Script" gewählt werden, um den Modellierungsprozess nachvollziehen zu können.

Weitere ausführliche Hinweise zur Nutzung des Data Architect erhalten Sie über das HELP-Menü.

Teil 4: Widerspiegelung formaler Datenbeziehungen in Formularen (Beispiel MS-Access)

Definierte formale Datenbeziehungen für das relationalen Datenmodell bilden auch die Grundlage der in DBMS-Tools integrierten Formularentwicklungsumgebungen. Am Beispiel des DBMS MS-Access werden nachfolgend die wesentlichsten Formularbestandteile bzw. Formulararten skizziert, die eine Verbindung mehrerer Relationen implizieren:

1. Anzeige eines Fremdfeldes aus einer korrespondierenden Tabelle:



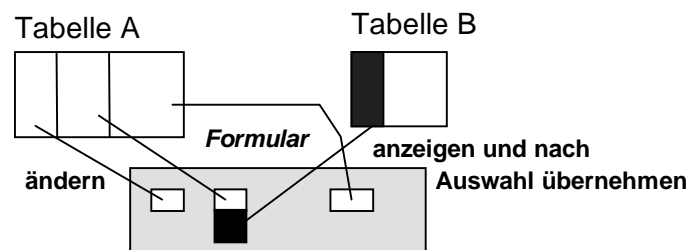
Methode: LookUp-Join, Formularfeld Steuerelementinhalt (Eigenschaft)

Funktion DomWert(Ausdruck; Domäne, Kriterien)

Beispiel: Eintrag der Eigenschaft *Steuerelementinhalt*:

=DomWert("[BANK_NAME]"; "[BANKLZ]"; "[BLZ]=[bankleitzahl]")

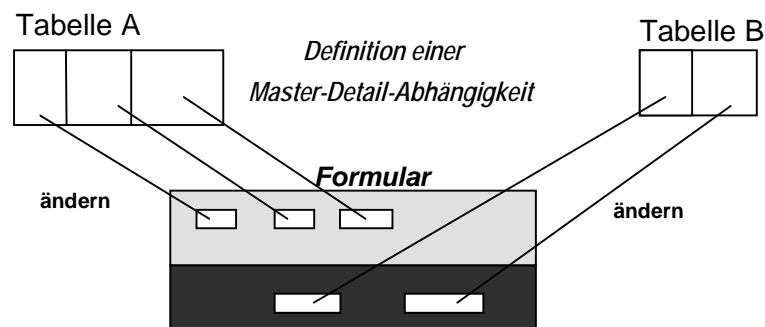
2. Auswahl aus einer Domänenrelation:



Methoden: Anzeige und Auswahl aus Listenfeld oder Kombinationsfeld

Beispiel: Eintrag der Eigenschaft *Datensatzherkunft* für das Kombinations- oder Listenfeld: Tabellename oder Name einer Abfrage

3. Bearbeitung abhängiger Relationen (Master-Detail-Beziehung):



Methoden: Haupt-Unter-Formular

Teil 5: Sachwörter Datenmodellierung und Datenbanken

Aktualisierung (einer Datenbank)	Daten einer Datenbank auf den aktuellen Stand bringen (auch Pflege oder Wartung einer Datenbank).
Attribut	Bezeichnung einer Eigenschaft eines Objektes (Entität) oder einer Beziehung (siehe auch Feld).
Auswertung (einer Datenbank) auch Abfrage bzw. Selektion	Daten einer Datenbank für konkrete Aufgaben benutzen.
API	Application Programming Interface (einer DB)
Backend	Im Hintergrund (Server) agierender Teil eines DBMS.
BCNF	Boyce-Codd-Normalform.
Benutzer (einer Datenbank) auch User	Mitarbeiter, der die in einer Datenbank gespeicherten Informationen für seine fachliche Arbeit nutzt.
Beziehung (Relationship)	Funktionale Verbindung zwischen zwei Entitäten.
CODASYL	Normenausschuss Conference on Data Systems and Languages der Data Base Task Group (DBTG), der u.a. die CODASYL-DDL für das Netzwerkmodell normierte.
Constraint	Prüfbedingung, die als Datenbankeigenschaft formuliert werden kann (CHECK, REFERENCES, PRIMARY KEY, UNIQUE).
Data Mart	Data Warehouse für einen kleinen semantisch abgegrenzten Anwendungsbereich.
Data Warehouse	Datenbankapplikation, die zur Sammlung und nutzergerechten Aufbereitung von Daten dient und als Datenquelle die Verbindung zu anderen Datenbanken nutzt. Ein typisches Merkmal ist die Multidimensionalität von Daten.
Datenbank (DB)	Doppelterwendung zur Bezeichnung <ul style="list-style-type: none">- aller zu einer Datenbanktechnologie gehörenden Tabellen (Relationen) oder- eines Softwareproduktes zur Bearbeitung von Datenbankaufgaben (auch Datenbanksystem genannt).
Datenbankadministrator	Mitarbeiter, der für die Entwicklung und technologische Betreuung einer Datenbanktechnologie zuständig ist.
Datenbank-Applikation	Software, die eine (mehrere) Datenbank(en) nutzt.
Datenbankmanagementsystem (DBMS)	Softwaresystem zur Verwaltung von Datenbanken.
Datenbankschema	Menge $S := \{R_1, \dots, R_n\}$ von Relationenschemata, die in einer Datenbank organisiert sind.
Datenbankdatei	Eine in der Datenbank gespeicherte Tabelle (Relation).
Datenbanktechnologie	Sammelbegriff für typische datenbankorientierte Prozessabläufe.
Datenmodell	Struktur zur geordneten Darstellung und damit modellhaften Abbildung von Daten.
Datensatz	Gesamtheit der Angaben in einer Zeile der Tabelle (Tupel).

Datensatzzeiger	Interner Zeiger der Datenbank, der auf den jeweils aktuellen Datensatz zeigt.
Datensicht (View)	Ausschnitt aus einer Datenbank, der speziell auf eine(n) Aufgabe/Nutzer zugeschnitten ist.
DDL	Data Definition Language (Datendefinitionssprache).
DML	Data Manipulation Language (Datenmanipulationssprache).
Domäne	Wertebereich für ein Attribut (s. auch Wertevorrat).
Entity (Entität)	Semantische Einheit als Teil der realen Welt (s. auch Objekt).
Entity-Relationship-Modell (ERM)	Modell zur Darstellung semantischer Datenbeziehungen.
Entity-Set	Menge gleichartiger Entities.
Entity-Typ	Durch die Potenzmenge aller gleichartigen Entities definierte Beschreibung der Entitystruktur.
Eigentümer (einer Datenbank) siehe auch Owner	Mitarbeiter, der für den Inhalt der in der Datenbank gespeicherten Informationen verantwortlich ist und unter dessen Login die DB angelegt wurde.
Existenzabhängigkeit	Abhängigkeit zwischen den Tupeln von zwei Relationen, die ausdrückt, dass wechselseitig ein Tupel in einer Relation nur dann existieren kann, wenn er mindestens einen "Partner"-Tupel in der anderen Relation besitzt.
Feld (Attribut)	Bezeichnung einer Spalte aus der Tabelle (Datei).
Feldtyp	Typ (z. B. numerisch, character) der in einem Feld gespeicherten Daten.
Fremdschlüssel (foreign key)	In einer Relation enthaltener Schlüssel, der sich auf ein primär in einer anderen Relation enthaltenes Attribut bezieht.
Frontend	Im Vordergrund (Client) agierender Teil eines DBMS (Nutzerschnittstelle).
Integrität	Widerspruchsfreiheit der in der Datenbank erfassten Daten (Konsistenz).
Katalog	Verzeichnis aller zu einer Datenbanktechnologie gehörigen Dateien.
Join	Verbindung zweier Tabellen (Relationen).
Master-Detail-Beziehung	Über- (Master) Unter (Detail) - geordnete Beziehung zwischen zwei Relationen (siehe auch Existenzabhängigkeit).
Konsistenz	Logische Widerspruchsfreiheit aller in einer Datenbank gespeicherten Informationen (Integrität).
Normalisierung	Verfahren zur Beseitigung von Redundanzen und Anomalien innerhalb und zwischen Tabellen (Relationen) durch verlustfreie Zerlegung der Relation.

Objekt	Ausschnitt aus der realen Welt, bildet eine semantische Einheit, ist klassifizierbar (z. B. Baum, Zweig, Blüte) und besitzt funktionale Eigenschaften, die für alle Objekte gleichen Typs gleich sind.
ODMG	Object Data Management Group, inoffizielle Standardisierungskommission für OODBS, Erarbeitung des ODMG-93-Standards und des ODMG-93-Datenmodells.
OLAP (Online Analytical Processing)	Datenbanktyp zur Verwaltung mehrdimensionaler Informationen vorwiegend für die Lösung von Entscheidungsaufgaben im Leitungsprozess (Data Warehouse).
OODBS	Objektorientiertes Datenbanksystem.
Owner	Datenbankobjekt-Eigentümer.
Primärschlüssel (primary key)	Mindestmenge verbundener Attribute einer Relation, die zur eindeutigen Unterscheidung eines Tupels von allen anderen Tupeln notwendig ist.
Projektion	Neubildung einer Tabelle durch Auswahl von Spalten aus einer gegebenen Tabelle.
RDBMS	Relationales DBMS.
Redundanz	Mehrfache Speicherung gleicher Informationen.
Relation	Exakte Begriffsbezeichnung der mathematischen Logik für die "Datenbank-Jargon-Bezeichnungen" Tabelle bzw. Datei.
Relationship	Beziehung zwischen Objekten (Entities).
Relationship-Set	Menge gleichartiger Relationships.
Relationship-Typ	Durch die Potenzmenge aller gleichartigen Relationships definierte Beschreibung der Relationshipstruktur.
Schlüssel (auch Schlüssel- feld(er), Schlüsselattribut(e))	Ausgewählte(s) Feld(er) einer Tabelle, auf Grund dessen/derer jeder Datensatz (jede Zeile einer Tabelle) eindeutig von allen anderen Datensätzen (Zeilen der Tabelle) unterschieden werden kann.
Selektion	Auswahl von Datensätzen aus einer Tabelle.
Semantik	Betrachtung von Informationen aus der Sicht ihrer Bedeutung.
semistrukturierte Daten	Daten, die nicht nach einem externen Schema (z. B. Relation) sondern nach internen Gliederungsebenen strukturiert werden (z. B. ein Buch). Als Datenmodell wird häufig eine XML-Auszeichnungssprache genutzt.
SERM	Strukturiertes Entity-Relationship-Modell nach Sinz.
SQL	Structured Query Language - standardisierte Datenbanksprache für das relationale Datenmodell.
Stored Procedures	In der Datenbank gespeicherte Prozeduren, die von allen Datenbankanwendern genutzt werden können.
Suchen	Umgangssprachlich für Selektion.

Syntaktik	Betrachtung von Informationen aus der Sicht ihrer formal richtigen Darstellung.
Tabelle	Anordnung von Daten in Tabellenform (Zeilen=Datensätze; Spalten=Felder); als Bestandteil einer relationalen Datenbank (exakte Bezeichnung: Relation).
Trigger	Automatisch von Datenbankfunktionen ausgelöste Operationen über die Datenbank.
VDBMS	Verteiltes DBMS.
Verbund	Siehe Join.
Verbundschlüssel (auch Fremdschlüssel)	Ausgewählte(s) Feld(er), die in mehreren Tabellen mit gleicher Bedeutung und mit zumindest in einer Teilmenge gleichem Wertevorrat auftreten und die auf Grund dieser Eigenschaft semantisch zusammengehörige Informationen über mehrere Tabellen verbinden können.
View	Siehe Datensicht.
Wertevorrat (auch Domäne)	Gesamtheit aller für ein Feld einer Tabelle möglichen Werte.

Teil 6: Literaturhinweise

Die Hinweise sind generell als wahlweise nutzbare Ergänzungsliteratur zu verstehen. Bitte beachten Sie die aktualisierten Literaturhinweise unter "<http://web.f4.fhtw-berlin.de/zschockelt/labor/>", Rubrik "Lehre" → "Literatur" oder direkt <http://dblabor.f4.fhtw-berlin.de/lehre/literatur1.asp>.

- Studienliteratur Informatik: Lehrbriefreihe Datenbanken, VMS Verlag Modernes Studieren Hamburg - Dresden GmbH
- Biethahn, Muksch, Ruf: Ganzheitliches Informationsmanagement, Band II, Oldenburg, München Wien 2000
- Bleimann/Dippel/Turetschek/Wente: Betriebsinformatik, Hanser Studienbücher, München Wien 1990, Abschnitt 13 (Datenmodellierung)
- Connolly, T./Begg, C./Strachan, A.: Datenbanksysteme, Addison-Wesley, Bonn 2002
- Elmasri/Navathe: Grundlagen von Datenbanksystemen 3. Auflage, Addison-Wesley 2002
- Gabriel/Röhrs: Gestaltung und Einsatz von Datenbanksystemen, Springer Lehrbuch, Berlin Heidelberg 2002
- Hald,A./Dallmüller,K.: Datenbankmanagement, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Wiesbaden 2002
- Hars, A.: Referenzdatenmodelle, Grundlagen effizienter Datenmodellierung, Gabler 1994
- Heuer, A./Saake, G.: Datenbanken Konzepte und Sprachen, International Thomson Publishing 2000
- Hughes, J. G.: Objektorientierte Datenbanken, Carl Hanser Verlag, München Wien 1992
- Jackson: Entwurf relationaler Datenbanken, Carl Hanser Verlag, München Wien 1990
- Jarosch, H.: Datenbankentwurf, Friedr. Vieweg & Sohn Verlagsges. mbH, Wiesbaden 2002
- Kemper/Eickler: Datenbanksysteme, 3. Auflage, R. Oldenbourg Verlag, München Wien 1999
- Kießling, W./ Köstler, G.: Multimedia-Kurs Datenbanksysteme , Springer Verlag 1998, mit CD-ROM, zahlreichen Abb., Animationen.
- Kleinschmidt/Rank: Relationale Datenbanksysteme, Springer Verlag 2001
- Lackes/Brandl/Siepermann: Datensicht von Informationssystemen, Springer Verlag 1998 mit CD-ROM (siehe auch <http://www.wiso.uni-dortmund.de/LSFG/WI>)
- Lusti, M.: Dateien und Datenbanken, Springer Verlag 1997, 3. Auflage
- Marsch/Fritze: Erfolgreiche Datenbanktechnologie mit SQL, Friedr. Vieweg & Sohn Verlagsges.mbH, 5. Aufl. Wiesbaden 1999
- Matthiesen, Günter/Unterstein, Michael: Relationale Datenbanken und SQL, Addison-Wesley, Bonn 1997
- Moos,A./Daues,G.: Datenbank-Engineering, Friedr. Vieweg & Sohn Verlagsges.mbH, Wiesbaden 1997
- Müller-Ettrich u.a.: Fachliche Modellierung von Informationssystemen, Addison-Wesley, Bonn, München 1993
- Pernul, G./Unland, R.: Datenbanken in Unternehmen, R. Oldenbourg Verlag, München 2001
- Rauh/Stickel: Konzeptuelle Datenmodellierung, B. G. Teubner, Stuttgart Leipzig 1997

- Sauer, H.: Relationale Datenbanken, Theorie und Praxis inklusive SQL-3, 4. Auflage, Addison-Wesley, Bonn, München 1999
- Scheer: Wirtschaftsinformatik - Informationssysteme im Industriebetrieb, Springer Verlag Berlin Heidelberg 1994, Teil A II.2 (Daten)
- Scheer: Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse, 4. Aufl., Springer Verlag Berlin Heidelberg 1997
- Scheer: Architektur integrierter Informationssysteme (Grundlagen der Unternehmensmodellierung), Springer Verlag Berlin Heidelberg 1997
- Schicker: Datenbanken und SQL (Eine praxisorientierte Einführung), B.G.Teubner Verlag, Stuttgart 1996
- Schlagether/Stucky: Datenbanksysteme, Konzepte und Modelle, B. G. Teubner Verlag, Stuttgart 1983
- Schwinn, H.: Relationale Datenbankmanagementsysteme, Carl Hanser Verlag München Wien 1992
- Steiner, R.: Theorie und Praxis relationaler Datenbanken, Friedr. Vieweg & Sohn Verlagsges.mbH, Braunschweig/Wiesbaden 4. Auflage 2000
- Vetter, M.: Aufbau betrieblicher Informationssysteme mittels objektorientierter konzeptioneller Datenmodellierung, B. G. Teubner Stuttgart 1991
- Vetter, M.: Informationssysteme in der Unternehmung, B. G. Teubner Stuttgart 1994
- Vossen, G.: Datenbanktheorie, International Thomson Publishing GmbH, 1995
- Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, R. Oldenbourg Verlag, München Wien 2000, 4. Auflage
- Zehnder, C. A.: Informationssysteme und Datenbanken, ab 5. Aufl. 1989, B. G. Teubner Stuttgart
- Zilaki-Szabo: Wirtschaftsinformatik, R. Oldenbourg Verlag, München Wien 1993