

Inhalt

1. Historische Entwicklung
2. Informationsbedarfsplanung
3. Konzeptioneller Entwurf/semantische Datenmodellierung
 - Entity-Relationship-Modell
 - Unternehmensweite Datenmodellierung
4. Logischer Entwurf
 - Hierarchisches Modell
 - Netzwerkmodell
 - **RELATIONALES MODELL**
 - Denormalisieren
 - Exkurs: Wann ist ein Datenbanksystem relational?

Inhalt

5. Datenbanksprachen
 - SQL
 - Query by Example
- (6. MS Access)

Literaturliste

1. Datenmodellierung:

- Eberhard Stickel: **Datenbankdesign: Methoden und Übungen**, Gabler-Verlag, Wiesbaden 1991 - relativ einfach und leicht verständlich
- Vinek, Rennert, Tjoa: **Datenmodellierung: Theorie und Praxis des Datenbankentwurfs**, Physica Verlag, 1982 - Pflichtwerk zur Datenmodellierung, teilweise aber sehr stark mathematisch und auf den Informatiker zugeschnitten

2. Datenbanken - allgemein:

- Zehnder: **Informationssysteme und Datenbanken**, Teubner Verlag, 4. Auflage, 1987 - recht gut
- Schlageter, Stucky: **Datenbanksysteme: Konzepte und Modelle**, Teubner Verlag, 2. Aufl., 1983 - recht gut, teilweise sehr "informatisch"

Literaturliste

- Date: **Relational Databases: Selected Writings**, Addison-Wesley Verlag, 1986 - der "Klassiker" im Bereich der relationalen Datenbanksysteme
- Neumaier: **Relationale Datenbanken**, Oldenbourg Verlag, 1989
- James Martin: **Einführung in die Datenbanktechnik**, Hanser Verlag, 5. Aufl., 1987 - beleuchtet auch betriebswirtschaftliche Aspekte des Datenmanagements

3. MS Access

- div.

Datenbankkenntnisse - wozu?

1. Lösung eigener Probleme

Ein Datenbankbenutzer sollte in der Lage sein, kleine Probleme selbst, ohne Beiziehung der zentralen EDV-Abteilung, zu lösen.

2. Informationsvorsprung

In einer (zentralen) Unternehmensdatenbank sind alle Unternehmensdaten gespeichert. Bei entsprechenden Datenbankkenntnissen und erlaubten Zugriffsberechtigungen kann der Benutzer alle entsprechenden Informationen aus der Datenbank abfragen.

Datenbankkenntnisse - wozu?

3. Bedeutung der Daten

"Die Daten des Unternehmens sind ein solch wichtiger und wertvoller Produktionsfaktor, daß der Verantwortliche für die Daten für so wichtig wie der Finanzchef zu erachten ist."

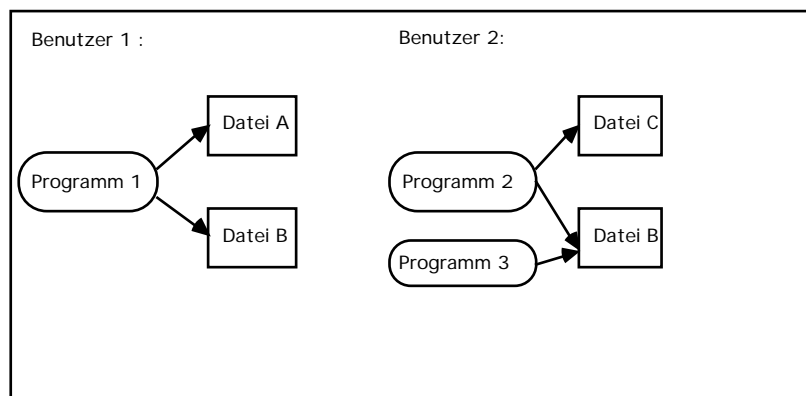
(James Martin: Einführung in die Datenbanktechnik, S. 299)

HISTORISCHE ENTWICKLUNG: Einzeldateien ohne Integration

Charakteristik:

- jeder Benutzer arbeitet mit privaten Dateien, teilweise unabhängig von anderen Benutzern
- Dateien werden nur von wenigen Programmen verwendet
- physische Datenabhängigkeit: Details der Datenformate, Sortierung, inhaltliche Beschreibung, etc. werden durch die Programme festgelegt
-
-

Einzeldateien ohne Integration



Einzeldateien ohne Integration

Nachteile:

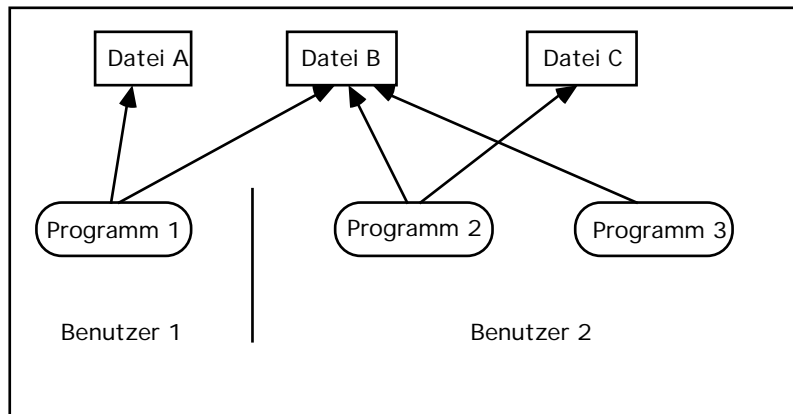
- **Daten werden mehrfach gespeichert (Redundanz), dadurch besteht die Gefahr, daß die Daten inkonsistent sind**
- **physische Datenabhängigkeit:**
 - EDV-Kenntnisse der Dateibenutzer
 - Dateistruktur muß genau bekannt sein
 - eine geringfügige Änderung der Datenstruktur erfordert die Änderung der darauf zugreifenden Programme

Physische Integration der Dateien

Charakteristik:

- **die Funktion der Datenerhebung, -kontrolle und -speicherung wird zentralisiert, diese Funktion wird durch den "Datenkoordinator" wahrgenommen**
- **die Daten werden einheitlich in einem "data dictionary" beschrieben: für jedes Feld einer jeden Datei erfolgt eine genaue Beschreibung des Inhaltes**
- **weiterhin physische Datenabhängigkeit**

Physische Integration der Dateien



Physische Integration der Dateien

Vor- und Nachteile:

- **Vorteile gegenüber Einzeldateien ohne Integration:**

- Daten sind nicht mehr mehrfach gespeichert

- **Nachteile:**

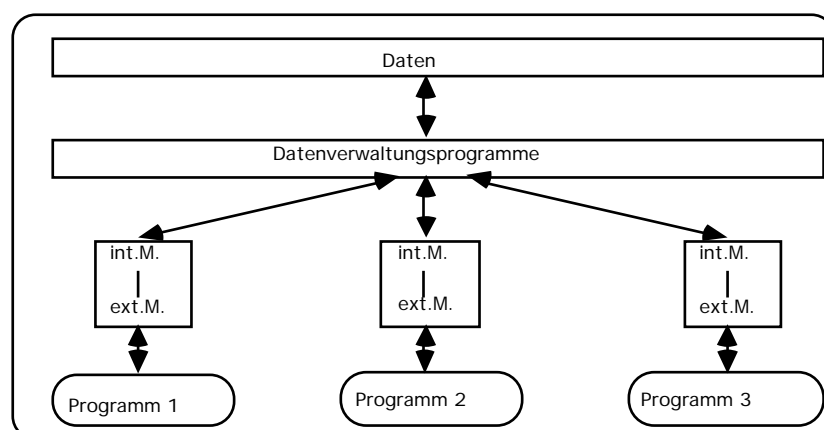
- bei gravierenden Änderungen der Datenstruktur sind wahrscheinlich sogar mehr Programme von Änderungen betroffen
- nunmehr ist es auch möglich, auf vertrauliche Daten zuzugreifen

Datenbanksystem: Zwei-Schichten-Modell

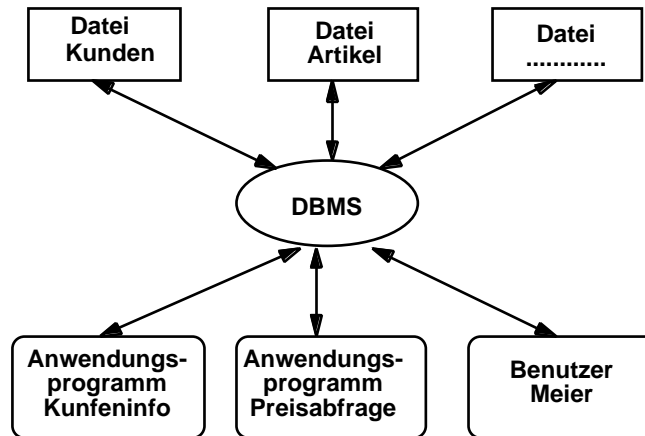
Charakteristik:

- es wird eine Trennungsebene zwischen den Anwendungsaufgaben (Benutzersicht) und den gespeicherten Daten (interne Sicht) eingeführt, um einen direkten Zugriff auf die Daten zu verhindern
- die Daten werden nunmehr von eigenen Programmen verwaltet, mit denen der Benutzer nichts mehr zu tun hat; er kommuniziert über eine standardisierte Schnittstelle
- Änderungen der physischen Struktur beeinflussen die Abbildungsprogramme, nicht aber das externe Modell

Zweischichten-Modell



Zweischichten-Modell

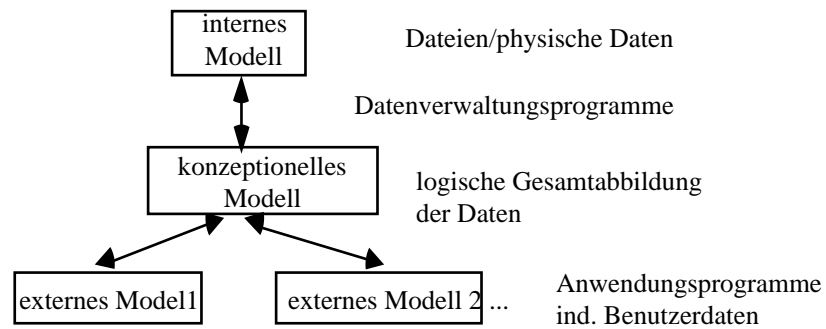


Datenbanksystem: Drei-Schichten-Modell

Logische Datenunabhängigkeit ist die Möglichkeit, logische Änderungen an der Datenbasis durchzuführen, ohne die darauf zugreifenden Anwendungsprogramme signifikant zu beeinflussen.

Zur Erfüllung dieser Forderung wird neben der internen und externen Sicht noch zusätzlich die sogenannte konzeptionelle Datensicht eingeführt.

Drei-Schichten-Modell



Drei-Schichten-Modell

Ein Datenbanksystem setzt sich also nunmehr aus folgenden Modellen zusammen:

1. externes Modell
2. konzeptionelles Modell und
3. internes Modell

Das Konzeptionelle Modell ist eine von allen Benutzern gemeinsam akzeptierte einheitliche Darstellung der realen Welt bzw. des Ausschnittes der realen Welt, der durch die Datenbank dargestellt werden soll.

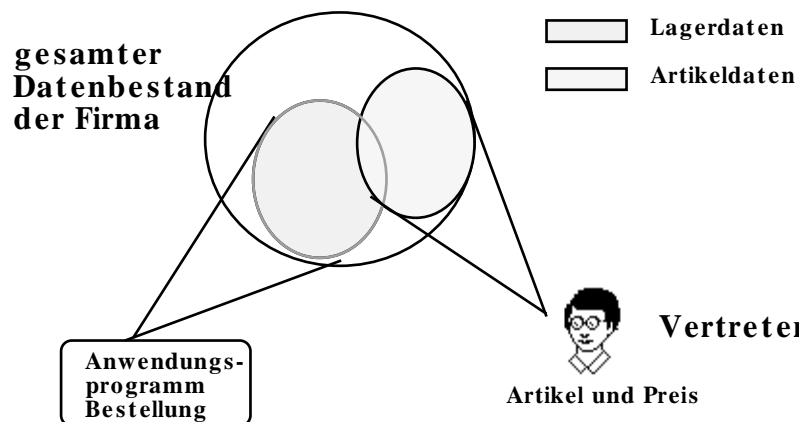
Drei-Schichten-Modell

Das externe Modell umfaßt den für diesen Benutzer interessanten (oder erlaubten) Teil der Daten, ihre Aufbereitung, sowie die Operationen, die darauf ausgeführt werden dürfen. Externe Modelle stellen die Benutzersichten auf die Datenbank dar. Jeder Benutzer hat individuelle Anforderungen an die Datenbank.

Das interne Modell enthält alle für die physische Implementierung notwendigen Aufgaben, wie Dateibenennung, Auswahl der Datenträger, Speicher- und Zugriffsmethoden (z. B. Indexdateien).

Drei-Schichten-Modell

Externes Modell:

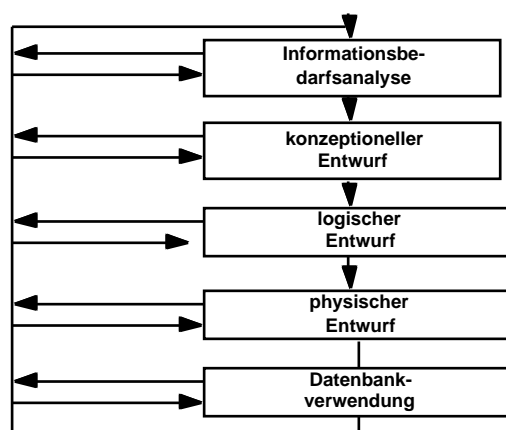


Drei-Schichten-Modell

3 Berufsgruppen nach ANSI/SPARC:

- **Anwendungsadministrator: externes Modell**
- **Unternehmensadministrator: konzeptionelles Modell**
- **Datenbankadministrator: internes Modell**

SCHRITTE BEIM DATENBANKENTWURF



INFORMATIONSBEDARFSANALYSE

DATA-ID-Verfahren:

1. Identifikation der Organisationseinheiten
 2. Identifikation der ggf. zu unterstützenden Aufgaben und der damit befaßten Organisationseinheiten
 3. Erstellung eines Anforderungs-Sammelplanes:
 - Identifizierung der zu befragenden Personen
 4. Anforderungs-Sammlung bei den Informationslieferanten:
Erhebung von:
 - Aufgabenziele
 - betroffene (Daten)Objekte
 - erforderliche Operationen
 - reguläre Ausnahmebedingungen
-

Informationsbedarfsanalyse

5. Filterung der Anforderungs-Sammelblätter
6. Satzklassifikation:
 - 1. Objekt (Daten): Verben wie "ist", "hat", "besteht aus", "betrifft", ...
 - 2. Operationen: Aktivitätsverben wie "machen", "erledigen", ...
 - 3. Ereignisse: bedingte Sätze wie "wenn", "falls", ...Erstellung von:
 - ♣ Datenanforderungsblättern
 - ♣ Operationsanforderungsblättern
 - ♣ Ereignisanforderungsblättern
7. Übertrag der jeweiligen Anforderungsblätter in entsprechende Verzeichnisse

Informationsbedarfsanalyse

"Aufgaben-/Organisationsmatrix"

	F & E	Einkauf	Produktion	Verkauf
Malzverträge abschließen				X
Produktionsplanung			X	
Gersteneinkauf		X		
Malzbestellungen		X	X	X

KONZEPTIONELLER ENTWURF / SEMANTISCHE DATENMODELLIERUNG

Datenmodellierung =

der Vorgang, das konzeptionelle Modell aus der Realität abzuleiten

Datenmodelle =

1. Formalismen, die diesen Vorgang unterstützen
2. Ergebnisse des Modellierungsprozesses

Konzeptioneller Entwurf

Nach Nijssen läuft der Prozeß der Datenmodellierung in folgenden Schritten ab:

1. Benennen:

Jedem Objekt der realen Welt und jeder Beziehung zwischen Objekten der realen Welt wird ein eindeutiger Name zugeordnet.

2. Auswahl:

Die Anzahl der Objekte der realen Welt ist für praktische Anwendungen viel zu groß. Man wählt daher nur den relevanten Ausschnitt der Realität aus.

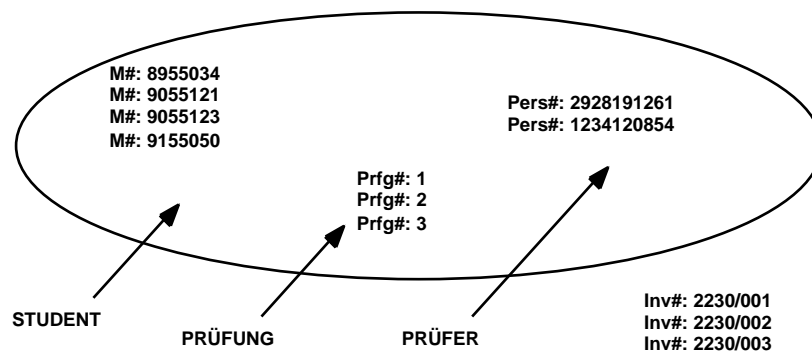
3. Klassifikation:

Die Menge der Objekte und Beziehungen werden einer Klasse zugeordnet. Diese Klasseneinteilungen ergeben sich oft natürlich.

Ergebnisse des Modellierungsvorganges sind nicht eindeutig!

Datenmodellierung - Schritte

Datenmodell für Prüfungsverwaltung:



Entity-Relationship-Modell

Veröffentlichung: Chen (1976).

Es ist durch seine

- graphische Darstellungsweise
- klare Definition

besonders benutzerfreundlich.

Standardelemente des ERM und deren Darstellung:

- Entity (Objektyp): Rechteck
- Relationship (Beziehungstyp): Raute
- Attribute: Ovale
- Beziehungsart

•

Beziehungsarten

- **1:1-Typ:**

einem Objekt des ersten Objektyps wird (null oder) ein Objekt des zweiten Objektyps zugeordnet, vice versa

- **1:n-Typ:**

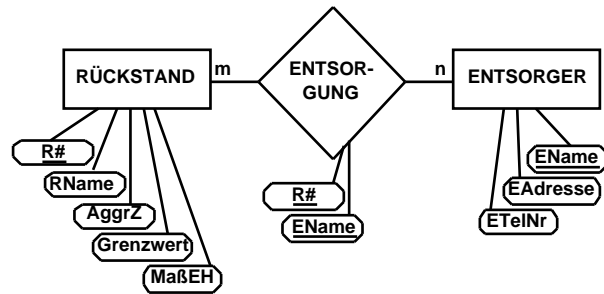
einem Objekt des ersten Objektyps können beliebig viele (auch null) Objekte des zweiten Objektyps zugeordnet werden, einem Objekt des zweiten Objektyps wird nur ein Objekt des ersten Objektyps zugeordnet

- **m:n-Typ:**

einem Objekt des ersten Objekt können n Objekte des zweiten Objektyps zugeordnet werden, vice versa.

(zwischen zwei Objektypen kann es mehrere Beziehungsarten geben)

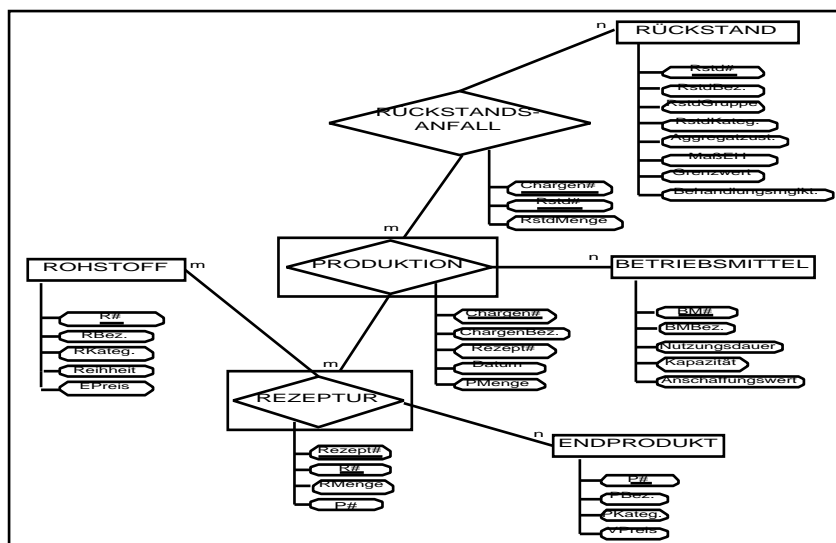
Beispiel zu ERM



m:n-Beziehungsart:

- ein bestimmter Rückstand kann von mehreren (n) Entsorgern entsorgt werden
- ein Entsorger kann mehrere (m) Rückstände entsorgen

Beispiel zu ERM



Weitere semantische Datenmodelle

Es wurde eine Reihe von Methoden zur semantischen Datenmodellierung entwickelt. Die bekanntesten sind:

- **strukturiertes ER-Modell (Sinz)**
- **Ansatz von James Martin**
- **Ansatz von Max Vetter**
- **NIAM (Nijssen)**
- **Objekttypenmethode (Ortner)**

Unternehmensweite Datenmodellierung

Ziel = fachliche Beschreibung aller im Unternehmen verwendeten Informationseinheiten (Objekttypen) und deren Beziehungen zueinander (Beziehungstypen) [ENDL92]

Vorgehensweise:

- **Top-Down**
- **Bottom-Up**

Unternehmensweite Datenmodellierung

Top-Down-Vorgehensweise:

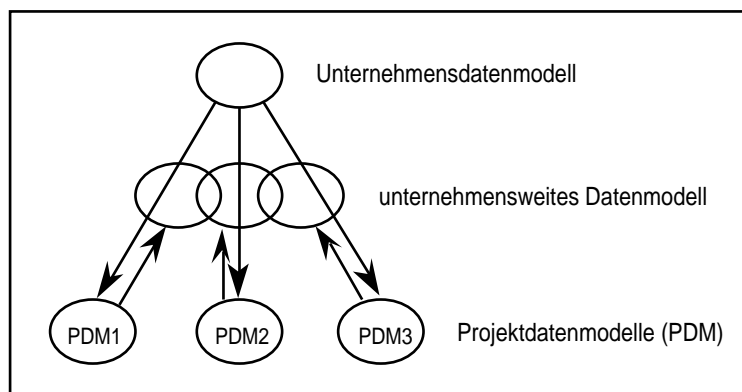
1. Erarbeiten des Unternehmensdatenmodells (= stark aggregiertes Datenmodell): ca. 70 Objekt- und 130 Beziehungstypen
2. Unternehmensdatenmodell durch projektbezogene Detaillierung soweit konkretisieren, bis alle für das Unternehmen relevanten Informationseinheiten und deren Beziehungen erfasst sind.

1. Für ein Projekt relevanter Ausschnitt der Datenarchitektur wird abgegrenzt.

2. Abgegrenzter Ausschnitt wird im Verlauf des Projekts verifiziert, korrigiert und konkretisiert (= Projektdatenmodell: PDM)
3. So entstandenes PDM wird in teilweise bestehendes unternehmensweites Datenmodell (uwDM) eingebettet.

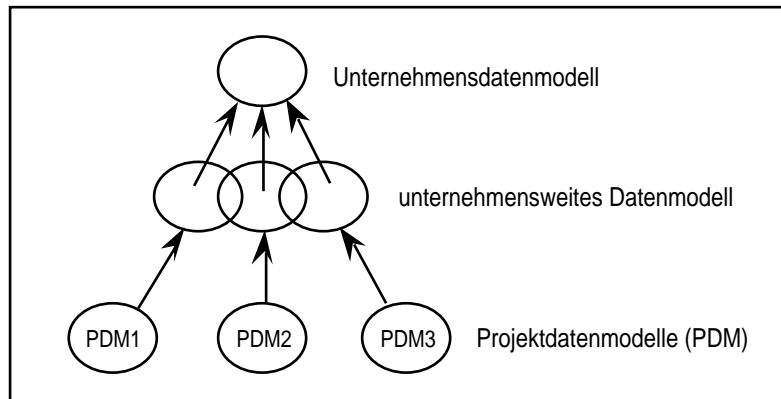
Unternehmensweite Datenmodellierung

Top-Down-Vorgehensweise:



Unternehmensweite Datenmodellierung

Bottom-Up-Vorgehensweise:



Unternehmensweite Datenmodellierung

Gründe [ENDL92]:

- Grundlage für präzise Datendefinition aus fachlicher und EDV-technischer Sicht
- klare, eindeutig definierte unternehmensweit gültige Begriffswelt
- Ermöglichen einer gezielten Abgrenzung von Projekten aus datenorientierter Sicht
- Schaffen einer Grundlage zur Erkennung von Unterschieden und Gemeinsamkeiten zu anderen Unternehmensteilen
- Vergleich und Bewertung von Standardsoftware
- Vermeidung von redundanten Daten - Datenstrukturen sollen möglichst anwendungsunabhängig entworfen werden

Unternehmensweite Datenmodellierung

Gründe:

- gute Datenbasis ist Grundlage für die Verwendung von sogenannten benutzerorientierten Abfrage- und Planungssprachen sowie Programmiersprachen der 4. Generation
- im Entwurf der Datenstrukturen kommt die informationelle Verflechtung verschiedener Anwendungsgebiete zum Ausdruck
- Eine sorgfältige Gestaltung der Datenbasis ist Grundlage eines Management-Informationssystems, das von den operativen Anwendungen ausgehend Daten über mehrere Verdichtungsstufen zur Unternehmensplanung zur Verfügung stellt.

Unternehmensweite Datenmodellierung

Fallstudie: Entwurf eines unternehmensweiten Datenmodells für den Schweizerischen Bankverein:

- **Initialisierungsworkshop: 2 Tage, 12 Vertreter aus In- und Ausland und allen Geschäftssparten**
 - > bedeutende Entitätsmengen wurden grob umrissen
- **Arbeitsgruppe: 7 Mitarbeitern, sechs mehrtägige Treffen,**
 - > 40-seitiges Dokument mit wichtigsten Entitätsmengen und Beziehungstypen, Aufwand der Arbeitsgruppe: 150 Mann-Tage!!!
 - Aufwand für delegierte Aufgaben: ein vielfaches davon!!!
 - Verabschiedung des globalen Datenmodells im Grobentwurf: nach einem Jahr
- **Verfeinerung des Grobentwurfs: Aufwand um ein Vielfaches höher als in den vorangegangenen Stufen!!!**
- **Analyse der bestehenden Datenstrukturen und deren Semantik**

LOGISCHER DATENBANKENTWURF

Klassische (logische) Datenmodelle:

1. graphenorientierte Modelle:

- Hierarchisches Modell
- Netzwerkmodell

2. tabellenorientierte Modelle:

- Relationales Modell

–

Das verwendete Datenmodell beeinflusst natürlich die Datenmodellierung.

Hierarchisches Modell

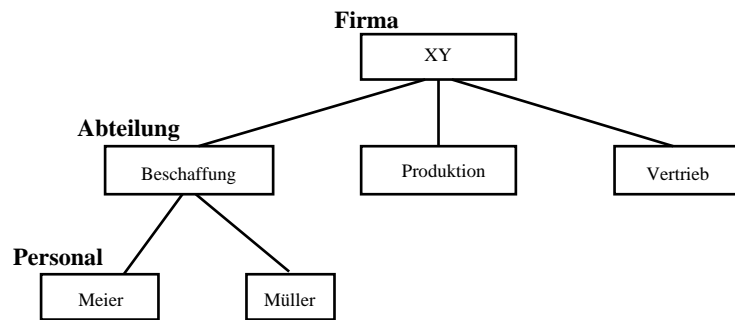
Das hierarchische Modell enthält:

- eine Menge von Objekttypen
- eine Menge von 1:n-Beziehungstypen

Daraus folgt:

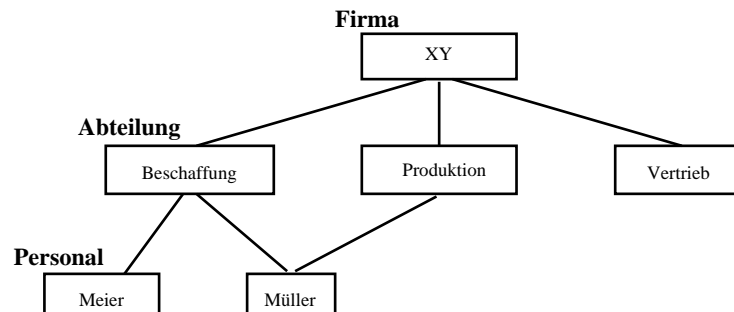
- Zyklen sind nicht erlaubt
- jeder untergeordnete Objekttyp darf höchstens einen übergeordneten Objekttyp haben, es gibt einen oder mehrere Objekttypen ohne Vorgänger (Wurzeltyp)
-

Hierarchisches Modell



Netzwerkmodell

Das Netzwerkmodell läßt die strenge Forderung nach Baumhierarchien fallen. Dadurch können in der Praxis häufig anzutreffende m:n-Beziehungen modelliert werden.



Graphenorientierte Modelle

In einfachen graphenorientierten Modellen erfolgt der Zugriff zu den einzelnen Datenelementen (Datensätzen) durch Navigation.

Die Suche gestaltet sich so, daß man ausgehend von einem bestimmten Startobjekt über diverse Zwischen-objekte schließlich zum gesuchten Zielobjekt gelangt.

Diese Zugriffsmethode setzt voraus, daß

- mögliche Zugriffswege zwischen den Objekten bekannt sein müssen
- die Einstiegspunkte zur Suche genau vordefiniert sind.
-

Graphenorientierte Modelle

Nachteil hierarchisches / Netzwerkmodell:

- Beziehungen werden explizit durch spezielle interne Datentypen (Pointer) dargestellt ==> Anforderungen der Anwendung an Daten müssen von Anfang an bekannt sein
- genaue Angaben über Navigation zu den Daten, prozedurale Verarbeitung [MATH87]
- satzorientierte Verarbeitung

RELATIONENMODELL

Das Relationenmodell wurde 1970 von Codd vorgestellt.

Eigenschaften:

- Daten werden in Tabellenform dargestellt (nicht in Form eines Graphen)
- es wird keine Unterscheidung zwischen Objekten und Beziehungen gemacht

Folgende Datenbanksysteme sind unter anderem relational:
DB2, Oracle, Ingres, Informix, Access, ...

RELATIONENMODELL

Nomenklatur:

- **Attribut:**
Ein Attribut ist ein Merkmal eines Objekttyps. Ein Attribut bezeichnet somit eine Spalte einer Tabelle.
- **Wertebereich:**
Dieser spezifiziert alle möglichen Merkmalsausprägungen eines bestimmten Attributes.
- **Tupel:**
Die Merkmalsausprägungen eines Objekts bezeichnet man als Tupel. Ein Tupel ist somit eine Zeile der Tabelle.
- **Schlüssel:**
Jene Teilmenge der Attribute, die eine Tabelle eindeutig identifiziert, wird als Schlüssel bezeichnet.

RELATIONENMODELL

Nomenklatur:

- **Relation (Tabelle):**

Eine Relation wird als Menge von Tupel definiert.

Aus der Mengendefinition folgt, daß

- die Reihenfolge der Tupel unwesentlich ist und
- daß es keine zwei gleichen Tupel geben darf.

- **Relationenname**

- **Relationenschema (Tabellengerüst):**

Dieses umfaßt Relations- und Attributnamen. Ein Relationenschema stellt somit die Struktur einer Relation dar.

- **Relationales Datenbankschema:**

Dieses setzt sich aus den einzelnen Relationenschemata zusammen.

RELATIONENMODELL

The diagram shows a table with the following structure:

RÜCKSTAND	RNr	RName	AggrZuStd	Grenzwert	MaßEH
1	CO2	gasförmig	50	cm3	
2	SO2	gasförmig	15	cm3	
3	Nitrat	fest	50	mg	

Labels and arrows in the diagram:

- Tabellenname**: points to the header row.
- Schlüssel**: points to the first column (Rückstand).
- Attribute**: points to the columns RName, AggrZuStd, Grenzwert, and MaßEH.
- 1 Tupel**: points to the first data row.

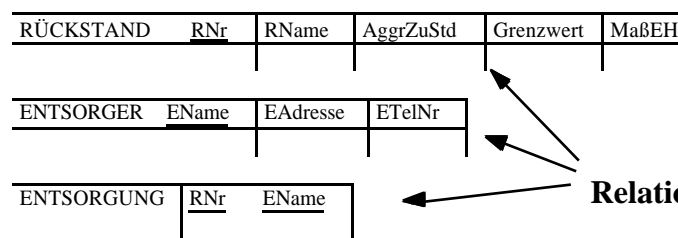
Wertebereich für AggrZuStd: fest, flüssig, gasförmig

RELATIONENMODELL

Tabelle (Relation):

1	CO2	gasförmig	50	cm3
2	SO2	gasförmig	15	cm3
3	Nitrat	fest	50	mg

Relationales Datenbankschema:



Operationen auf Relationen

Es gibt folgende Operationen:

- Selektion
- Projektion
- Verbund
- Mengenoperationen
 - Vereinigung
 - Durchschnitt
 - Mengendifferenz

Operationen auf Relationen

Selektion:

Die Selektion wählt alle jene Tupel aus einer Relation aus, die eine bestimmte Bedingung erfüllen.

Eigenschaften:

- Anwendung auf eine Relation
- Relationenschema (Anzahl der Attribute) bleibt gleich
- Zahl der Tupel kann reduziert werden

–

Operationen auf Relationen

Beispiel:

Es sollen alle Tupel der Relation RÜCKSTAND selektiert werden, die den Aggregatzustand gasförmig haben:

Ergebnis:

1	CO2	gasförmig	50	cm3
2	SO2	gasförmig	15	cm3

Operationen auf Relationen

Projektion:

Die Projektion ergibt als Ergebnis eine Relation mit denjenigen Attributwerten, auf deren Attribute die Projektion angewandt wurde.

Eigenschaften:

- Anwendung auf eine Relation
- Relationenschema (Anzahl der Attribute) kann sich verringern
- Zahl der Tupel kann reduziert werden, falls es in der verbleibenden Relation identische Tupel gibt

Operationen auf Relationen

Beispiel:

Auf obige Ergebnisrelation soll eine Projektion auf die Attribute RName, Grenzwert und MaßEH durchgeführt werden.

Ergebnis:

CO2	gasförmig	cm3
SO2	gasförmig	cm3

Operationen auf Relationen

Verbund:

Bei der Verbundoperation werden zwei Relationen mit einem gemeinsamen Attribut zu einer neuen Relation verbunden. Das neue Relationsschema enthält alle Attribute der beiden Relationen.

Die Ergebnisrelation enthält all jene Tupel, für die das "Verbundattribut" identisch ist.

Eigenschaften:

- Anwendung auf mehrere Relationen
- Anzahl der Attribute kann sich vergrößern
- Zahl der Tupel kann sowohl zu- als auch abnehmen

Operationen auf Relationen

Verbund:

Zusammenhang zwischen Verbund-Operation im Relationenmodell und Beziehungstyp im Entity-Relationship-Modell:

Im relationalen Modell ist ein Verbund zwischen zwei Relationen in der Regel nur dann möglich, wenn im ERM zwischen diesen beiden Objekttypen ein Beziehungstyp existiert.

Operationen auf Relationen

Beispiel:

ENTSORGUNG	1	ASA
	1	MüGu
	3	Saubermacher
	3	ASA
	3	MüGu

Ergebnis: RÜCKSTAND join ENTSORGUNG (projiziert auf die Attribute RNr, RName, EName):

1	CO2	ASA
1	CO2	MüGu
3	Nitrat	Saubermacher
3	Nitrat	ASA
3	Nitrat	MüGu

Beachten Sie:

Es müssen nicht alle Tupel der an einer Verbundoperation teilnehmenden Tabellen in der Ergebnisrelation enthalten sein. In obigen Beispiel ist das Tupel <2, SO2, ...> nicht in der Ergebnisrelation enthalten, weil es für die Rückstandnummer 2 keine Übereinstimmung in der Tabelle ENTSORGUNG gibt.

Operationen auf Relationen

Mengenoperationen:

Mengenoperationen werden auf zwei (oder mehrere) Relationen mit gleicher Attributmenge angewandt. Das Ergebnis ist wieder eine Relation mit gleicher Attributmenge.

Lediglich die Anzahl der Tupel verändert sich.

- **Vereinigung:**

Die Ergebnisrelation enthält alle Tupel beider Relationen.

- **Durchschnitt:**

Die Ergebnisrelation enthält nur jene Tupel, die sowohl in der ersten als auch in der zweiten Relation vorkommen.

- **Differenz:**

Die Operation $R1 \setminus R2$ ergibt als Ergebnis eine Relation, die alle jene Tupel der Relation R1 enthält, die nicht in der Relation R2 vorkommen.

Operationen auf Relationen

Beispiel:

RÜCKSTAND U <4, Jauche, flüssig, 10, cl>

Ergebnis:

1	CO2	gasförmig	50	cm3
2	SO2	gasförmig	15	cm3
3	Nitrat	fest	50	mg
4	Jauche	flüssig	10	cl

Operationen auf Relationen

Beispiel:

RÜCKSTAND \ <1, CO2, gasförmig, 50, cm3>

Ergebnis:

2	SO2	gasförmig	15	cm3
3	Nitrat	fest	50	mg
4	Jauche	flüssig	10	cl

Operationen auf Relationen

Interpretationen der Operationen:

- **Selektion:** Abfrage
- **Verbund:** Verbinden von Informationen aus verschiedenen Relationen
- **Projektion:** wunschgerechte Aufbereitung von Informationen, z. B. einige Attribute sind bei einer bestimmten Abfrage nicht relevant
- **Vereinigung:** Einfügen
- **Differenz:** Löschen

•

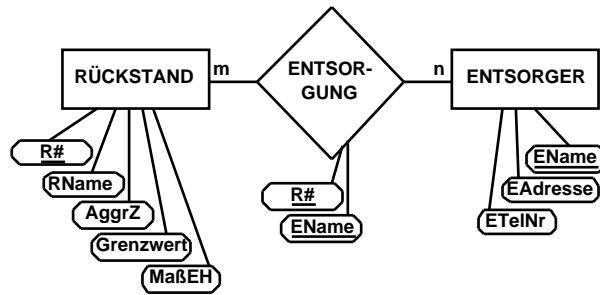
ER-Modell --> Relationenmodell

Bei der Umsetzung eines ER-Modells in ein Relationenmodell ist folgendes zu beachten:

1. Objekttypen werden automatisch in Relationen überführt
2. Nur m:n-Beziehungstypen werden zu eigenen Relationen.
3. Bei 1:n-Beziehungstypen werden die Attribute des Beziehungstyps beim "n-Objekttyp" untergebracht.
4. Bei 1:1-Beziehungstypen können die beiden daran beteiligten Objekttypen in der Regel zu einer Relation zusammengefaßt werden.

ER-Modell --> Relationenmodell

Beispiel:



ER-Modell --> Relationenmodell

Es ergeben sich folgende drei Relationen:

RÜCKSTAND

R#
RName
AggrZ
Grenzwert
MaßEH

ENTSORGER

ENName
EVName
EStraße
EPlz
EOrt
ETelNr

ENTSORGUNG

R#
ENName

ER-Modell --> Relationenmodell

Speziell dann, wenn ein ER-Modell in einem geringeren Detailliertheitsgrad erstellt wurde, enthält es Anomalien, weil:

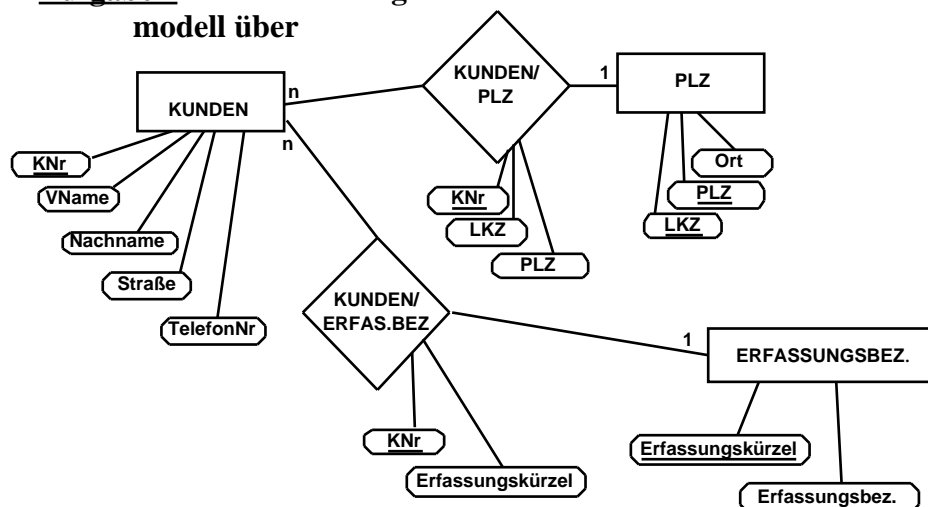
1. die Relationen Redundanzen enthalten
2. die Relationen einen schlechten Aufbau haben.

(Letztes war der Grund dafür, daß "EName" in "EVName" und "ENName", sowie "EAdresse" in "EStraße", "EPlz" und "EOrt" aufgespalten wurden. Die Relation "ENTSORGER" enthält aber immer noch Redundanzen ("EPlz" und "EOrt").)

Aus diesem Grund müssen die Relationen, bevor sie auf ein konkretes Datenbanksystem überführt werden können, "normalerweise" noch normalisiert werden.

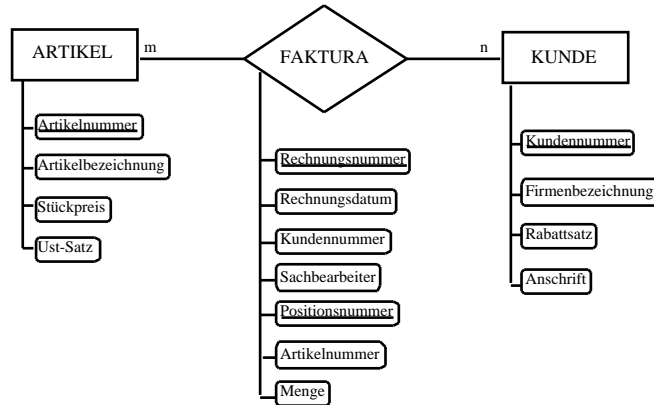
ER-Modell --> Relationenmodell

Aufgabe 1: Führen Sie folgendes ER-Modell in ein Relationenmodell über



ER-Modell --> Relationenmodell

Aufgabe 2: Führen Sie folgendes ER-Modell in ein Relationenmodell über



Normalformen

Intuitiv aufgestellte Relationen enthalten folgende Anomalien:

- Speicheranomalie
- Löschanomalie
- Änderungsanomalie

Normalformen

Tatbestand:

**Jeder Mitarbeiter ist genau einer Abteilung zugeordnet
Ein Mitarbeiter kann mehrere Aufgabenbereiche haben**

AUFG./MA	Aufgabenbereich	Mitarbeiter	Abteilung
	Terminierung	Meier	Produktion1
	Reparaturen	Meier	Produktion1
	EDV	Müller	Controlling
	Bestellwesen	Kunz	Beschaffung

Normalformen

Anomalien:

• **Speicheranomalie:**

Man möchte speichern, daß Herr Berger neuer Mitarbeiter der Abteilung Beschaffung ist. Nachdem Herrn Berger aber noch kein Aufgabengebiet zugeordnet wurde, muß man mit der Speicherung zuwarten.

• **Löschanomalie:**

Herr Kunz gibt seinen Aufgabenbereich Bestellwesen ab, bleibt aber nachwievor im Unternehmen und somit der Abteilung Beschaffung zugeordnet. Würde man nun das vierte Tupel löschen, so geht auch die Information verloren, daß Herr Kunz Mitarbeiter der Abteilung Beschaffung ist.

Normalformen

- **Änderungsanomalie:**

Wechselt nun Herr Meier von der Abteilung Produktion1 in die Abteilung Produktion3, so muß man alle Tupel suchen und ändern, in denen Herr Meier vorkommt.

1. Normalform

Derartige Anomalien können durch einen geeigneten Entwurf von Relationenschemata umgangen werden. Es wurden sogenannte Normalformen definiert, das sind Einschränkungen bestimmter Art, denen die Relationen genügen müssen, um solche Anomalien zu vermeiden.

1. Normalform

Ein Relationenschema ist in erster Normalform, wenn

- die Attribute nicht mehr weiter zerlegbar (= atomar) sind
- es keine Wiederholungsgruppen gibt

1. Normalform

Atomare Attribute:

ENTSORGER	EName	EAdresse	ETelNr
	Saubermacher Alois	Herrengasse 10, 8010 Graz	0316/3831
	Fa. ASA	1140 Wien, Linzerstraße 23	0222/238234
	Werner MüGu	Amstetten, 3300, Hauptplatz 17	07472/2010

Probleme:

- alphabetische Liste der Entsorger
- Adreßetiketten drucken

==>

ENTS.	ENName	EVName	EStraße	EPlz	EOrt	ETelNr
	Saubermacher	Alois	Herrengasse 10	8010	Graz	0316/3831
	ASA		Linzerstraße 23	1140	Wien	0222/238234
	MüGu	Werner	Hauptplatz 17	3300	Amstetten	07472/2010

1. Normalform

Wiederholdungsgruppen

PERSON	SozVNr	NName	Kind1	Kind2	Kind3	Kind4	Kind5
	2930191261	Tischler	Günther	Klaus			
	4711100165	Metz	Helmut	Ingo	Dieter	Boris	Helga
	5812012063	Urschler	Gerald				

Probleme:

- Speicherplatzverschwendung
- nachtragliche anderungen der Dateistruktur konnen nicht ausgeschlossen werden
- schwierige Datenabfrage

1. Normalform

==>

PERSON	SozVNr	NName	Kind
	2930191261	Tischler	Günther
	2930191261	Tischler	Klaus
	4711100165	Metz	Helmut
	4711100165	Metz	Ingo
	4711100165	Metz	Dieter
	4711100165	Metz	Boris
	4711100165	Metz	Helga
	5812012063	Urschler	Gerald

2. Normalform

$f: M \rightarrow N : f \text{ } M \times N \text{ } \& \text{ } 'x,y,y' (x,y) \text{ } f \text{ } \& \text{ } (x,y) \text{ } f$
 $y=y'$

x	y
1	1
2	4
3	9

x	y
1	1
2	1
3	1

x	y
1	Meier
2	Huber
3	Müller

x	y
1	1
1	-1
2	4
2	-4

x	y
1	Meier
2	Huber
2	Muster
3	Müller

Funktion?

2. Normalform

Funktionale Abhängigkeit:

Eine Attributmeng X bestimmt die Attributmeng Y funktional ($X \rightarrow Y$, Y ist von X funktional abhängig), genau dann, wenn es für jeden X -Wert genau einen Y -Wert gibt.

Volle funktionale Abhängigkeit:

Ein Attribut A ist von einer Attributmeng X voll funktional abhängig, genau dann, wenn es keine Teilmenge von X gibt, die A funktional bestimmt.

2. Normalform

ENTSORGER

ENName
EVName
EStraße
EPlz
EOrt
ETelNr

Funktionale Abhängigkeiten:

ENName EVName
ENName EStraße
ENName EPlz
ENName EOrt
ENName ETelNr
EPlz EOrt
ETelNr ENName
ETelNr EVName
ETelNr EStraße
ETelNr EPlz
ETelNr EOrt

2. Normalform

PERSONAL

SozVNr

NName

Kind

Funktionale Abhängigkeiten:

SozVNr → NName

Keine funktionale Abhängigkeiten:

SozVNr → Kind

Begründung:

Eine Person kann mehrere Kinder haben. In diesem Fall würden einer Sozialversicherungsnummer mehrere Kindervornamen zugeordnet.

NName → SozVNr

Begründung:

In der Relation PERSON könnte es mehrere Personen mit demselben Nachnamen geben. In diesem Fall würden aber einem bestimmten Nachnamen (z. B. Meier) mehrere Sozialversicherungsnummern zugewiesen!

2. Normalform

Nunmehr ergibt sich eine Erweiterung des Begriffes des Relationenschemas:

Ein Relationenschema setzt sich zusammen aus:

- Namen der Relation
- Attributen
- funktionalen Abhängigkeiten

Schlüssel:

Ein *Schlüssel* ist jene minimale Attributmenge, die jedes Tupel einer Relation eindeutig bestimmt.

Ein *Teilschlüssel* ist eine Teilmenge des Schlüssels.

Ein *Nichtschlüsselattribut* ist ein Attribut, das kein Teilschlüsselattribut ist.

2. Normalform

Ein Relationenschema ist in 2. Normalform, wenn es in erster Normalform ist, und zusätzlich gilt, daß jedes Nichtschlüsselattribut vom Schlüssel voll funktional abhängig ist.

Wenn ein Schlüssel nur aus einem Attribut besteht, dann ist es in 2. Normalform, wenn jedes Nichtschlüsselattribut vom Schlüssel funktional abhängig ist.

2. Normalform

Beispiel :

ENTSORGER

ENName

EVName

EStraße

EPlz

EOrt

ETelNr

Schlüssel: ENName

Jedes Nichtschlüsselattribut vom Schlüssel voll funktional abhängig:

ENName EVName

ENName EStraße

ENName EPlz

ENName EOrt

ENName ETelNr

==> ENTORGER in 2. Normalform

2. Normalform

Beispiel :

PERSON
SozVNr
 NName
Kind

Schlüssel: SozVNr + Kind

Das Nichtschlüsselattribut NName nur von einem Teil des Schlüssels funktional abhängig:
 SozVNr NName

==> PERSON nicht in 2. Normalform

==> Zerlegung (funktionale Abhängigkeiten müssen erhalten bleiben):

KINDER PERSON
SozVNr SozVNr
Kind NName

2. Normalform

Beispiel :

REL	Vorlesung	Lektor	Semester	Institut
	Progr.	Sorgenfrei	WS89/90	Informatik
	Ökonom.	Anders	WS89/90	Statistik
	Progr.	Sorgenfrei	SS90	Informatik
	Ökonom.	Danner	WS89/90	Statistik

Funktionale Abhängigkeiten:
 {Lektor} -> {Institut}
 {Lektor, Semester} -> {Vorlesung}

Schlüssel: {Lektor, Semester}

==> REL ist nicht in 2. Normalform und muß unter Wahrung der funktionalen Abhängigkeiten aufgespalten werden:

REL1	Vorlesung	Lektor	Semester
	Progr.	Sorgenfrei	WS89/90
	Ökonom.	Anders	WS89/90
	Progr.	Sorgenfrei	SS90
	Ökonom.	Danner	WS89/90

REL2	Lektor	Institut
	Sorgenfrei	Informatik
	Anders	Statistik
	Danner	Statistik

3. Normalform

Transitive Abhängigkeit:

Eine funktionale Abhängigkeit $X \rightarrow Z$ ist eine transitive Abhängigkeit, wenn es zwei funktionale Abhängigkeiten $X \rightarrow Y$ und $Y \rightarrow Z$ gibt (und weiters gilt: X ist ungleich Y , Z ist keine Teilmenge von Y und X ist nicht funktional von Y abhängig).

Ein Relationenschema ist in 3. Normalform, wenn es in 2. Normalform ist und zusätzlich gilt, daß kein Nichtschlüsselattribut von irgend einem Schlüssel transitiv abhängig ist.

3. Normalform

Ein Relationenschema mit einer transitiven Abhängigkeit ($A \rightarrow B \ \& \ B \rightarrow C \implies A \rightarrow C$) muß folgendermaßen aufgespalten werden:

REL1 | A B REL2 | B C

Beispiel :

ENTSORGER

ENName

EVName

EStraße

EPlz

EOrt

ETelNr

Befindet sich das Relationenschema ENTSORGER in 3. Normalform? Wenn nein, dann führen Sie die entsprechenden Zerlegungen durch.

3. Normalform

Beispiel :

ENName EVName
ENName EStraße
ENName EPlz
ENName EOrt
ENName ETelNr
EPlz EOrt

Es gibt eine transitive Abhängigkeit:

ENName EPlz & EPlz EOrt \implies ENName EOrt

Das Nichtschlüsselattribut EOrt ist transitiv vom Schlüssel ENName abhängig.

\implies

ENTSORGER ORT
ENName EPlz
EVName EOrt
EStraße
EPlz
ETelNr

3. Normalform

Algorithmus zum Zerlegen in 3. Normalform:

1. Alle Abhängigkeiten f mit identischer linker Seite werden zusammengefaßt: $(X \rightarrow A1, X \rightarrow A2, \dots, X \rightarrow An) \implies (X \rightarrow A1, A2, \dots, An)$.
2. Die Attribute der einzelnen Abhängigkeiten formen die Attributmengen der neuen Relationenschemata: $RS1((X, A1, A2, \dots, An), (X \rightarrow A1, A2, \dots, An))$
3. Wenn in einem Relationenschema alle Schlüsselattribute vorkommen, so ist das relationale Datenbankschema eine verbundtreue Zerlegung des Relationenschemas.

3. Normalform

(Verbundtreu heißt, daß die ursprüngliche Information durch die Verbundoperation zwischen den zerlegten Teilrelationen wiederhergestellt werden kann.)

Anderenfalls muß ein zusätzliches Relationenschema eingefügt werden, das aus allen Schlüsselattributen besteht.

Dieser Algorithmus löst nicht alle in der Realität auftretenden Probleme.

Er kann jedoch in den meisten Fällen eingesetzt werden.

Denormalisierung

Gegeben seien folgende zwei Relationen (Dateien) - 3. Normalform:

Name	Straße	Plz	
Tischer	Josefiberg 5	3363	
Gatterbauer	Hausmeninger Str.	1103300	
Stiebellehner	Kühberg	3370	
...	

Plz	Ort
3300	Amstetten
3363	Hausmening
3370	Seitenstellen

Annahmen:

1. Sehr wenige neu hinzukommende Daten
2. Adresse muß sehr oft gedruckt werden, Zugriffszeit soll daher minimal sein.

Für das Drucken einer Adresse muß auf zwei Dateien zugegriffen werden, diese müssen miteinander verbunden werden (Verbundoperation).

Denormalisierung

Aus Effizienzüberlegungen werden die beiden Dateien zu einer Datei zusammengefaßt. Die daraus resultierende Datei ist nicht mehr in 3. Normalform!!!

Dafür muß man nur mehr auf eine Datei zugreifen, um eine Adresse zu drucken.

Name	Straße	Plz	Ort		
Tischer	Josefberg 5	3363	Hausmening		
Gatterbauer	Hausmeninger Str.	1103300	Amstetten		
Stiebellehner	Kühberg	3370	Seitenstetten		
...		

Denormalisierung

Schritte beim Datenbankentwurf:

1. Datenmodellierung
2. Übertragung des Entwurfsergebnisses auf ein bestimmtes Datenbanksystem
3. Anpassung der Datenbankstruktur
 - A. Ergänzung abgeleiteter Datenstrukturen
 - B. Denormalisierung (i. e. S.).

Denormalisierung

Gründe für Optimierung der Datenbankstruktur:

1. technische und organisatorische Anforderungen werden nicht erfüllt
 - Schnittstellen zu bestehenden Systemen
 - Datenschutz
 - etc.
2. Leistungsverhalten der Datenbank reicht nicht aus (siehe voriges Beispiel):
 - Antwortzeiten
 - Rechenzeit
 - Speicher
 - etc.

Denormalisierung

A. Abgeleitete Datenstrukturen:

Abgeleitete Datenstrukturen sind Datenstrukturen, die auf in der Informationsstruktur enthaltene Informationen zurückgeführt werden können:

Kopien

- verdichtete Informationen
- unterschiedlich zusammengestellte Informationen: z. B. Berechnungen
- zusätzliche Zugriffstrukturen: Indizes, Verkettungen

Abgeleitete Datenstrukturen führen zu einer Ergänzung der Datenbankstruktur um die benötigten Satztypen, Beziehungen und Datenelemente.

Denormalisierung

Verdichtete Informationen:

KSt-Nr	Art-Nr	Kosten	
1	171	0000	
1	291	3000	
2	102	5000	
3	331	3400	

KSt-Nr	Kosten
1	23000
2	13000
3	13400

Unterschiedlich zusammengestellte Informationen:

Art-Nr	NettoVkB	BruttoVkB	
10100	120		
11200	240		
121000	1200		

Denormalisierung

Gründe für die Aufnahme von abgeleiteten Datenstrukturen:

- Fachbegriffe, die auf ein oder mehrere Datenelemente zurückgeführt werden können: z. B. Umsatz = Menge * Wert
- Managementinformationen - verdichtete Informationen
- Kopien bei Datensicherung, verteilten Datenbanken, etc.
- Vereinfachung des Programmablaufs, Berücksichtigung von Schnittstellen zu existierenden Anwendungssystemen, Datenschutz-Gründe (z. B. Views)

Denormalisierung

B. Denormalisierung (i. e. S.)

Bei der Denormalisierung werden keine neuen Datenstrukturen hinzugefügt, sondern die Struktur der vorhandenen Datenstrukturen wird verändert.

Grund:

häufige bzw. kritische Zugriffe auf die Datenbank werden vorweggenommen und statisch implementiert; dadurch wird die Zahl der Zugriffe reduziert.

Zwei Möglichkeiten:

1. Zusammenfassung von Relationen (Dateien): siehe voriges Beispiel
2. Zerlegung von Relationen
 - nach Attributen (Vorwegname der Projektion)
 - nach Tupeln (Vorwegnahme der Selektion)

Denormalisierung

Zerlegung nach Attributen:

Für sehr viele Anwendungen wird die Straße einer Person nicht benötigt:

Name Plz	Name Straße
Tischer 3363	Tischer Josefiberg 5
Gatterbauer3300	GatterbauerHausmeninger Str. 110
Stiebellehner3370	StiebellehnerKühberg
...

Denormalisierung

Zerlegung nach Tupeln:

Sehr oft wird nur auf Personen zugegriffen, die in Amstetten wohnen (Plzl = 3300)

Name	Straße	Plzl	Ort		
Gatterbauer	Hausmeninger Str.	1103300	Amstetten		
...	...	3300	Amstetten		
...		

Name	Straße	Plzl	Ort		
Tischer	Josefberg 5	3363	Hausmening		
Stiebellehner	Kühberg	3370	Seitenstetten		
...		

Denormalisierung

Empfehlungen:

- Die beste Denormalisierung ist keine Denormalisierung.
- Zuerst Normalisieren, erst dann eventuell denormalisieren.
- Die Einführung abgeleiteter Datenstrukturen ist besser als denormalisieren.
- Stabile Datenstrukturen (wenig Änderungen) sind bei der Denormalisierung zu bevorzugen.

Wann ist ein Datenbanksystem relational?

Ein Datenmodell besteht im wesentlichen aus drei Komponenten:

- 1. Menge der logischen Datenstrukturen (Strukturkomponente)**
- 2. Menge von Operatoren (Manipulationskomponente)**
- 3. Menge von statischen und dynamischen Integritätsbedingungen (Integritätskomponente)**

Wann ist ein Datenbanksystem relational?

Strukturkomponente:

Im relationalen Datenmodell gibt es dafür:

- Attribute,**
- Schlüsselkandidaten,**
- Primärschlüssel**
- Wertebereiche**
- Relationen**

Manipulationskomponente:

- Selektion,**
- Verbund,**
- Projektion,**
- Mengenoperationen**

Wann ist ein Datenbanksystem relational?

Integritätskomponente:

Es handelt sich dabei um eine Menge von semantischen Einschränkungen.

- Entitätsintegrität: ein Primärschlüssel darf in einer Relation niemals unbekannt sein.
- Verweisintegrität: wenn auf ein Objekt in der Datenbank Bezug genommen wird, dann muß dieses Objekt in der Datenbank existieren.

Wann ist ein Datenbanksystem relational?

Damit ein Datenbanksystem relational ist, müssen folgende zwei Bedingungen erfüllt sein (nach CODD):

1. Das Datenbanksystem muß Tabellensichten unterstützen, ohne daß sichtbare Navigationen (vergleiche graphenorientierte Datenmodelle) durchgeführt werden müssen.
2. Es muß eine Datenmanipulationssprache existieren, die minimale Operationen erfüllt (Projektion, Selektion, Verbund).

Ist diese Bedingung nicht erfüllt, dann ist das Datenbanksystem nur tabellenorientiert.

Wann ist ein Datenbanksystem relational?

Ein Datenbanksystem ist relational vollständig wenn folgende Bedingungen erfüllt sind:

1. Das Datenbanksystem muß relational sein.
2. Die gesamte relationale Algebra wird unterstützt.

Ein Datenbanksystem ist voll relational, wenn folgende Bedingungen erfüllt sind:

1. Das Datenbanksystem ist relational vollständig.
2. Es werden mindestens Entitäts- und Verweisintegrität unterstützt.
- (3. Die Behandlung von unbekanntem Werten (der Wert für ein bestimmtes Attribut ist nicht bekannt) wird von der relationalen Algebra unterstützt.)

DATENBANKSPRACHEN

Aus funktionalen Gründen unterscheidet man zwischen

- Datenbeschreibungssprachen (DDL) und
- Datenmanipulationssprachen (DML)

wobei beide Funktionen in der Regel in einer Sprache integriert sind

Datenbeschreibungssprachen:

helfen dem Datenbankadministrator, die Struktur einer Datenbank, ihre zulässigen Zustände, die notwendigen Sichten und die Zugriffsberechtigungen zu beschreiben.

Datenbanksprachen

Datenmanipulationssprachen:

- Veränderungen des Datenbankinhaltes durch die Operationen Einfügen, Löschen und Ändern (insert, delete, update)
- Abfragen (queries) des Inhalts der Datenbank auf der Basis eines externen Modells und Bereitstellung der Ergebnisse für eventuelle weitere Verarbeitung
-
-

Datenbanksprachen

Unterscheidungsmöglichkeiten von Datenmanipulations-sprachen:

1. Nach dem Umfang der Sprache:

- Veränderungen und Abfragen möglich
- nur Abfragen möglich: reine Abfragesprachen bezeichnet man auch als "query languages"

Bei Abfragen gibt es zwei Gruppen:

- Ja/nein-Abfrage
- Abfrage einer Menge von Datensätzen

Datenbanksprachen

2. Nach der Mächtigkeit der Operationen:

- mengenorientierte Sprachen: im Relationenmodell sind die Operanden Tabellen (Menge von Tupeln)
 - tupelorientierte Sprachen: im hierarchischen Modell und im Netzwerkmodell kann hingegen in einer Operation nur ein Tupel als Operand auftreten
-

Datenbanksprachen

3. Nach der Art der Spezifizierung der gewünschten Informationen:

- prozedurale Sprachen: Es muß angegeben werden, "WIE" die gewünschte Information ermittelt werden soll. Es muß also die Operationsfolge, mit der die Ergebnisrelation konstruiert werden soll, spezifiziert werden.
 - deskriptive Sprachen: Es wird das "WAS" angegeben, das gesucht werden soll. Die Transformation vom "WAS" in das "WIE" wird von der Datenbanksprache vorgenommen. Dieser Sprachtyp bringt für den Benutzer natürlich wesentliche Erleichterungen.
-

Zu jedem dieser Ansätze gibt es verschiedene Sprachvarianten. Der Übergang zwischen den beiden Sprachtypen ist fließend.

Datenbanksprachen

Selbständigkeit einer Datenbanksprache:

- **Einbettung in eine höhere Programmiersprache:** Historisch gesehen ist das die ältere Methode (z. B.: DL/1 ist in PL/I eingebettet)
- **eigene Programmiersprache:** (z. B.: DBASE)
- **Mischformen:** Datenbanksprache kann auch alleine existieren (z. B.: SQL)

SQL

Charakteristik:

- selbständig oder eingebettet verfügbar
- deskriptiv
- mengenorientiert
- relational
- "Standard"

Operationen:

- Datenabfrage
- Datenmanipulation
- Integritätsprüfung
- Zugriffskontrolle

SQL

Musterbeispiel:

3 Relationen: P Personal
P-MPersonal-Maschinenausbildung
MMaschinen

P	P-Nr	Name	Abt-Nr	Lohn
	300	Meier	12	12.000
	301	Huber	10	14.000
	302	Müller	7	10.900
	303	Bauer	5	13.000
	304	Schmid	12	17.000

P-M	P-Nr	M-Nr
	301	4
	301	3
	303	1
	304	4
	300	2
	300	1

SQL

Datendefinition:

```
CREATE TABLE P (P-Nr DECIMAL (3) NOT NULL,  
                Name CHAR (20),  
                Abt-Nr DECIMAL (2),  
                Lohn DECIMAL (6),  
                UNIQUE (P-Nr) )
```

Datenmanipulation:

- SELECT
- INSERT
- UPDATE
- DELETE

SQL

Eine **Abfrage** hat in SQL folgende Struktur:

```
SELECT <Attributliste>  
FROM <Relation(en)>  
WHERE <Bedingung>
```

Im **SELECT**-Teil werden die Attribute angegeben, die in der Ergebnisrelation enthalten sein sollen.

Im **FROM**-Teil werden alle Relationen spezifiziert, aus denen die gewünschten Ergebnisse gesucht werden sollen.

Im **WHERE**-Teil wird die Bedingung angegeben, unter der die Tupel aus den Relationen ausgewählt werden sollen.

SQL

Bei einer Bedingung sind folgende Vergleichsoperatoren möglich:

- =
- <> (ungleich)
- >
- >=
- <
- <=
- between ... and ...

SQL

Im WHERE-Teil können auch mehrere Bedingungen angeführt werden, die mittels logischer Operatoren miteinander verknüpft werden.

Folgende logische Operatoren sind möglich:

- AND
- OR
- IN
- NOT
-

SQL

Projektion und Selektion:

Es sollen Name und Lohn aller Mitarbeiter, die in der Abteilung Nummer 12 arbeiten, ausgegeben werden:

```
SELECT Name, Lohn
FROM P
WHERE Abt-Nr=12
```

Es sollen Personalnummer, Lohn und Name aller Mitarbeiter, die in Abteilung 12 arbeiten und weniger als 15.000 S verdienen, ausgegeben werden. Die Ergebnisrelation soll absteigend nach Lohn sortiert sein:

```
SELECT P-Nr, Name, Lohn
FROM P
WHERE Abt-Nr=12 AND
      Lohn<15.000
ORDER BY Lohn DESC
```

SQL

Eingebaute Funktionen:

Count, Sum, Avg, Max, Min, Group by.....

Liste die Personalkosten der Abteilung 12:

```
SELECT SUM (Lohn)
FROM P
WHERE Abt-Nr = 12
```

SQL

Verbund (Join)

Liste jede Maschine mit den Namen der Personen, die auf ihr ausgebildet sind (sortiert nach M-Nr).

```
SELECT M-Nr, Name
FROM P, P-M
WHERE P.P-Nr = P-M.P-Nr
ORDER by M-Nr
```

SQL

Daten einfügen, ändern, löschen:

```
insert
into P-M (P-Nr, M-Nr)
values (302,2)
```

Der Mitarbeiter mit der Personalnummer
302 hat Ausbildung auf Maschine 2 beendet

```
update P
set Lohn = Lohn * 1,2
where Abt-Nr = 10
```

Alle Mitarbeiter der Abteilung 10 bekommen
eine Gehaltserhöhung um 20 %

```
delete
from P-M
where M-Nr = 4
```

Maschine 4 wurde verkauft

SQL

Views:

Tables sind physische Relationen (Dateien). Views dagegen sind logische Relationen. Sie dienen dazu, dem Benutzer einen für ihn maßgeschneiderten Ausschnitt der gesamten DB zur Verfügung zu stellen.

```
CREATE VIEW PERSONAL (P-Nr, Name, Abt-Nr)
AS SELECT P.P-Nr, P.Name, P.Abt-Nr
FROM P
```

In der "View" Personal ist Lohn nicht vorhanden

SQL

Sicherheit und Integrität:

GRANT INSERT, DELETE, UPDATE ON Personal TO Schmid

**CREATE TABLE P (P-Nr DECIMAL (3) NOT NULL,
Name CHAR (20),
Abt-Nr DECIMAL (2),
Lohn DECIMAL (6),
UNIQUE (P-Nr))
CHECK (P.Abt-Nr BETWEEN 1 AND 15)**

Entity Integrity:

gewährleistet durch NOT NULL


SQL

Referential Integrity:

**CREATE TABLE P (P-Nr DECIMAL (3) NOT NULL,
Name CHAR (20),
Abt-Nr DECIMAL (2),
Lohn DECIMAL (6),
PRIMARY KEY (P-NR))**

**CREATE TABLE P-M (P-Nr DECIMAL (3) NOT NULL,
M-Nr DECIMAL (2) NOT NULL,
PRIMARY KEY (P-Nr, M-Nr)
FOREIGN KEY (P-Nr) REFERENCES P)**

Referential
Integrity
gewährleistet



Query by Example

Charakteristik:

- deskriptiv
- mengenorientiert
- graphisch

Da die Datenabfrage graphisch unterstützt wird, kommt diese Sprache speziell dem ungeübten Benutzer entgegen.

Query by Example

Ablauf:

- Bestimmung der Relationen durch den Benutzer
- System wirft Roh-tabelle mit Attributen auf den Bildschirm
- Benutzer macht exemplarische Eingaben

Eingaben:

- Mayer.....unterstrichene Werte stehen für ein Beispiel einer möglichen Antwort (Variable)
- P. Print-Befehl
- Mayer echter Wert (Konstante)
- Verbund wird über sog. linking-Variable

Query by Example

Liste die Namen aller Personen der Abteilung 12

P	P-Nr	Name	Abt-Nr	Lohn
		<u>P.Mayer</u>	12	

← Rohtabelle

exemplarische Angabe

Query by Example

Liste die Namen der Personen, die auf Maschine 4 ausgebildet sind

P	P-Nr	Name	Abt-Nr	Lohn
	<u>99</u>	<u>P.Mayer</u>		

P-M	P-Nr	M-Nr
	<u>99</u>	4

linking Variable für Verbund

Studie: Datenmodellierung in der Praxis

"Datenmodellierung ermöglicht die exakte Abbildung der realen Welt und wird hauptsächlich zur Definition von fachlichen Anforderungen an Informationssysteme eingesetzt. Datenmodellierung ist aber darüber hinaus ein wichtiger betriebswirtschaftlicher Faktor. Sie ist mitentscheidend für Produktivität und Qualität in der Informationsverarbeitung und somit für die Wettbewerbsfähigkeit von Unternehmen."

(Zandt Norbert: Nutzenargumente zur Datenmodellierung. In: Information Management 4/93, S. 81-82)

Studie: Datenmodellierung in der Praxis

- **Einführungsaufwand pro Mitarbeiter: 12 - 14 Manntage**
- **Durchschnittliche Amortisationszeit: 2,3 Jahre**
- **Erwarteter Nutzen:**
 1. bessere Qualität der Arbeitsergebnisse
 2. erhöhte Produktivität
 3. verbesserte Kommunikation im Unternehmen
 4. ganzheitliches Denken und Vorgehen auf unternehmensweiter Ebene wurde gefördert

Studie: Datenmodellierung in der Praxis

- **Produktivitätssteigerung:**

- 20 %: keine Angabe
- 4 %: negativ
- 17 %: keine Auswirkungen
- 34 %: 1 - 10 %
- 18 %: 11 - 25 %
- 7 %: > 25 %

–

Volltext-Datenbanksysteme

Eigenschaft:

- zur Abspeicherung von umfangreichen Texten (unstrukturierte Informationen)
- einzelnen Wörter (außer Stoppwörter) werden in einer invertierten Liste abgelegt

Vorteile:

- Suche nach einzelnen Wörtern möglich
- schnell (binäre Suche)

Nachteil:

- Erweiterung/Aktualisierung der Textdatenbank sehr zeitaufwendig (z. B. ca. 10 Minuten für 1 MB (386-Prozessor) intertierte Listen müssen umorganisiert werden ==> im Hintergrund durchführen)

Objektorientierte Datenbanksysteme

Objektorientierte Datenmodelle beseitigen einige Unzulänglichkeiten des relationalen Datenmodells:

Im relationalen Modell werden Objekt- und Beziehungstypen durch Relationen modelliert. Diese Art der Modellierung bringt aber einige Nachteile mit sich:

1. Keine strukturierten Attribute möglich:

Nicht zuletzt aufgrund der 1. Normalform müssen alle Attribute atomar sein. Dadurch geht aber die Struktur komplexerer Attribute verloren.

Beispiel:

STUDENT	NName	VName	Straße	Plzl	Ort

Objektorientierte Datenbanken

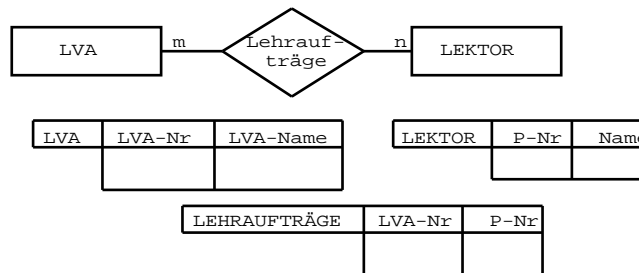
besser wäre also:

STUDENT NName	Name		Adresse		
	VName	Straße	Plzl	Ort	

Objektorientierte Datenbanken

2. Für m:n-Beziehungstypen muß eine eigene (künstliche) Relation eingeführt werden.

Beispiel:



3. In der Realität verhalten sich Objekttypen und Beziehungstypen verschieden, z. B. beim Löschen.

Objektorientierte Datenbanken

Objektorientiertes (Daten)Modell:

- Behebung der Nachteile des relationalen Modells
- realitätsnähere Datenmodellierung (PERSON, ADRESSE, ...) ==> höherer Abstraktionsgrad
- Berücksichtigung nicht-datensatzorientierter Daten (Graphik, Sprache, Video, ...) und deren optimale Umsetzung in Speicherstrukturen
- keine (künstliche) Trennung von Daten (Daten-Datenbank) und Programmen (Programm-Datenbank) ==> Beseitigung der Änderungsproblematik: Klasse (Objektyp) = Daten + Programme

Objektorientierte Datenbanken

OODBMS - Fähigkeiten:

- Datenabstraktion
- gute Objektmodellierungsfähigkeiten
- "Objektidentität"
- "Datenkapselung" und "data hiding"
- aktive Daten
- Vererbung
- Nachrichtenaustausch
- Erweiterbarkeit

Objektorientierte Datenbanken

Vorteile:

- die Modellierung der realen Welt ist wesentlich einfacher
- wesentlich schnellere Anwendungen, speziell im multimedialen Bereich
- wesentliche Einsparungen bei der Erstellung von Programmen: z. B. Fa. AT&T's Bell Laboratories: 500 Programmierer, die an 150 Applikationen arbeiten - vor Einführung einer objektorientierten Programmiersprache: 20 % der bestehenden Programme konnten wiederverwendet werden, 80 % mußten neu geschrieben werden, nach Einführung: umgekehrtes Verhältnis.
- Schätzungen: Kostensenkungen auf 1/10 bis 1/5 der jetzigen Entwicklungskosten, andere Schätzung: Kosteneinsparung von 20 % - 30 %.

Zeittafel - Datenbanken

- **1962: IDS (Netzwerkdatenbank)**
- **1968: IMS (hierarchische Datenbank)**
- **1970: CODASYL DBTG - Normung von Netzwerkdatenbanken**
- **1970: Vorstellung des relationalen Modells durch Codd**
- **1972: Definition der ersten drei Normalformen durch Codd**
- **1974: Definition von SQL**
- **1975: Dreischichtenmodell (ANSI SPARC)**
- **1975: Query by Example**
- **1976: Entity Relationship Modell (Chen)**
- **80er-Jahre: Verbreitung von Datenbanken auf PCs**

Zeittafel - Datenbanken

- **Mitte/Ende 80er-Jahre: erste kommerzielle Hypertext-/Hypermediasysteme**
- **1989: erste Datenbankserver (Ashton Tate, Microsoft, Sybase)**
- **Angang 90:**
 - **Multimedia-Datenbanken**
 - **Windows-Datenbanksysteme (graphische Benutzeroberflächen) verteilte Datenbanksysteme**
 - **"Aktionsdatenbanken" (Triggerkonzept)**
 - **heterogene Datenbanksysteme**
 - **induktive Datenbanksysteme**
- **90er-Jahre: Verbreitung von objektorientierten Datenbankmanagementsystemen?**

Zeittafel - Datenbanken

- **Mitte 90er-Jahre: Siegeszug des Internet**
- **Mitte/Ende 90er-Jahre: Datenbankanbindung an Internet**
- **90er-Jahre: Verbreitung von objektorientierten Datenbankmanagementsystemen?**