

VU „Methoden der Wirtschaftsinformatik 2
“Folienskriptum, SS 2011

a.Univ.-Prof. Dipl.-Ing. Mag. Dr. Christian Schlögl
Institut für Informationswissenschaft und
Wirtschaftsinformatik
Universitätsstrasse 15/F3
8010 Graz

Inhalt

4. Datenbanksprachen

- SQL
- Query by Example

5. MS Access

Ziele

1. Bewusstsein für die Bedeutung der Daten
2. Erstellung von einfacheren Datenmodellen
3. Erstellung von Datenbankanwendungen mit MS Access

Methode und Anforderungen

Methode:

- Vortrag durch den LVA-Leiter
- dazu laufend praktische Übungen
- Arbeiten am Computer

Anforderungen:

- Kleines Anwendungsprojekt: Projektbericht und Präsentation (50 % der vorläufigen Gesamtbewertung)
 - Klausur (50 % der vorläufigen Gesamtbewertung)
 - ca. 5 Hausübungen (bis auf eine Hausübungen müssen alle abgegeben werden)
 - 1 Access-Aufgabe am Computer
- (Alle Teilbewertungen müssen positiv sein.)
- laufende Mitarbeit (Verbesserung oder Verschlechterung um einen Notengrad)

Inhalt

1. Historische Entwicklung
2. Konzeptioneller Entwurf/semantische Datenmodellierung
 - Entity-Relationship-Diagramm
 - Unternehmensweite Datenmodellierung
 - Management-Informationssysteme (MIS)
3. Logischer Entwurf
 - Hierarchisches Modell
 - Netzwerkmodell
 - RELATIONALES MODELL
 - Denormalisieren

Literaturliste

1. Datenmodellierung:

- Rauh O, Stickel E.: Konzeptuelle Datenmodellierung, Teubner, Stuttgart, Leipzig 1997 – umfassendes Werk zur konzeptionellen Datenmodellierung.
- Steiner René: Grundkurs Relationale Datenbanken, 5. Aufl., Vieweg, Braunschweig 2003 – didaktisch gut aufgebaut, neben Datenmodellierungswissen werden auch SQL- und Projektmanagementkenntnisse vermittelt
- Stickel Eberhard: Datenbankdesign: Methoden und Übungen, Gabler-Verlag, Wiesbaden 1991 - relativ einfach und leicht verständlich (Semesterhandapparat)
- Vetter M.: Aufbau betrieblicher Informationssysteme mittels pseudo-objektorientierter konzeptioneller Datenmodellierung, 8. Auflage, Teubner, Stuttgart 1998 – semantische Datenmodellierung (etwas Vetter-spezifisch und relationales Modell, gute Verständlichkeit)
- Lackes Richard, Brandl Wolfgang, Siepermann Markus: Datensicht von Informationssystemen. PC-Trainer (CDROM), Springer, 1998, ISBN 3-540-14700-4 – einfache eLearning-Anwendung zur ER-Modellierung.

Literaturliste

2. Datenbanken - allgemein:

- Zehnder: Informationssysteme und Datenbanken, Teubner Verlag, 7. Auflage, vdf-Hochschulverlag, 2001
- Date C. J.: An Introduction to database systems, 5. Auflage, Band 1, Reading / Mass. u.a. 1991 - der "Klassiker" im Bereich der relationalen Datenbanksysteme.

3. MS Access

- Diverse

HISTORISCHE ENTWICKLUNG: Einzeldateien ohne Integration

Charakteristik:

- jeder Benutzer arbeitet mit privaten Files, 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.

Datenbankkenntnisse - wozu?

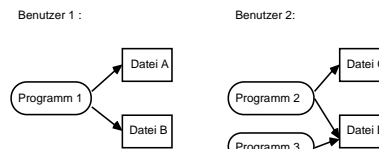
1. Erstellung von kleineren Datenbankanwendungen

Kleinere Datenbankanwendungen sollten selbständig erstellt werden können. Relationale Datenbankmanagementsysteme im PC-Bereich (z. B. MS Access) sind mittlerweile so benutzerfreundlich, dass dies auch ohne umfangreiche EDV-Kenntnisse möglich ist.

2. Informationsvorsprung

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

Einzeldateien ohne Integration



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)

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:

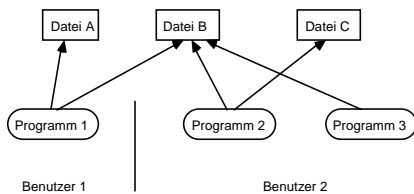
- die Funktion der Datenerhebung, -kontrolle und -speicherung wird zentralisiert, diese Funktion wird durch den "Datenkoordinator" wahrgenommen
- die Daten werden einheitlich in einem "data dictionary" beschrieben: für jedes Feld (einer jeden Datei) erfolgt eine genaue Beschreibung des Inhaltes
- weiterhin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

Datenbanksystem: Zwei-Schichten-Modell

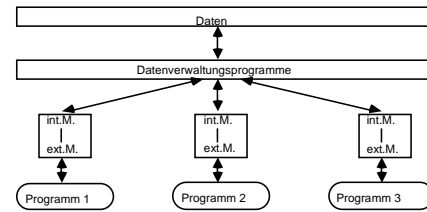
Charakteristik:

- 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.

Physische Integration der Dateien



Zweischichten-Modell



Physische Integration der Dateien

Vor- und Nachteile:

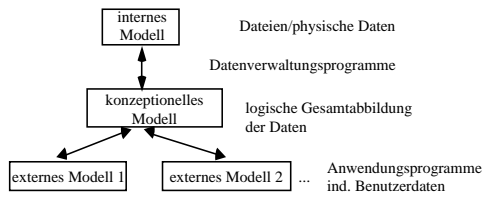
- Vorteile gegenüber Einzeldateien ohne Integration:
 - Daten sind nicht mehr mehrfach gespeichert
- Nachteile:
 - bei gravierenden Änderungen der Datenstruktur sind wahrscheinlich sogar mehr Programme von Änderungen betroffen
 - nunmehr ist es auch möglich, auf vertrauliche Daten zuzugreifen

Datenbanksystem: Drei-Schichten-Modell

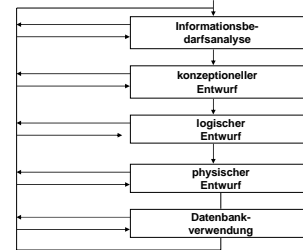
Logische Datenunabhängigkeit ist die Möglichkeit, logische Änderungen an der Datenbasis durchzuführen, ohne die darauf zugreifenden Anwendungsprogramme signifikant zu beeinflussen.

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

Drei-Schichten-Modell



SCHRITTE BEIM DATENBANKENTWURF



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.

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 (Informations)Objekte
 - erforderliche Operationen
 - reguläre Ausnahmen

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).

Informationsbedarfsanalyse

DATA-ID-Verfahren:

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", ...

KONZEPTIONELLER ENTWURF / SEMANTISCHE DATENMODELLIERUNG

Datenmodellierung =
der Vorgang, das konzeptionelle Modell aus der Realität abzuleiten.

Datenmodelle =

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

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 (Kardinalität)

Konzeptioneller Entwurf

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

1. Benennen:

Einem Objekt der realen Welt und einer 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!

Beziehungsarten (Kardinalitäten)

• **1:1-Typ:**

einem Objekt des ersten Objektyps wird genau ein Objekt des zweiten Objektyps zugeordnet, vice versa

• **1:n-Typ:**

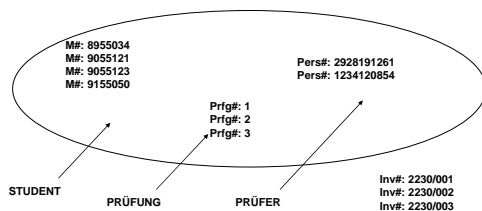
einem Objekt des ersten Objektyps werden beliebig viele (auch null) Objekte des zweiten Objektyps zugeordnet, einem Objekt des zweiten Objektyps wird nur ein Objekt des ersten Objektyps zugeordnet

• **m:n-Typ:**

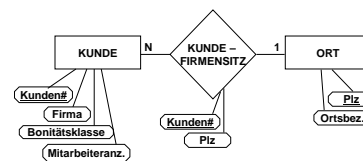
einem Objekt des ersten Objektyps werden n Objekte des zweiten Objektyps zugeordnet, vice versa.

Konzeptioneller Entwurf

Beispiel - Prüfungsverwaltung:



Beispiel zu ERM



1:N-Beziehungsart:

- ein Kunde hat an genau einem Ort den Firmensitz
- An einem Ort können mehrere Kunden den Firmensitz haben

Beispiel ERM



Unterschiede zu Access (Beziehungsfenster):

- Beziehungstypen werden nur als Kante dargestellt (→ Plz (Fremdschlüssel) wandert in die Tabelle „Kunden“)
- Access kennt nur 1:1 und 1:N-Beziehungen
- ER-Modelle können auf viel höherem Aggregationsgrad (bzw. geringerem Detaillierungsgrad) erstellt werden (hängt vom jew. Zweck ab).

Benennung von Objekt- und Beziehungstypen

Grundsätzlich sollten Bezeichnungen gewählt werden, wie sie aus dem täglichen Umgang bzw. aus der Fachsprache bekannt sind.

Benennung von Objekttypen:

- Üblicherweise Substantive bzw. Hauptwörter
- **Negativbeispiel:** AVB-191261, Bestelldaten

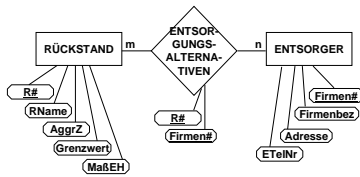
Benennung von Beziehungstypen:

Möglichkeiten:

- Ebenfalls Substantive
- Verben bzw. Tätigkeitswörter
- Kunstworte

Negativbeispiele: inkonsistente Benennung, Präpositionen

Beispiel zu ERM



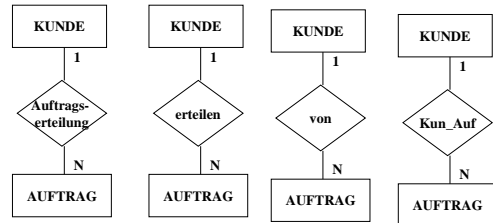
BEACHTET:
Attribute müssen richtig (dem passenden Objekt- bzw. Beziehungstyp) zugeordnet sein.
Keine redundanten Attribute (Ausnahme: Schlüsselattribute).
Beziehungstyp muss die Schlüsselattribute der an der Beziehung teilnehmenden Objekttypen (Fremdschlüssel) enthalten.

m:n-Beziehungsart:

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

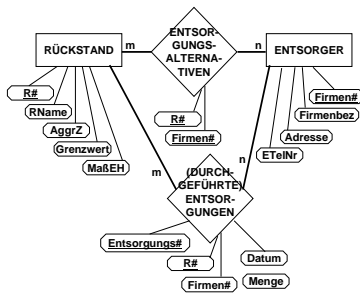
Benennung von Objekt- und Beziehungstypen

Alternative Benennungen eines Beziehungstyps:

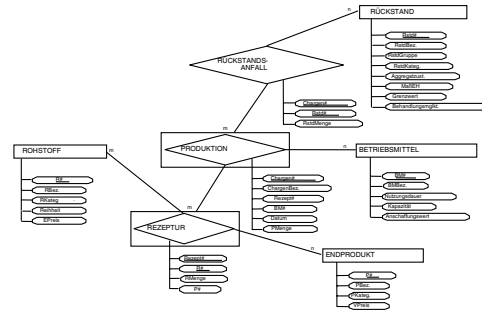


Beispiel zu ERM

Zwischen 2 Objekttypen können auch mehrere Beziehungen bestehen:



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

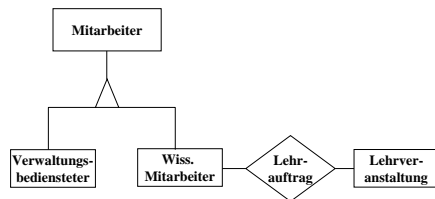
Ziel = fachliche Beschreibung aller im Unternehmen verwendeten Informationsobjekte (Objekttypen) und deren Beziehungen zueinander (Beziehungstypen) [ENDL 1992]

Vorgehensweise:

- Top-Down
- Bottom-Up

ER-Erweiterungen

Generalisierung/Spezialisierung:



Unternehmensweite Datenmodellierung

Top-Down-Vorgehensweise:

1. Erarbeiten des Unternehmensmodells (= stark aggregiertes Datenmodell): Umfang zum Beispiel 70 Objekt- und 130 Beziehungstypen
2. Unternehmensmodell 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.

ER-Erweiterungen

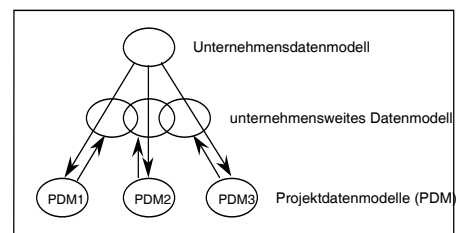
Generalisierung/Spezialisierung:

Vorteile:

- **Vererbung:** Eigenschaften der „oberen“ Objekte (Attribute, Beziehungstypen) gelten auch für die darunter liegenden, diese Eigenschaften müssen auf den unteren Ebenen nicht noch einmal definiert werden
- Differenzierte Darstellung von Begriffen
- Integritätsbedingungen, die nur für ein „Unterobjekt“ gelten – z. B.: nur wissenschaftliche Mitarbeiter dürfen Lehraufträge erhalten – lassen sich direkt mit dem ER-Diagramm zeigen

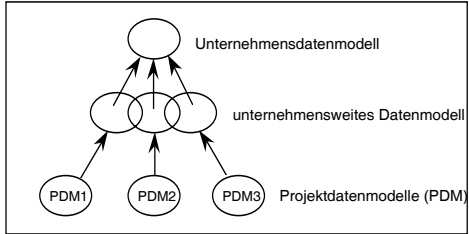
Unternehmensweite Datenmodellierung

Top-Down-Vorgehensweise:

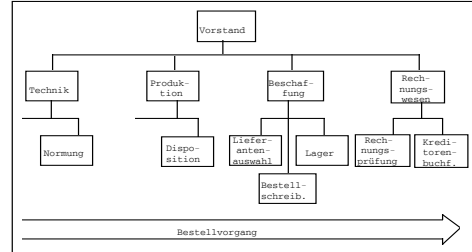


Unternehmensweite Datenmodellierung

Bottom-Up-Vorgehensweise:



Unternehmensweite Datenmodellierung (Nutzen integrierter Daten)



(Quelle: Scheer 1990)

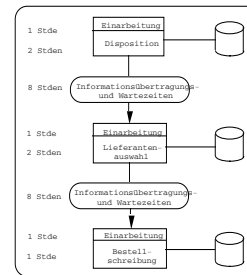
Unternehmensweite Datenmodellierung

Gründe/Nutzen [ENDL 1992]:

- 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

Unternehmensweite Datenmodellierung (Nutzen integrierter Daten)

Durchlaufzeit ohne Datenintegration:



(Quelle: Scheer 1990)

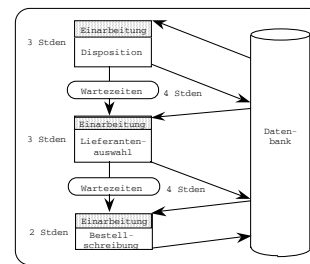
Unternehmensweite Datenmodellierung

Gründe/Nutzen:

- Vermeidung von redundanten Daten - Datenstruktur soll möglichst anwendungsunabhängig entworfen werden
- 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 (Nutzen integrierter Daten)

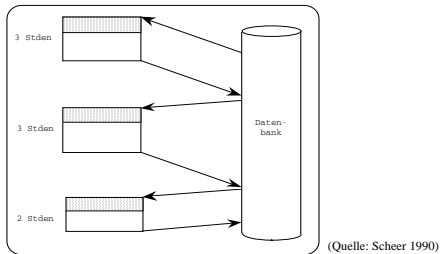
Durchlaufzeit bei Datenintegration:



(Quelle: Scheer 1990)

Unternehmensweite Datenmodellierung (Nutzen integrierter Daten)

Durchlaufzeit bei gezielter Zuteilung von Teilvorgängen:



LOGISCHER DATENBANKENTWURF

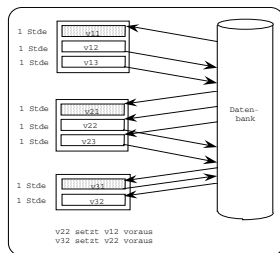
Klassische (logische) Datenmodelle:

1. graphenorientierte Modelle:
 - Hierarchisches Modell
 - Netzwerkmodell
2. tabellenorientierte Modelle:
 - Relationales Modell.

Das verwendete Datenmodell beeinflusst natürlich die Datenmodellierung.

Unternehmensweite Datenmodellierung (Nutzen integrierter Daten)

Durchlaufzeit bei weiterer Zerlegung der Teilvorgänge



Hierarchisches Modell

Das hierarchische Modell enthält:

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

Daraus folgt:

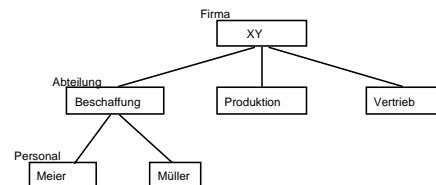
- Zyklen sind nicht erlaubt
- jeder untergeordnete Objekttyp darf höchstens einen übergeordneten Objekttyp haben, es gibt einen oder mehrere Objekttypen ohne Vorgänger (Wurzeltyp).

Unternehmensweite Datenmodellierung

Fallstudie: Entwurf eines unternehmensweiten Datenmodells für den Schweizerischen Bankverein

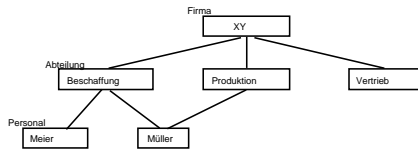
- Initialisierungsworkshop: 2 Tage, 12 Vertreter aus In- und Ausland und allen Geschäftssparten
-> bedeutende Entitätsmengen wurden grob umrissen
- Arbeitsgruppe: 7 Mitarbeitern, sechs mehrtägige Treffen,
-> 40-seitiges Dokument mit wichtigsten Entitätsmengen und Beziehungstypen, Aufwand der Arbeitsgruppe: 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!!!

Hierarchisches Modell



Netzwerkmodell

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



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, ...

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 Zwischenobjekte 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.

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 Attributmengende, die eine Tabelle eindeutig identifiziert, wird Schlüssel bezeichnet.

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 [MATH 1987]
- satzorientierte Verarbeitung

RELATIONENMODELL

Nomenklatur:

- **Relation (Tabelle):**
Eine Relation wird als Menge von Tupel definiert. Aus der Mengendefinition folgt, dass
 - die Reihenfolge der Tupel unwesentlich ist und
 - es keine zwei gleiche 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.

ER-Modell → Relationenmodell

Bei der Überführung eines ER-Modells in ein Relationenmodell ist Folgendes zu beachten:

1. Objekttypen werden 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 werden die Attribute des Beziehungstyps bei einem der beiden an dieser Beziehung teilnehmenden Objekttypen untergebracht. Wenn es sinnvoll erscheint, können die beiden Objekttypen zu einer Relation zusammengefasst werden.

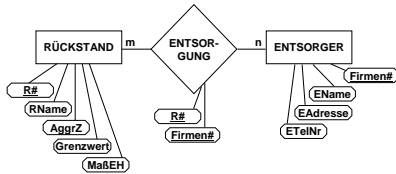
Operationen auf Relationen

Es gibt folgende Operationen:

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

ER-Modell → Relationenmodell

Beispiel:



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.

ER-Modell → Relationenmodell

RÜCKSTAND	ENTSORGER	ENTSORGUNG
<u>R#</u>	<u>EFirmen#</u>	<u>R#</u>
RName	EName	EFirmen#
RAggrZ	EAdresse	
RGrenzwert	ETeilNr	
RMaßEH		

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

Verbund:

Bei der Verbundoperation werden zwei Relationen mit einem gemeinsamen Attribut zu einer neuen Relation verbunden. Das neue Relationenschema 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

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

Aufgabe: RM-Op1

Gegeben seien folgende zwei Relationen:

P	P-Nr	Name	Abt-Nr	Lohn	P-M	P-Nr	M-Nr
	300	Meier	12	12.000		301	4
	301	Huber	10	14.000		301	3
	302	Müller	7	10.900		303	1
	303	Bauer	5	13.000		304	4
	304	Schmid	12	17.000		300	2
						300	1

Es soll ein Bericht erstellt werden, der Auskunft darüber gibt, welche Mitarbeiter auf welchen Maschine eingesetzt werden können. Die Ergebnisrelation soll folgende Spalten enthalten (Projektion): Maschinenummer, Personalnummer und Name. Welche Tupel enthält die Ergebnisrelation?

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.

Operationen auf Relationen

Verbund:

Zusammenhang zwischen Verbund-Operation und 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.

Normalformen

Intuitiv aufgestellte Relationen enthalten folgende Anomalien:

- Speicheranomalie
- Löschanomalie
- Änderungsanomalie.

Normalformen

Beispiel:

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

AUFG-MA	Aufgabenbereich	Mitarbeiter	Abteilung
	Maschinen	Meier	Produktion
	Maschinen	Müller	Produktion 1
	EDV	Müller	Produktion 1
	Bestellwesen	Kunz	Beschaffung

1. Normalform

Entwurf der Relationenschemata:

- intuitiv
- regelbasiert (Normalformen): Einschränkungen, denen die Relationenschemata genügen müssen, um (oben beschriebene) Anomalien zu vermeiden

1. Normalform

Ein Relationenschema ist in erster Normalform, wenn

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

Normalformen

Anomalien:

- Speicheranomalie:
Man möchte speichern, dass der 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 nach wie vor 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.

2. Normalform

Funktionale Abhängigkeit:

Eine Attributmenge X bestimmt die Attributmenge 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 Attributmenge X (z. B. $\{X_1, X_2, X_3\}$) voll funktional abhängig, genau dann, wenn es keine Teilmenge von X (z. B. $\{X_1, X_2\}$) gibt, die A funktional bestimmt.

Normalformen

Anomalien:

- Änderungsanomalie:
Wechselt nun Herr Müller von der Abteilung Produktion 1 in die Abteilung Produktion 2, so muss man alle Tupel suchen und ändern, in denen Herr Müller vorkommt.

Derartige Anomalien können durch einen geeigneten Entwurf von Relationenschemata umgangen werden.

2. Normalform

Nunmehr ergibt sich eine Erweiterung des Begriffes des Relationenschemas:

Ein Relationenschema setzt sich somit 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 (falls dieser aus mehreren Attributen besteht).

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 (bzw. von keinen Teilschlüssel) abhängig ist.

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

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).

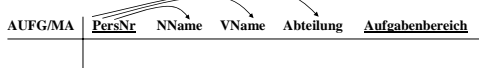
Ein Relationenschema ist in 3. Normalform, wenn es in 2. Normalform ist und zusätzlich gilt, dass kein Nichtschlüsselattribut von irgend einem Schlüssel transitiv abhängig ist.

2. Normalform

Beim Prüfen auf 2. Normalform fallen folgende Schritte an:

1. Schlüssel bestimmen: $\{PersNr, Aufgabenbereich\}$
2. Funktionen bestimmen: $PersNr \rightarrow NName$, $PersNr \rightarrow VName$, $PersNr \rightarrow Abteilung$
3. Prüfung, ob alle Nichtschlüsselattribute vom gesamten Schlüssel (nicht nicht etwa von einer Teilmenge davon) funktional abhängig sind: $NName$, $VName$ und $Abteilung$ sind nicht vom gesamten Schlüssel ($\{PersNr, Aufgabenbereich\}$) sondern nur von einer Teilmenge davon funktional abhängig.

Beispiel:



3. Normalform

Beispiel:

KuNr	KuName	Plz	Ort
1	Meyer	3363	Hausmening
2	Gül	3300	Amstetten
3	Hofmarcher	3300	Amstetten
4	Tischler	3300	Amstetten

Annahme: Plz bestimmt Ort funktional (entspricht nicht Realität!!!)

Z. B. unter der Postleitzahl 3332 werden zwei Orte „geführt“: Allhartsberg und Kröllendorf)

1. Formale Erklärung:

$KuNr \rightarrow KuName$

$KuNr \rightarrow Plz$

$KuNr \rightarrow Ort$

$Plz \rightarrow Ort$ (lt. Annahme)

$KuNr \rightarrow Plz$ und $Plz \rightarrow Ort \implies KuNr \rightarrow Ort$

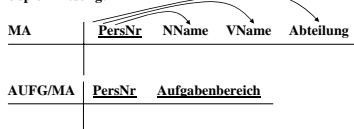
Das Nichtschlüsselattribut Ort wird transitiv (über die Funktionen $KuNr \rightarrow Plz$ und $Plz \rightarrow Ort$) vom Schlüssel bestimmt. Auf diese Funktion ($KuNr \rightarrow Ort$) kann (beim Zerlegen) verzichtet werden.

2. Normalform

Falls ein Relationenschema nicht in 2. Normalform ist, muss es folgendermaßen in mehrere Relationenschemata zerlegt werden:

- Der ganze Schlüssel darf nicht verloren gehen
- Er dürfen keine Funktionen verloren gehen.

Beispiel - Lösung:



3. Normalform

\implies

KuNr	KuName	Plz
1	Meyer	3363
2	Gül	3300
3	Hofmarcher	3300
4	Tischler	3300

Plz	Ort
3363	Hausmening
3300	Amstetten

$Plz \rightarrow Ort$

$KuNr \rightarrow KuName$

$KuNr \rightarrow Plz$

Auf $KuNr \rightarrow Ort$ kann verzichtet werden, da diese Funktion aus $KuNr \rightarrow Plz$ und $Plz \rightarrow Ort$ hergeleitet werden kann.

3. Normalform

KuNr	KuName	Plz	Ort
1	Meyer	3363	Hausmening
2	Gül	3300	Amstetten
3	Hofmarcher	3300	Amstetten
4	Tischler	3300	Amstetten

Logische Erklärung:

In obiger Tabelle kommen Orte („Amstetten“) mehrmals vor. Diese Redundanzen lassen sich dadurch beseitigen, dass obige Tabelle in eine Kundentabelle und in eine Ortentabelle aufgespalten wird.

3. Normalform

InstBuchNr	Titel	EJahr	Verlag	KlassNr
1	Rel. DBMS	1990	Elsevier	HIW-038
2	Information Management	2005	Springer	HIW-035
3	Data Warehousing	2004	Teubner	HIW-038
4	IT does not matter	2004	Wesely	HIW-035
...				
100000	Einführung in die InfoWiss	2001	UVK	HIW-035

100000 Bücher

3. Normalform

Beispiel: Bibliothek

InstBuchNr	Titel	EJahr	Verlag	KlassNr	KlassBezeichnung
1	Rel. DBMS	1990	Elsevier	HIW-038	Datenbanksysteme
2	Information Management	2005	Springer	HIW-035	Informationsmanagement
3	Data Warehousing	2004	Teubner	HIW-038	Datenbanksysteme
4	IT does not matter	2004	Wesely	HIW-035	Informationsmanagement
...					
100000	Einführung in die InfoWiss	2001	UVK	HIW-035	Informationsmanagement

100000 Bücher

die ca. 1000 verschiedenen Klassen zugeordnet werden
 → Hohe Redundanzen bei Klassifikationsbezeichnung

3. Normalform

Tabelle KLASSIFIKATION (mit maximal 1000 Datensätzen)

KlassNr	KlassBezeichnung
...	...
HIW-035	Informationsmanagement
...	
HIW-038	Datenbanksysteme
...	

3. Normalform

InstBuchNr → KlassNr und
 KlassNr → KlassBezeichnung

==>

InstBuchNr → KlassBezeichnung

Das Nichtschlüsselattribut KlassBezeichnung ist vom Schlüssel (InstBuchNr) transitiv (funktional) abhängig

→ Relation ist nicht in 3. NF. → Aufspaltung in zwei Tabellen, wobei InstBuchNr → KlassBezeichnung entbehrlich ist (da sie transitiv hergeleitet werden kann)

3. Normalform

Algorithmus zum Zerlegen in 3. Normalform:

1. Alle Abhängigkeiten f mit identischer linker Seite werden zusammengefasst: $(X \rightarrow A1, X \rightarrow A2, \dots, X \rightarrow An) \Rightarrow (X \rightarrow A1, A2, \dots, An)$.
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.

4. Normalform

Ein Relationenschema ist in 4. Normalform, wenn es keine mehrwertigen Abhängigkeiten enthält.

Ein Relationenschema $RS(X,Y,Z)$ mit einer mehrwertigen Abhängigkeit $X \twoheadrightarrow Y/Z$ muss in zwei Relationenschemata $RS1(X,Y)$ und $RS2(X,Z)$ aufgespalten werden

4. Normalform

Mehrwertige (schwache) Abhängigkeiten:

$X \twoheadrightarrow Y/Z$ (X bestimmt Y mehrwertig in Bezug auf Z) : \Leftrightarrow

Wenn ein X-Wert x eine Menge von Y-Werten bestimmt
 DANN muss jeder dieser Y-Werte mit allen Z-Werten auftreten,
 Die von x bestimmt werden

4. Normalform

BuchNr	Autor
1	Meier
1	Müller
2	Huber

BuchNr	Schlagwort
1	Programmierung
1	Datenbanken
1	Datenmodellierung
2	Programmierung

4. Normalform

BuchNr	Autor	Schlagwort
1	Meier	Programmierung
1	Meier	Datenbanken
1	Meier	Datenmodellierung
1	Müller	Programmierung
1	Müller	Datenbanken
1	Müller	Datenmodellierung
2	Huber	Programmierung

X-Wert 1 bestimmt Y-Werte {Meier, Müller}
 X-Wert 1 bestimmt Z-Werte {Programmierung, Datenbanken, Datenmodellierung}
 \Rightarrow {Meier, Müller} müssen jeweils auch mit {Programmierung, Datenbanken, Datenmodellierung} auftreten (siehe oben)
 detto: X-Wert 2
Logische Erklärung: obige Tabelle enthält redundante Daten: zum Beispiel sind der Autor „Meier“ Dreimal und das Schlagwort „Datenbanken“ zweimal unnötigerweise (redundant) gespeichert.

DENORMALISIERUNG

Gegeben seien folgende zwei Relationen (Dateien) - 4. Normalform:

Name	Straße	Plz
Tischer	Josefberg 5	3363
Gatterbauer	Hausmeninger Str. 110	3300
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 ich nur mehr auf eine Datei zugreifen, um eine Adresse zu drucken.

Name	Straße	Plz	Ort
Tischer	Josefberg 5	3363	Hausmening
Gatterbauer	Hausmeninger Str. 110	3300	Amstetten
Stiebellehner	Kühberg	3370	Seitenstetten
...

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:

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

Verdichtete Informationen:

KSt-Nr	KArt-Nr	Kosten
1	17	10000
1	29	13000
2	10	25000
3	33	13400

KSt-Nr	GKosten
1	23000
2	13000
3	13400

Eine zusätzliche Tabelle mit verdichteten Daten macht vor allem dann Sinn, wenn die Tabelle mit den Detaildaten sehr umfangreich ist und diese, z. B. bei Kostenstellenauswertungen immer wieder aufs Neue ausgewertet werden muss.

Unterschiedlich zusammengestellte Informationen:

Art-Nr	NttoVk	BttoVk
10	100	120
11	200	240
12	1000	1200

Da der Bruttoverkaufspreis einfach (und immer gleich) zu berechnen ist, ist es sinnvoll, diesen nicht abzuspeichern.

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

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):
2. Zerlegung von Relationen
 - nach Attributen (Vorwegnahme der Projektion)
 - nach Tupeln (Vorwegnahme der Selektion)

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.

Denormalisierung

Zerlegung nach Attributen:

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

Name	Plz	Name	Straße
Tischer	3363	Tischer	Josefberg 5
Gatterbauer	3300	Gatterbauer	Hausmeninger Str. 110
Stiebellehner	3370	Stiebellehner	Kühberg
...

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.

Denormalisierung

Zerlegung nach Tupeln:

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

Name	Straße	Plz	Ort
Gatterbauer	Hausmeninger Str. 110	3300	Amstetten
...	...	3300	Amstetten
...

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

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 Datenmanipulationssprachen:

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

Selbständigkeit einer Datenbanksprache:

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

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.

SQL

Charakteristik::

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

Operationen:

- Datendefinition
- Datenabfrage
- Datenmanipulation
- Integritätsprüfung
- Zugriffskontrolle

Datenbanksprachen

3. Nach der Art der Spezifizierung der gewünschten Informationen:

- prozedurale Sprachen: Es muss angegeben werden, "WIE" die gewünschte Informationen 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.

SQL

Musterbeispiel:

3 Relationen: P Personal
P-M . Personal-Maschinenausbildung
M Maschinen

P	P-Nr	Name	Abt-Nr	Lohn	P-M	P-Nr	M-Nr
	300	Meier	12	12.000		301	4
	301	Huber	10	14.000		301	3
	302	Müller	7	10.900		303	1
	303	Bauer	5	13.000		304	4
	304	Schmid	12	17.000		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

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

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

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 1500 € verdienen, ausgegeben werden. Das Ergebnis soll absteigend nach Lohn sortiert sein:

```
SELECT P-Nr, Name, Lohn  
FROM P  
WHERE Abt-Nr=12 AND  
      Lohn<1500  
ORDER BY Lohn DESC
```

SQL

Bei einer Bedingung sind folgende Vergleichsoperatoren möglich:

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

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

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 (Abt-Nr BETWEEN 1 AND 15)
```

SQL

Daten einfügen, ändern, löschen:

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

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

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

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

Query by Example

Charakteristik:

- deskriptiv
- mengenorientiert
- graphisch.

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

Beispiel: Abfrageassistent von MS Access

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)

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	Plz	Ort

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

Objektorientierte Datenbanken

besser wäre also:

STUDENT	Name		Adresse		
	NName	VName	Straße	Plz	Ort

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)

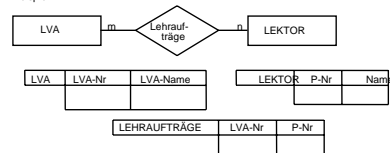
Nachteil:

- Erweiterung/Aktualisierung der Textdatenbank zeitaufwändig → invertierte Listen müssen umorganisiert werden ==> im Hintergrund durchführen)

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 (Objekttyp) = Daten + Programme

Zeittafel - Datenbanken

- Mitte/Ende 80er-Jahre: erste kommerzielle Hypertext-/Hypermediasysteme
- Anfang 90er Jahre:
 - Multimedia-Datenbanken
 - Windows-Datenbanksysteme (graphische Benutzeroberflächen)
 - verteilte Datenbanksysteme
 - "Aktionsdatenbanken" (Triggerkonzept)
 - objektorientierte Datenbankmanagementsysteme
 - induktive Datenbanksysteme

Objektorientierte Datenbanken

OODBMS - Fähigkeiten:

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

Zeittafel - Datenbanken

- Mitte 90er-Jahre: Siegeszug des Internet (WWW)
- Mitte/Ende 90er-Jahre: Datenbankanbindung an Internet
- 90er-Jahre: Weiterentwicklung der relationalen DBMS in Richtung objektrelationale DBMS
- Anfang 2000: XML in Verbindung mit DBMS

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