Anwendungsentwicklung Datenbanken Datenbankentwurf

Stefan Goebel

Warum eine Datenbank?

- Nutzung von gleichen Daten durch viele Anwender auch an unterschiedliche Orten
- Daten können mit unterschiedlicher Software genutzt werden
- Effiziente Verwaltung der Daten
- Sehr flexibel und schnell abfragbar
- Hohe Datensicherheit erreichbar

Datenbanksystem

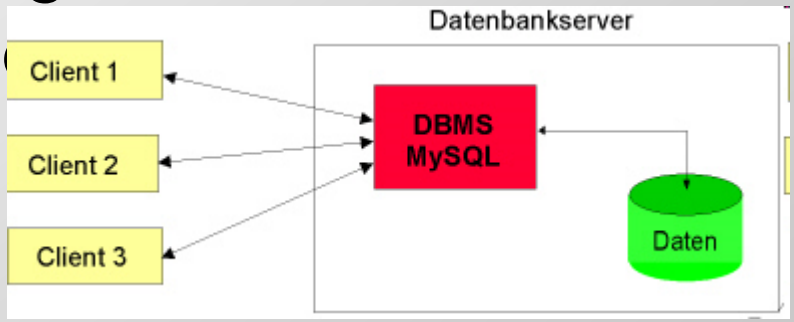
besteht aus:

- Datenbankmanagementsystem - die Software mit der verwaltet wird
- Anwenderdaten – das, was verwaltet wird
- Data Dictionary – beschreibt den Aufbau der Daten

Datenbankmanagementsystem

Ein Datenbankmanagementsystem ist ein aus einer Speicherungs- und einer Verwaltungskomponente bestehendes Programm. Die Speicherungskomponente erlaubt, Daten und ihre Beziehungen abzulegen, die Verwaltungskompenente stellt Funktionen und Sprachmittel zur Pflege und Verwaltung der Daten zur Verfügung.

Z.B. MySQL,



Datenbank

- Unter Datenbank versteht man ein System zur Beschreibung, Speicherung und Wiedergewinnung von umfangreichen Datenmengen.

- Abgrenzung der Begriffe
  Daten - Informationen und Wissen

Was sind Daten?

- Logisch gruppierte Informationseinheiten
- Digitale Repräsentation von
  - Information
  - realen Objekten abhängig von der Anwendung

Datenschutz

§ 3a Datenvermeidung und Datensparsamkeit

Gestaltung und Auswahl von Datenverarbeitungssystemen haben sich an dem Ziel auszurichten, keine oder so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen. Insbesondere ist von den Möglichkeiten der Anonymisierung und Pseudonymisierung Gebrauch zu machen, soweit dies möglich ist und der Aufwand in einem angemessenen Verhältnis zu dem angestrebten Schutzzweck steht. (BDSG)

3-Ebenen-Architektur

- große Datenunabhängigkeit
- Von ANSI/SPARC definiert:
  - Externe Ebene
    Sichten auf Teile der Daten, Masken und Dialoge
    (Oberflächendesigner)
  - Konzeptionelle, logische Ebene
    logische Gesamtsicht, Daten, Beziehungen, Zugriffsrechte, ER-Modell
    (Datenbankentwickler)
  - Interne, physische Ebene
    Speicherstrukturen
    (DBAdministrator)

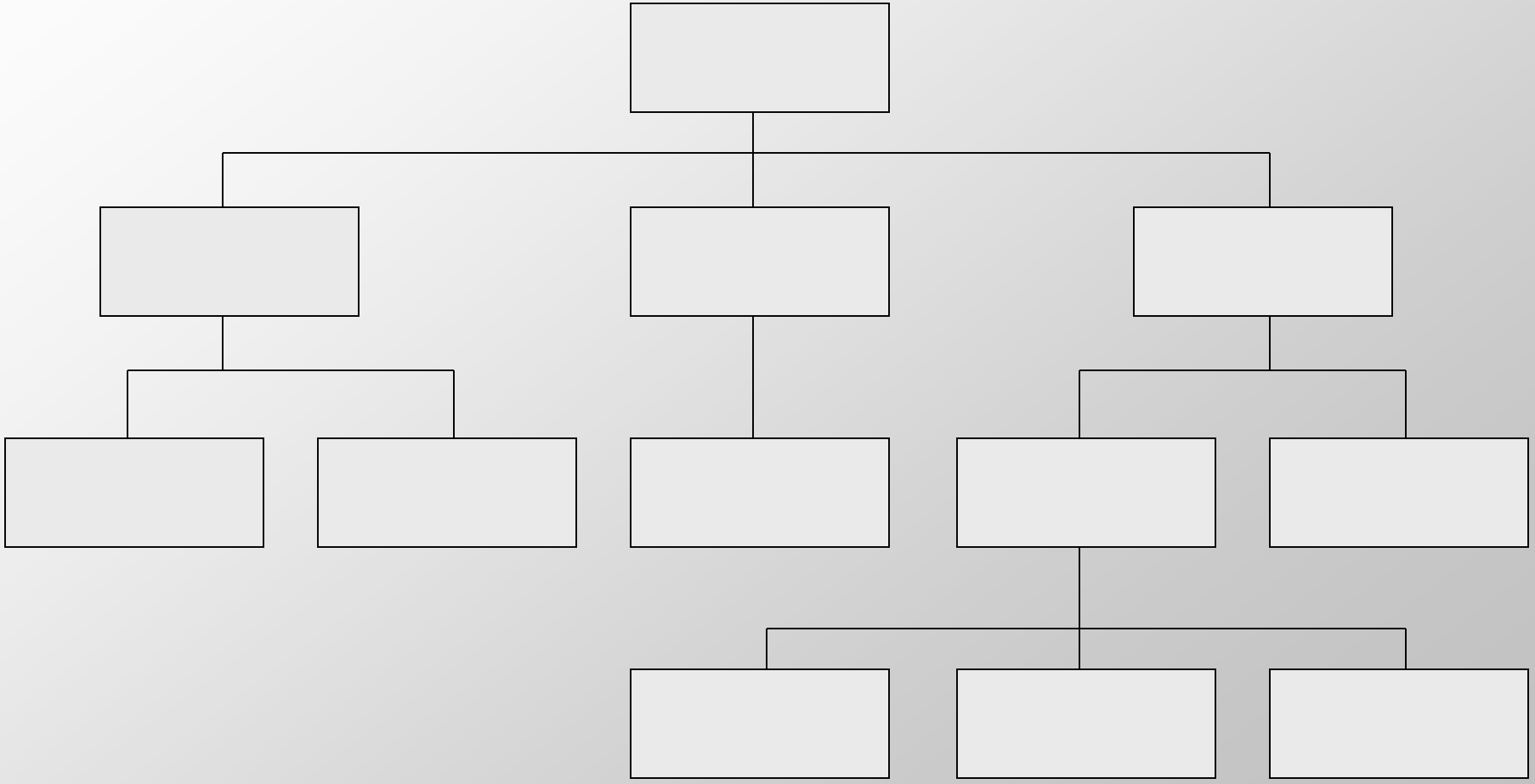
Geschichtliche Entwicklung

- Hierarchisches Datenmodell
- Netzwerk Datenmodell
- Relationales Datenmodell
  Entity-Relationship Modell
- Objektrelationales Datenmodell
- Objektorientiertes Datenmodell

Hierarchisches Datenmodell

- Grafische Darstellung der zu speichernden Daten und ihrer Beziehungen
- Eindeutige Beziehungen zwischen Objekttypen
- Jedes Element hat:
  - Einen Vorgänger
  - Beliebig viele Nachfolger
- Eindeutiger Weg zum Wurzelobjekt

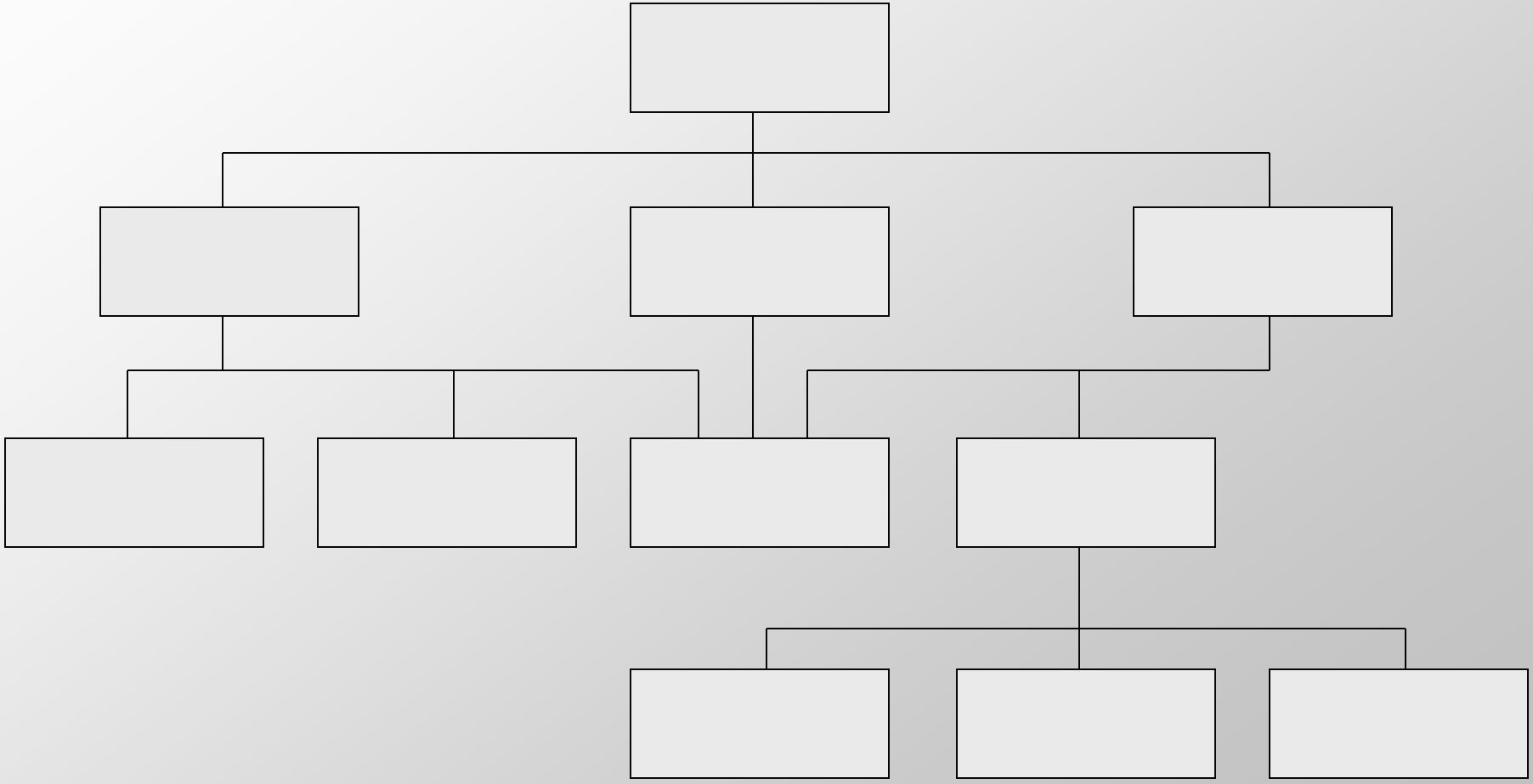
Hierarchisches Datenmodell



Netzwerk Datenmodell

- Erweiterung des Hierarchischen Modells
- Mehrere Vorgänger möglich, daher Mehrfachspeicherung vermeidbar

Netzwerk Datenmodell



Relationales Datenmodell

- Darstellung als Tabellen mit Zeilen und Spalten
- Beziehungen zwischen Tabellen durch
  - Primärschlüssel
  - Fremdschlüssel

Objektrelationale Datenbank

- Bindeglied zwischen klassischen relationalen Datenbanken und Objektdatenbanken.
- Über eine Relationale Datenbank wird eine objektorientierte Zugriffsschicht gesetzt.
- Einsatz: wo Mengen von Objekten in Beziehung zu anderen Daten oder Objekten gebracht werden müssen. Z.B.: Systeme zur Erfassung geographischer Daten (GIS)

Objektorientiertes Datenmodell

- Objekte bestehen aus
  - tabellenähnlich organisierten Daten
  - Methoden zur Veränderung der Daten
  - Objekten
- Vererbung möglich
- Polymorphismus (Methode ist kontextbezogen definiert) möglich

Schritte der Datenmodellierung

- Externe Phase:
  Anforderungsanalyse
- Konzeptionelle Phase:
  Daten und Beziehungen – ER-Modell
- Logische Phase:
  Relationales Datenmodell
- Physische Phase:
  Implementieren
- Test und Abnahme

Entity-Relationship Modell

- Modellierungswerkzeug für Datenbanken
- Darstellung von Daten und Beziehungen auf konzeptioneller Ebene
- Entity
- Entitytyp
- Properties
- Relationships

ERM: Entity

- Abbild eines realen Gegenstandes
  z.B. Kunde, Artikel, etc.
- Eigentliche Daten (Datensätze)
- Wird im ERM nicht dargestellt

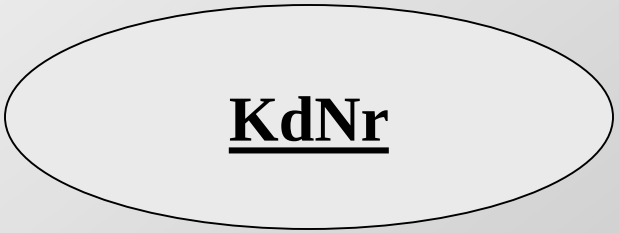
ERM: Entitytyp

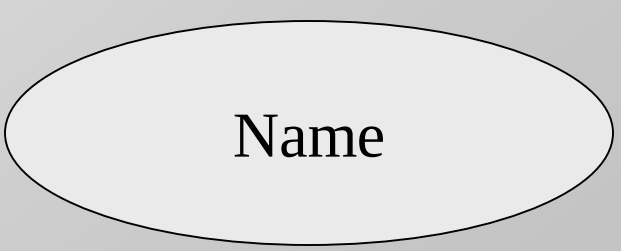
- Gleichartige Entities bilden Entitytyp
  Z.B. Müller, Meier, Schultz bilden den Entitytyp ‚Kunde‘
- Darstellung im ERM als Rechteck (Chen-Notation)

ERM: Attribute (Properties)

beschreiben die Entitytypen näher durch

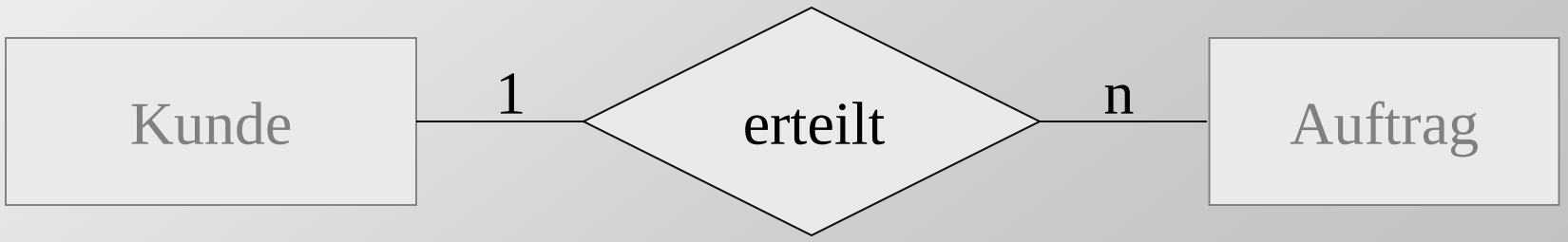
- Namen und Datentyp
- Darstellung im ERM als Ellipse
- Schlüsselattribute werden fett oder unterstrichen





ERM: Beziehungen

- Unterschiedliche Komplexitäten (Kardinalitäten) möglich
  - 1 : 1
  - 1 : n
  - m : n
- Darstellung durch Raute und Verbindungen zwischen den Entitytypen



Begriffserklärung

- Tabelle, Spalte, Zeile, Feld
- Schlüssel
- Redundanz
- Inkonsistenz, Anomalien
- Referenzielle Integrität

Tabelle (Relation)

besteht aus:

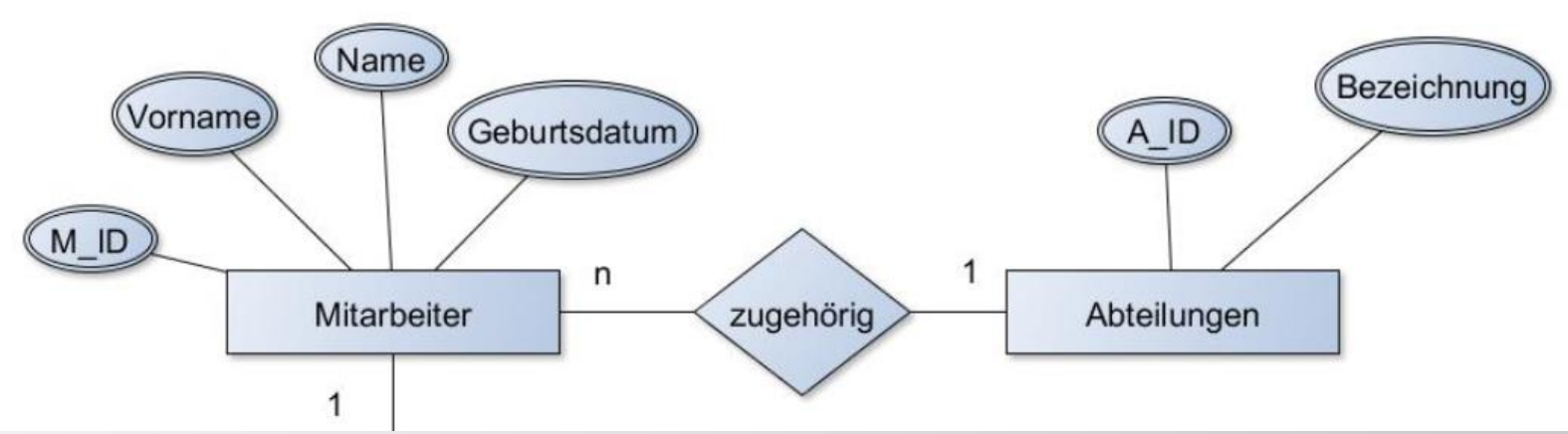
- Zeilen – Datensätze, Tupel
- Spalten – Attribut, Daten mit einem festgelegten Typ
- Spaltenüberschriften – Merkmale, Attribute
- Feld – Attributwert

Schlüssel

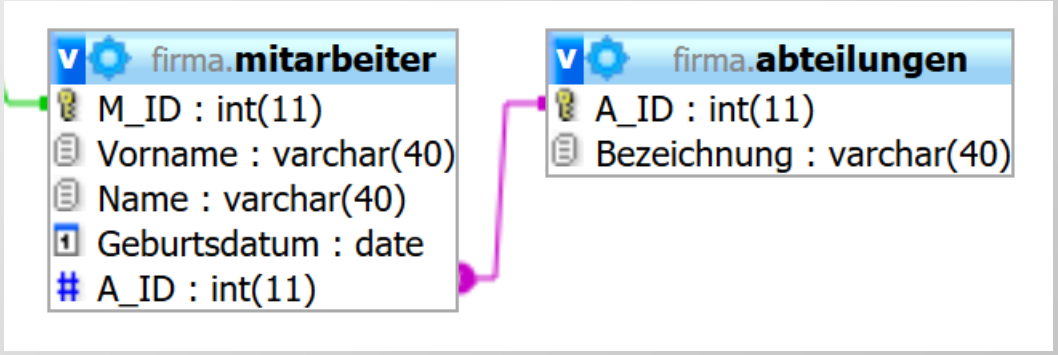
- Primärschlüssel: Attribut zur eindeutigen Identifizierung eines Tupels
- Fremdschlüssel: verweist auf den Primärschlüssel in einer anderen Tabelle
- Kein Schlüssel:
  Index einer Spalte beschleunigt die Suche

1:n-Beziehnung

ER-Modell

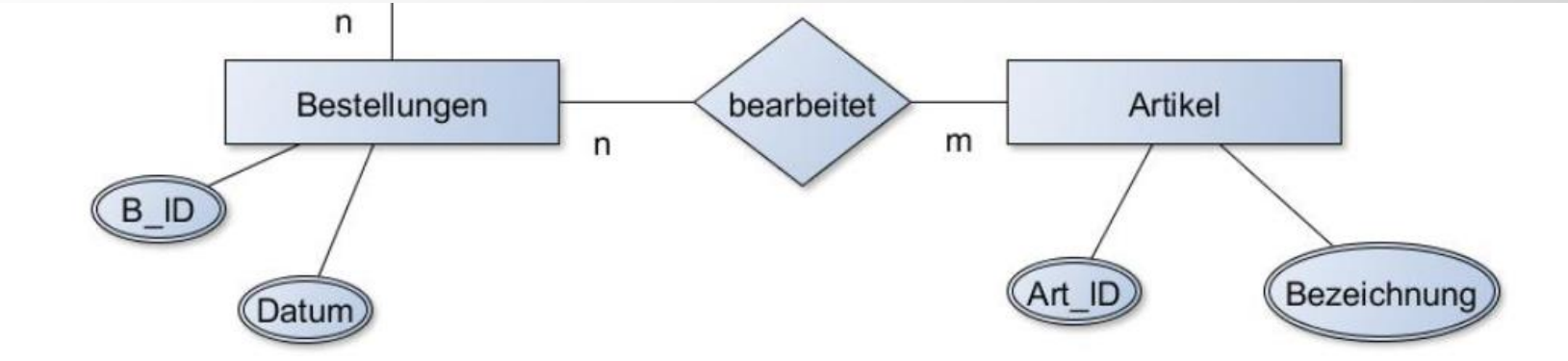


Relationales Datenmodell

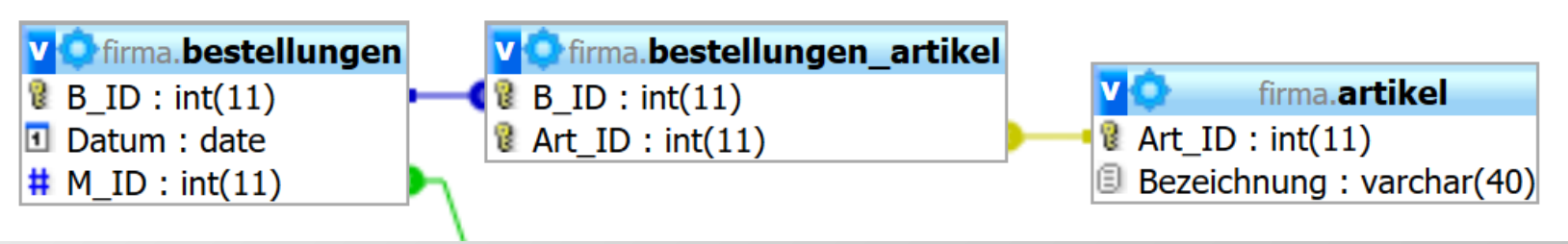


m:n-Beziehung

ER-Modell

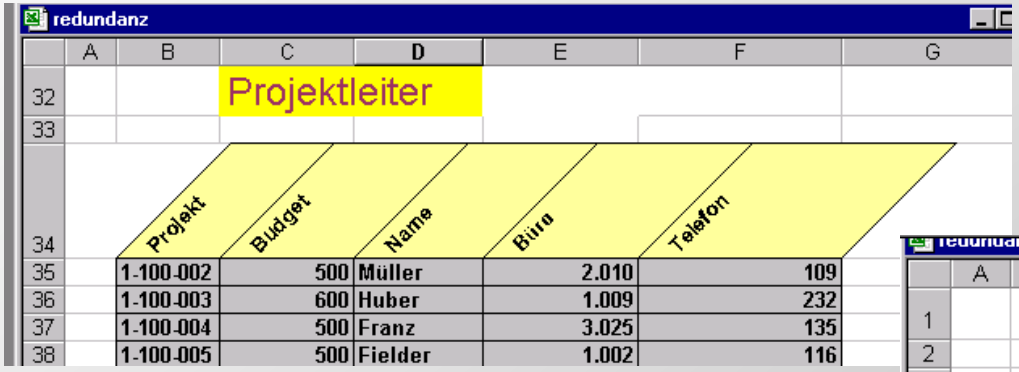


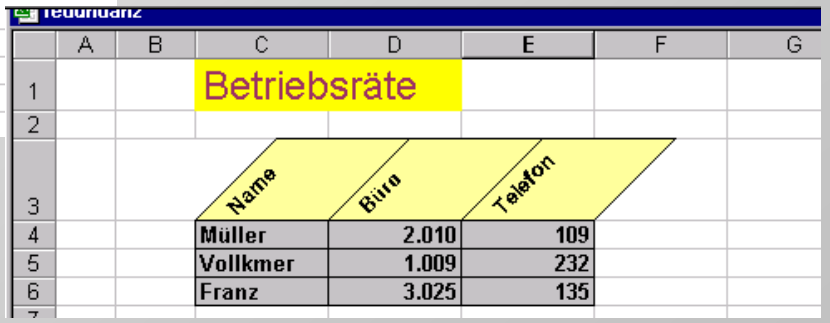
Relationales Datenmodell



Redundanz

- Mehrfache, überflüssige Speicherung eines Attributes, auch Prozessdaten
- Nachteile:
  - Speicherbedarf
  - Änderungen an mehreren Stelle nötig



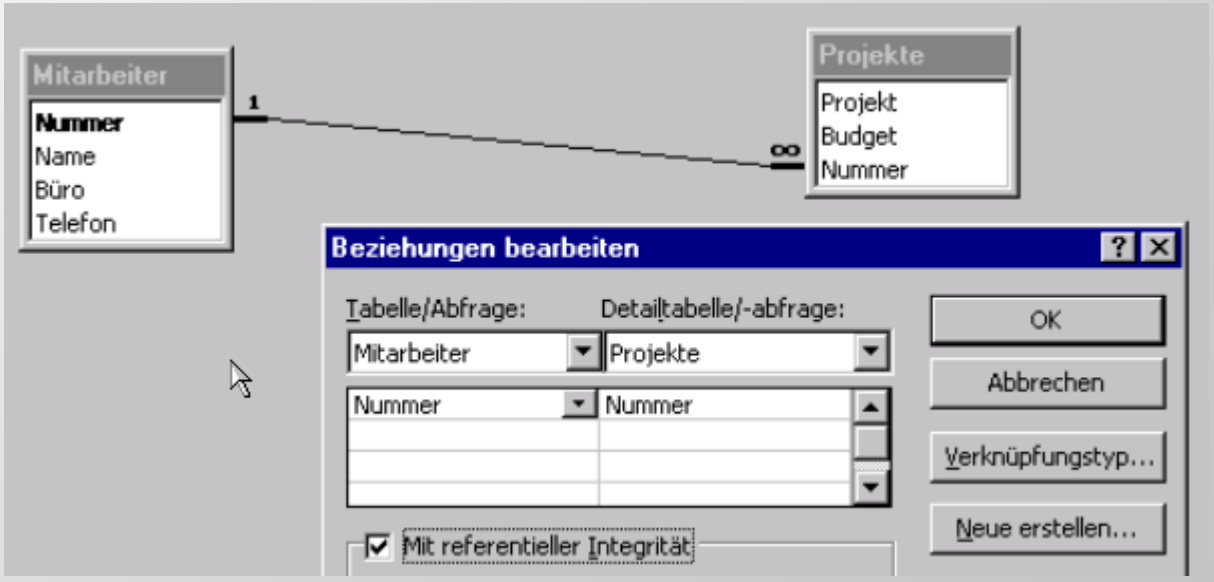


Inkonsistenz, Anomalien

- Mutationsanomalien
  - Einfügeanomalie
  - Änderungsanomalie
  - Löschanomalie
- Unterschiedliche Werte für ein Attribut
- entsteht durch Redundanz, wenn nur an einer Stelle geändert wird
- Kann durch Normalisierung vermieden werden

Referenzielle Integrität

- Trägt zur Datenkonsistenz bei
- Zu jedem Fremdschlüssel muss der passende Primärschlüssel existieren



Normalisieren

- Grund:
  - Vermeidung von Redundanzen
  - Vermeidung von Anomalien
- Normalformen nach Boyce-Codd
  1. Atomare Werte
  2. Jedes Nicht-Schlüsselattribut ist vom gesamten Schlüssel abhängig
  3. Nicht-Schlüsselattribute sind voneinander unabhängig

Unnormalisiert

MiNr	**Name**	**AbtNr**	**Abteilung**	**Projekte**
1	Egon	42	UML	7(DB),21(BMW),3(SI)
4	Kurt	42	UML	1(BK),21(BMW),7(DB)
9	Horst	44	GUI	3(SI),14(C#)

1. Normalform

Alle Attribute enthalten nur einen Wert

MiNr	**Name**	**AbtNr**	**Abteilung**	**ProNr**	**Projekt**
1	Egon	42	UML	7	DB
1	Egon	42	UML	21	BMW
1	Egon	42	UML	3	SI
4	Kurt	42	UML	1	BK
4	Kurt	42	UML	7	DB
4	Kurt	42	UML	21	BMW
9	Horst	44	GUI	3	SI
9	Horst	44	GUI	14	C#

2. Normalform

Jedes Nicht-Schlüsselattribut ist vom gesamten Schlüssel abhängig

MiNr	Name	AbtNr	Abteilung
1	Egon	42	UML
4	Kurt	42	UML
9	Horst	44	GUI

ProNr	Projekt
1	BK
3	SI
7	DB
14	C#
21	BMW

MiNr	ProNr
1	3
1	7
1	21
4	1
4	7
4	21
9	3
9	14

3. Normalform

Nicht-Schlüsselattribute sind voneinander unabhängig

MiNr	**Name**	**AbtNr**
1	Egon	42
4	Kurt	42
9	Horst	44

AbtNr	**Abteilung**
42	UML
44	GUI

Höhere Normalformen

- Selten verwendet und meist nicht sinnvoll, da
  - Unübersichtliche Datenbanken
  - Nachteile für Performance