

Zusammenfassung

Datenbanken management

Themen:

Einführung DB
→ bis

Normalformen

nicht enthalten sind:

- Codd'sche Regeln



Datenmanagement

- Traditionelle DB
 - ↳ Text, Zahlenform - Informationen
- Multimedialer DB
 - ↳ Bilder, Videos, Audio, ...
- Geo. Infosys. (GIS)
 - ↳ geobasierte Daten
- Online Analytical Processing
 - ↳ Data, warehouse, OLAP - Anal., Entscheidungsprozesse
- Echtzeit - akt. DB Systeme
 - ↳ Industrie, Fertigungsprozesse
- Webbereich
 - ↳ Onlineshop

Ansätze Datenspeicherung

- Dateibasiert
- Datenbank

DBS ... Datenbanksysteme

DMMS ... Datenbankmanagementsysteme

Datenbank

Sammlung logisch verwandter Daten

Daten

bekannte Fakten

Miniwelt

Ausschnitt realer Welt über die Daten in die Datenbank gespeichert werden.

DBMS

- kontrollierter Zugriff auf Daten
- Softwaresystem (erstellen, def., pflegen)
- einh. Beschr. d. Datenbank
- Korrektheit der Daten
- Korrektheit im Mehrbenutzerbetrieb
- Datensicherheit

DBS

$$\text{DBMS} + \text{DB} = \text{DBS}$$

- externe persistente Speichermedien
- Daten (einfügen, verändern, löschen, erstellen) nur über DBMS
- Struktur, Op., Konsistenzregeln durch Datenmodell beschrieben
- Beschr. Daten \rightarrow Data Dictionary

Programm - Daten - Unabhängigkeit

- Mehrere Ebenen
 - \hookrightarrow interne & externe Sicht auf Daten
- Datenabstraktion: Benutzer sieht nur extern
 \hookrightarrow intern kann sich ändern solang extern gleich bleibt
- Anwendungen bleiben unberührt von Def. etc.

Anforderungen DB

- Datenunabhängigkeit

↳ interne Speicherstruktur

- Datenunabhängigkeit mit Hilfe

↳ Data Definition Language

↳ Data Manipulation ———

↳ Data Control ———

- Interaktive Benutzeroberfläche

↳ für Daten Def. & Daten man.

- Programmschnittstelle

↳ ermöglicht Daten Def. & Daten von. aus Programm

- Mehrbenutzerfähigkeit

↳ paralleler Zugriff mehrerer Benutzer

- Dauerhaftigkeit

↳ Daten aufbewahren bis lesen

↳ Lebensdauer durch Benutzer bestimmt

↳ ——— unabhängig

↳ Daten dürfen nicht verloren gehen

- Integrität

↳ Daten korrekt

↳ fähig sein falsche oder widersprüchliche Daten zu erkennen

- Vermeidung redundanter Daten

↳ nicht gleiche Daten speichern

- Anwendungsunabhängig

↳ verwendbar sein ohne Wissen über Entwicklungsuntergrund

↳ Angaben für Beschreibung => Änderung inneren

Datenstruktur ohne Anwendungsprogramme umschreiben zu müssen

- Flexibilität

- ↳ einfügen, verändern, löschen, aufsuchen alter Daten
- ↳ Auswahl nach beliebigen Kriterien
- ↳ Sprach-schnittstelle unabhängig von int. Struktur & Speicher

- Größe

- ↳ nicht begrenzt außer physisch
- ↳ nicht durch Hauptspeicher begrenzt

- Mehrfachbenutzbarkeit

- ↳ mehrere Benutzer zur selben Zeit
- ↳ keine Wechselwirkungen

Aufgaben DBS

- Integration

- ↳ einh. Verwaltung aller von Anwendungen benötigte Daten

- Operationen

- ↳ Speicherung, Änderung, Suche im Datenbestand

- Data Dictionary

- ↳ Datenbeschreibung der DB

- Benutzersicht

- ↳ unterschiedliche Anw. unterschiedliche Sichten auf Datenbestand

- Konsistenzüberwachung, Integritäts sicherung

- ↳ Korrektheit d. Daten
- ↳ korrekte Ausführung von Änderungen

- Zugriffskontrolle

- ↳ keine unauthorisierten Zugriffe

- Transaktionen

- ↳ Zusammenfassung von DB-Änderungen zu Funktions-einheiten

- Synchronisation

↳ keine gegenseitige Beeinflussung

- Datensicherung

↳ Wiederherstellen von Daten

Architektur von DB

3 Ebenen



Externe Ebene

↳ UI, API

so wie Benutzer DB sieht

Konzept Ebene

↳ Datenstrukturen, abhängig von DB-Modell

so wie der Designer die DB sieht

Interne Ebene

↳ Einzelheiten Implementierung
Spiegelungen, Indexen

so wie der Rechner die DB sieht

Externe Architektur von DBS

- klassische

Client → Server

Frontend

Backend

- 3 Tier Architektur

Client

Tier 1

Frontend

Application

Tier 2

Applicationserver

Server

Tier 3

Backend

Datenbankbenutzer

- Daten- & Datenbankadmin
 - ↳ Verwaltung DB
- DB- Designer
 - ↳ Entwurf DB
- Anw. programmierer
 - ↳ Entwicklung App.
- Endbenutzer
 - ↳ Abfragen & Manip. Daten

Aufgaben DB- Admin

- Design , Software , Speicherverw. , Implementierung
Sicherheit , Laden v. Daten , Backup , Regenesisieren
von Datenbeständen , Systembeobachtung

Aufgaben DB- Designer

- Systemarch.
 - ↳ Entwurf DB
 - ↳ Aufgabenbeschreibung
 - ↳ Abstimmen bestehender Apps & Einbindung in komplex. Datenmodell
- Ad-hoc- Abfragen
 - ↳ Beobachtung DB & Datenmenge
 - ↳ Frontentwerfen Benutzerfragen
 - ↳ SQL

Aufgaben App- Entwickler

- Entwicklung Masken , Reports
- Erstellen DB-Prozeduren & DB-Triggern
- Anbindung DB an Internet

Aufgaben DB-Benutzer

- Benutzen von Programmen & Werkzeugen

Relationale Datenbanken

- Speichern der Daten in Tabellen mit Beziehungen

↳ Entität → Tabelle

Atribut → Spalte

Relation → Beziehung

Verwaltet mehrere DBs die meistens unabhängig von einander sind.

Tabellenaufbau

Atribut identifizieren die einzelnen Zeilen

der Tabelle eindeutig. → **PK = (Primary key)**

PK darf keine Nullwerte enthalten

○ --- Wert der Spalte nicht verfügbar oder nicht zugewiesen.

Beziehungen zw. Tabellen

1:1 → 1 Datensatz zugeordnet 1 Datensatz

1:n → ————— || ————— mehreren Datensätzen

m:n → ————— || ————— und anders herum

direkte m:n Beziehungen können

in RDBMS nicht abgebildet werden.

Schlüssel

- primary key

↳ jeder pk ist eindeutig zuordenbar

↳ Datensatz wird dadurch eindeutig

↳ jede Tabelle braucht pk

- foreign key

↳ stellt Beziehungen zu anderen Tabellen her

↳ fks sind gewöhnlich nicht eindeutig.

- composite key

↳ bei Tabellen die Beziehungen zw. anderen Tabellen bezeichnen

↳ besteht aus 2 oder -eh fks die die Fkt. eines pks übernehmen.

↳ zur Nat. Pk einfügen

Datenbankentwicklung

Analyse

→ Pflichtheft / Lastenheft



Design

→ log & phys. Datenmodell



Implementation → SQL-Skript

Einführung → Befüllen DB



Wartung

→ Import / Export Daten, Wartung

- Datenanalyse

↳ Sachverhalt

- ERM

↳ graph. Präs. der Objekte & ihrer Beziehungen

- RDB

↳ Umsetzung Entität-Bez.-Modell in Tabellen

- Normalformen

↳ Regeln Qualitätssicherung

- Erstellen & Befüllen

↳ befüllen der DB

Datenanalyse

↳ Sachverhalte in Textform

↳ Daraus Entitäten & Beziehungen ableiten

Datenbankentwurf

↳ ERM

↳ UML (⇒ Unified Modeling Language)

↳ objektorient. Modell

↳ auch für RDB geeignet

↳ weitere Modelle

ERM

→ Entitätsmenge

→ Attribut

→ Beziehungsmenge

RDB

↳ aus ERM kann ein RDB entwickelt werden

Normalform

↳ zur Überprüfung der Tabellen nach Qualitätskriterien

↳ bewirkt Verlust von Daten von größeren Tabellen

Implementation

↳ erstellt & Befüllung der DB mit SQL

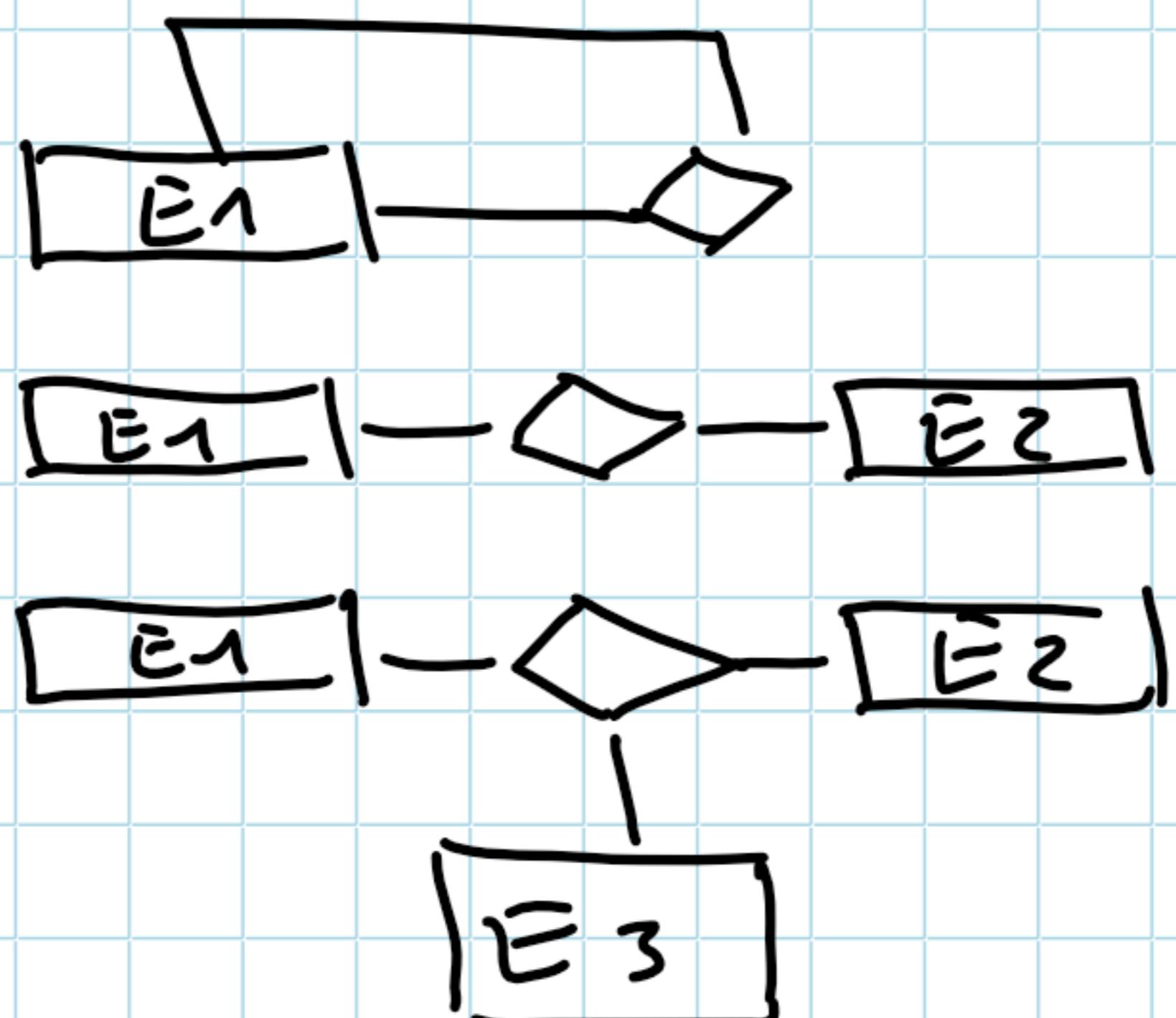
ERM (nach CHEN)

Beziehungen

- 1 → 1 Angestellter arbeitet für 1 Projekt
- c → 1 oder kein Angestellter — " —
- m → min. 1 oder mehrere — " —
- cm → 1, kein oder mehrere — " —

Beziehungsarten

- unär / rekursiv
- binär
- ternär



identifizierendes Attribut

- ist das Attribut welches Entität eindeutig unterscheidbar macht → ist aus gleich meistens der pk.

zusammengesetztes Attribut

LDS sind weiter aufgeteilte Attribute quasi zusammengestückt zum atoren Attribut

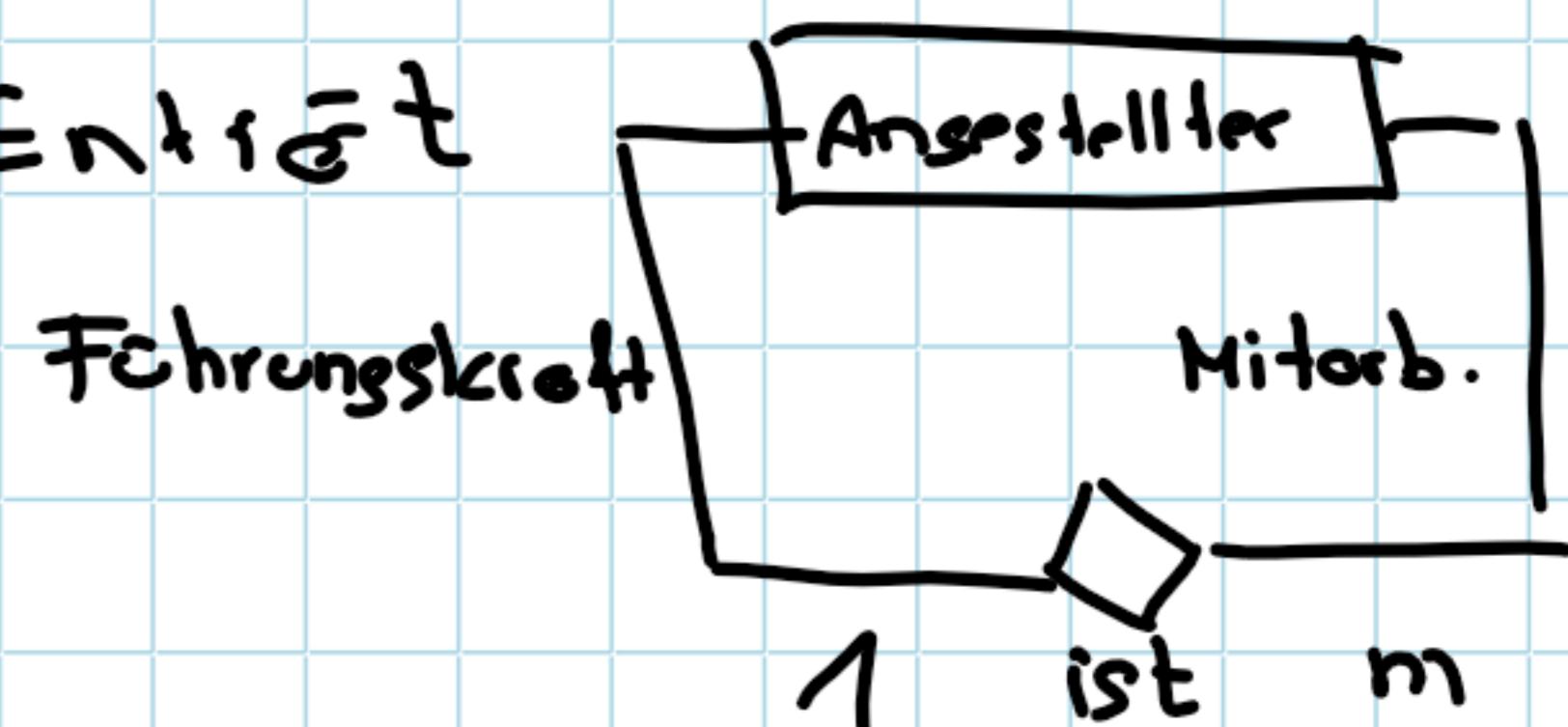
mehrwertiges Attribut

↳ Attribut ist dann mehrwertig wenn mehrere Werte haben kann z.B.: Titel

EERM

Reflexive Beziehung

↳ Beziehung zu einer oder der selben Entität



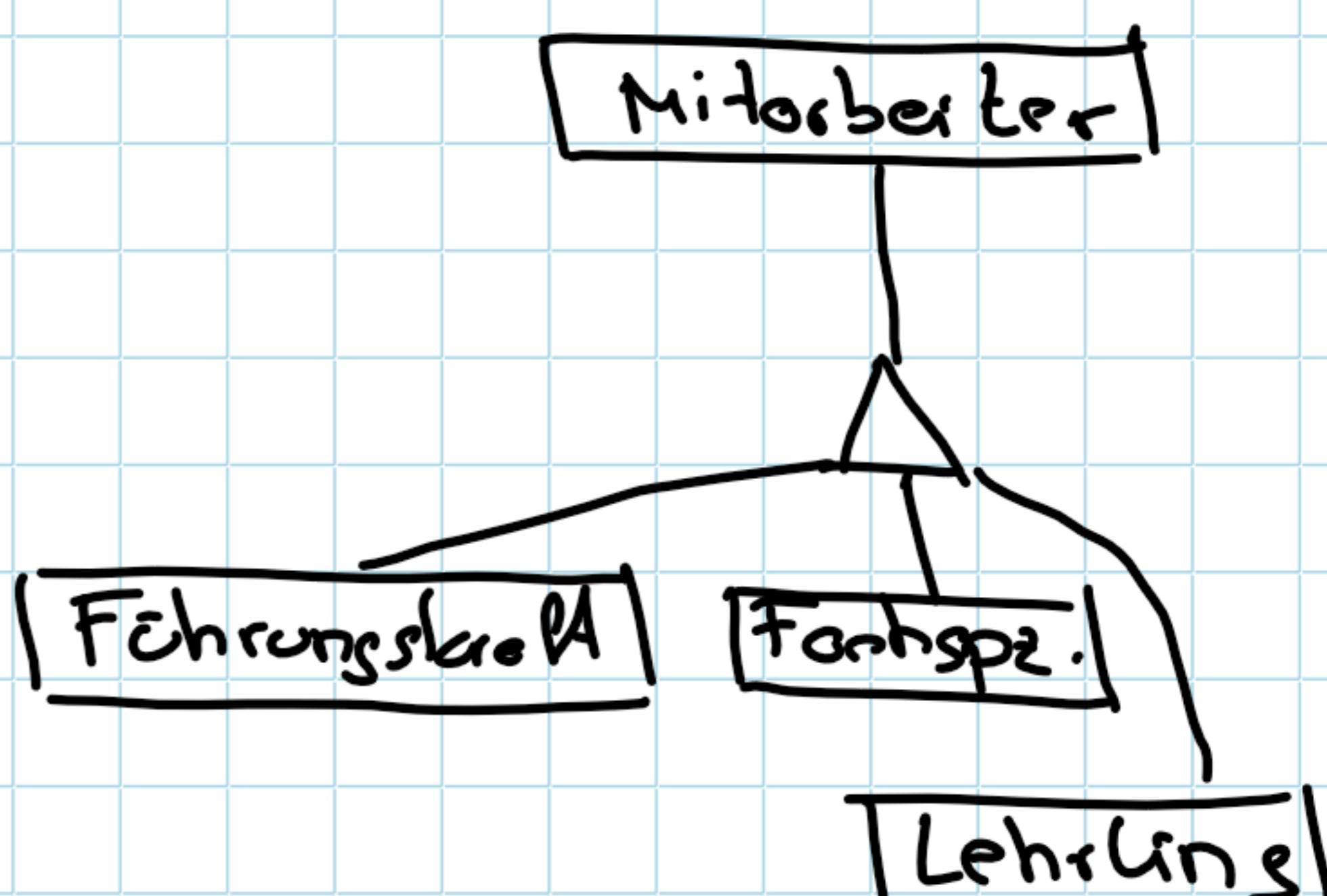
Vererbung

↳ Erweiterung Richtung Objektorientierung

↳ Gemeinsamkeiten werden in übergeordneten EntitätsTyp unterteilt

- 4 Möglichkeiten

1 - überlappend - unvollständig



2 - " " - vollständig

3 - disjunkt - unvollständig

4 - " " - vollständig

Bsp.:

1 überlappend

Mitarbeiter kann mehrere

2 disjunkt

Mitarbeiter kann nur ein

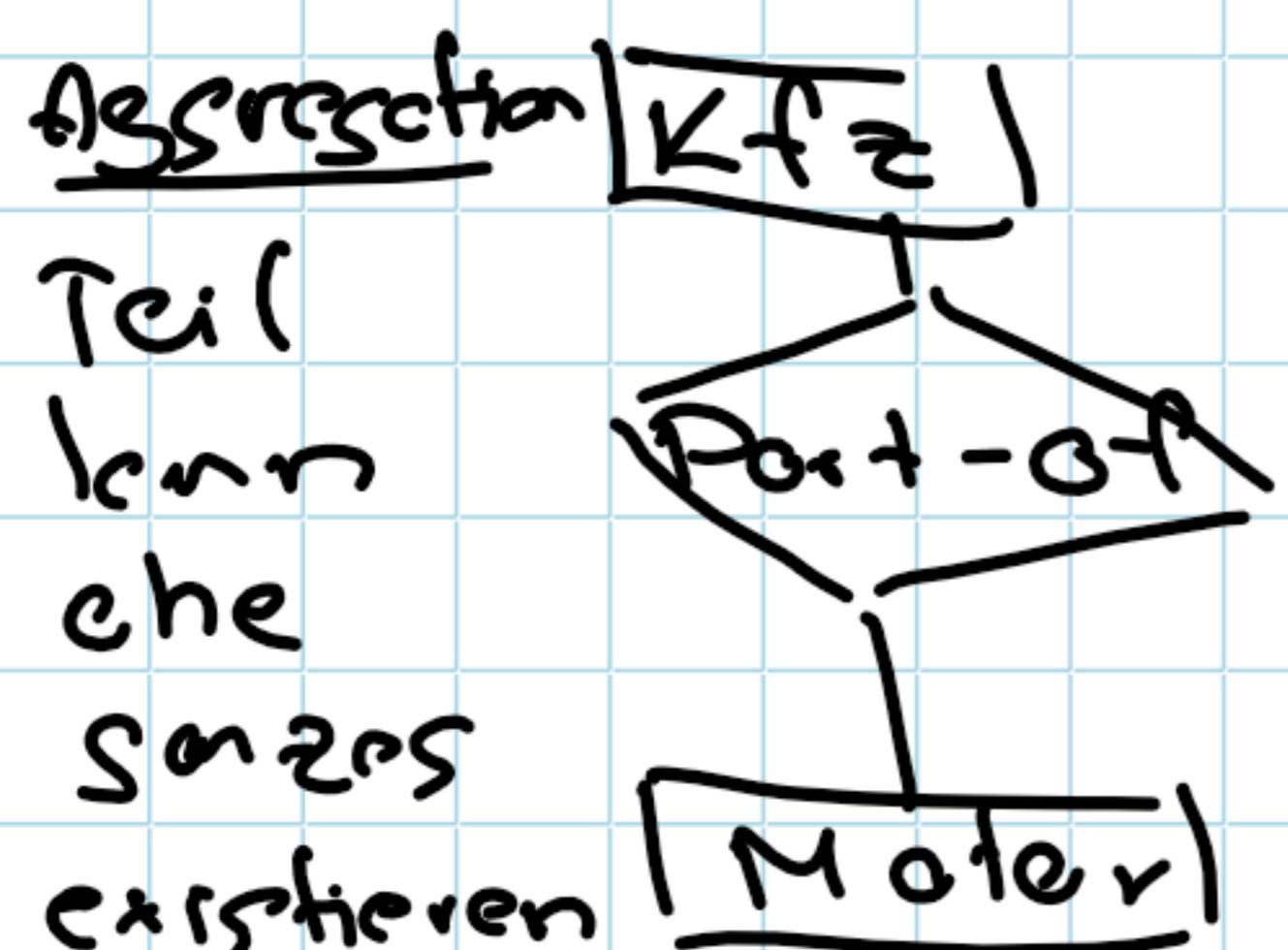
3 vollständig

jeder Mitarbeiter zuordnet einem

4) unvollständig

kein Mitarbeiter die keine

Aggregation, Komposition



Komposition:

Teil kann nicht ohne Sitzes existieren

Überführung ERM nach Tabellen

Regel 1

- ↳ Jede Entität → eigenständige Tabelle
- ↳ Merkmale Entität → Attribute (Spalten)
- ↳ Ein PK ist vorhanden
- ↳ Entitätsname = Tabellename
- ↳ gibt es einen Schlüssel wird dieser zum PK
- ↳ Eignet sich kein Attribut als PK wird ein neues hinzugefügt und zu PK (z.B. ID)

Regel 2

- ↳ Jede Beziehungsmenge → eigenständige Tabelle
- ↳ PK ist oft die Kombination der FK der zugehörigen Entitätsmenge
- ↳ Merkmale → Attribute (Spalten)
- ↳ eigene Tabelle oft nicht ideal

Regel 3

- ↳ Komplex-Komplex-Beziehungen müssen als eigenständige Tabelle definiert werden.
- ↳ PK ist Kombi aus beiden FKs oder eines Schlüsselkandidat

Regel 4

- ↳ Einfach-komplex-Beziehungen können ohne eigenständige Tabelle definiert werden
- ↳ in der Praxis empfohlen
- ↳ der Entitätsentabelle wird ein FK auf andere ET zusammen mit eventuellen weiteren Merkmalen der BM hinzugefügt.

Regel 5

- ↳ Einfach-Einfach-Beziehungen kann ohne eigenständige Beziehungstabelle definiert werden
- ↳ normalerweise bekannt c-Seite FK auf andere Entitätsentabelle
- ↳ wenn auf beiden Seiten der BM Kardinalität dann frei entscheidbar wo FK eingefügt wird.

Regel 6

↳ Generalisationen

- ↳ jede Entität = eigene Tabelle
- ↳ PK der übergeordneten Entitäten = PK der untergeordneten Entitäten (= Vererbung)
- ↳ Disjunktion sollte im ERM nachvollzogen werden
z.B. einfügen eines Merkmals in übergeordnete Tabelle

Integrität, Anomalie

wann in relat. DB-Modell Tabellen hochwertig

- ↳ Sachverhalt exakt dargestellt
- ↳ möglichst fehlerfreie Operationen

Qualitätsmessung

- ↳ auf log & Konzept Ebene
- ↳ auf Implementierungs- & Speicherebene

Qualitätsmaße

↳ 3 informelle Qualitätsmaße

- Semantik der Attribute
 - ↳ Bedeutung d. Attribute
 - ↳ gibt es Beziehungen zw. Attributen innerhalb einer Tabel (Abhängigkeit?)
- Reduzieren der redundanten Werte in Tupeln
 - ↳ Daten einer Tabels öft gespeichert
=> unerwünschte Nebeneffekte
- Reduzieren der Nullwerte in Tupeln
 - ↳ viele 0 = viel Speicher benötigt

Unerwünschte Nebeneffekte

- ↳ Abhängigkeiten & redundante Informationen
=> häufig Probleme
- ↳ Redundanz = viele Daten mehrfach gespeichert

Anomalien

- Änderungs - Anomalie
 - ↳ Beim ändern von Einträgen kann es zu Tipp-Fehlern kommen
 - ↳ Änderung eines Attribut → Änderung in mehreren Zeilen.
- Lösch - Anomalie
 - ↳ werden alle Einträge zu einer Abteilung gelöscht → Abteilung auch gelöscht
- Einfügs - Anomalie
 - ↳ Ich kann keine Abteilung einfügen wenn ich noch keine Angestellten eingetragene habe.

Folgen einer Anomalie

- ↳ Fehleranfälliger
- ↳ Erhöhter Speicherbedarf
- ↳ Leistungseinbußen

Strukturelle Integritätsbedingungen

Integrität bzw. Konsistenz => Widerspruchsfrei von Datenbeständen

- ↳ Daten fehlerfrei erfasst → Informationsgescholt kann wiedergegeben werden
- ↳ verletzt wenn Mehrdeutigkeiten oder widersprüchliche Sachverhalte auftreten.

strukturelle Integritätsbedingungen in RDB

- Teil 1

- Eindeutigkeitsbedingung

↳ jede Tabelle hat einen PK der Tabelle eindeutig erkennbar macht.

- Wertebereichsbedingung

↳ Merkmale in einer Tabelle können nur Werte in vorsezogenem Wertebereich annehmen.

- Teil 2

- referentielle Integritätsbedingung

jeder Wert eines FK muss effektiv als Schlüsselwert in referenzieller Tabelle existieren.

- Ablaufintegrität

↳ mehrere Benutzer können konkurrierend auf DB zugreifen

DB muss sich danach in korrekten Zustand befinden.

Relationenalgebra & Normalformen

- Relationenalgebra

X bestimmt funktional Y

Bsp.:

Y funktional abhängig von X

$$\{A\} \rightarrow \{B\}$$

a ₁	b ₁
a ₂	b ₂
a ₃	b ₂
a ₄	b ₂

R			
A	B	C	D
a ₄	b ₂	c ₄	d ₃
a ₁	b ₁	c ₁	d ₁
a ₁	b ₁	c ₁	d ₂
a ₂	b ₂	c ₃	d ₂
a ₃	b ₂	c ₄	d ₃

$$\{C, D\} \rightarrow \{B\}$$

c ₄ d ₃	b ₂
c ₁ d ₁	b ₁
c ₃ d ₂	b ₂

$$\{B\} \rightarrow \{C\}$$

b ₂	c ₄
b ₁	c ₁
b ₂	c ₃

nicht erlaubt X

$$\{A, B\} \rightarrow \{C\}$$

a ₄ b ₂	c ₄
a ₃ b ₂	c ₄
a ₁ b ₁	c ₁
a ₂ b ₂	c ₃

$$\{B, C\} \rightarrow \{A\}$$

b ₂ c ₄	a ₃
b ₁ c ₁	a ₁
b ₁ c ₁	a ₂
b ₂ c ₃	a ₂

nicht erlaubt X

$$\{B\} \rightarrow \{A\}$$

b ₂	a ₄
b ₁	a ₁
b ₂	a ₂
b ₂	a ₃

nicht erlaubt X

Begriffs festlegung

$X \rightarrow Y$

volle funktionale Abhängigkeit ... kein Attribut aus X

kann entfernt werden

ohne die fkt. Ab-

hängigkeit zu zerstören

Hier ist X ein Schlüssel

\Rightarrow Schlüssel ist Attributmenge für Relation R

wenn: Y aus R gilt = $X \rightarrow Y$ d.h. $X \rightarrow R$

alle Attribute sind voll funktional abhängig von X

Da Attributmenge X aus R nicht eindeutig sein muss

\rightarrow alle zutreffende X_i = Schlüsselkandidaten.

Beispiel:

$$R_A = \{A, B, C, D, E\}$$

$$R_E = \{A \rightarrow B, AC \rightarrow D, C \rightarrow E, E \rightarrow C\}$$

wir suchen Schlüsselkandidaten bedeutet welche Kombination

- $A \rightarrow B$

Kann alle anderen abbilden.

- $AC \rightarrow D$

- $AE \rightarrow BC \rightarrow D$

- $AC \rightarrow BD$

Hier ist es AC & AE

- $C \rightarrow E$

weil wir mit ihnen alle

- $AC \rightarrow BDE$

anderen abbilden können.

- $E \rightarrow C$

(A, B, C, D, E)

- $AE \rightarrow BC$

Normalformen

Ziel ist Vermeidung von Redundanzen & Anomalien durch Aufspaltung von Relationschemata.

Es gilt:

- es dürfen semantische Informationen nicht verloren gehen
- Rekonstruktion der Relationen muss gewährleistet sein
- **1. Normalform** (nur elementare Merkmalswerte erlaubt)

↳ Mengen, Folgen & Arrays sind nicht elementar

Eine Relation ist in der 1. Normalform, wenn die Werte der Attribute atomar (elementar) sind.

elementar -.. keine Komponenten die getrennt verarbeitet werden können.

- **2. Normalform** (abhängig von allen Schlüsselmerkmalen)

↳ volle fkt. Abhängigkeit nur dann, wenn Zusammengesetzter Schlüssel allein in seiner Gesamtheit die restlichen Nichtschlüsselattribute eindeutig bestimmt.

↳ besitzt Relation nur einen Schlüssel & erfüllt 1NF
erfüllt sie auch 2NF.

↳ Verletzung von 2NF nur dann wenn Schlüssel mehr als 1 Attribut hat

- 3. Normalform (nicht abhängig über Umwege)

↳ indirekte Abhängigkeiten untersuchen

Über die 3. Normalform hinaus

- 4. Normalform (Mehrwertabhängigkeiten eliminieren)
- 5. Normalform (garantiert größtmögliche Auflösung von Tabellen ohne Informationsverlust)