TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik | Institut Systemarchitektur | Professur Rechnernetze

WS 2012
LV Informatik-I für Verkehrsingenieure

6.1 Datenbanksysteme

Dr. rer.nat. D. Gütter

Mail: Dietbert.Guetter@tu-dresden.de
WWW: wwwpub.zih.tu-dresden.de/~guetter/

Datenhaltung, Datenlokalisierung

Problem der Programmierungstechnik

- Wie sollen die Daten verarbeitet werden (Algorithmen, s. Kap. 3)?

Weitere Probleme

- Wie sind die Daten sinnvoll zu strukturieren und
  an welchen Speicherpositionen sind die Daten zu lokalisieren?
- Wie kann eine flexible Datennutzung erreicht werden
  (unterschiedliche Anwendungszwecke, bzw. mehrere Nutzer)?

→ **Datenbanksysteme**

  - Entlastung der Programmierer von Routineaufgaben
    des Datenmanagements
  - hohe Effizienz der Nutzung
    (geringer Speicheraufwand, schneller Zugriff)

Klassische Datenhaltung - Karteien

- Notation relevanter Informationen auf **Karteikarten**

  z.B.: beim Arzt pro Patient
  Name, Vorname, Anschrift, Behandlungsdaten

- Sortierte Sammlung von Karteikarten in **Kartei**

  z.B.: Ordnung alphabetisch nach Patientennamen

  Problem: evtl. aufwendige Handsuche
  bei anderen Ordnungskriterien, z.B. Alter, Krankenkasse, …

- Rudimentäre Unterstützung
  der Anzeige weiterer Ordnungsmerkmale

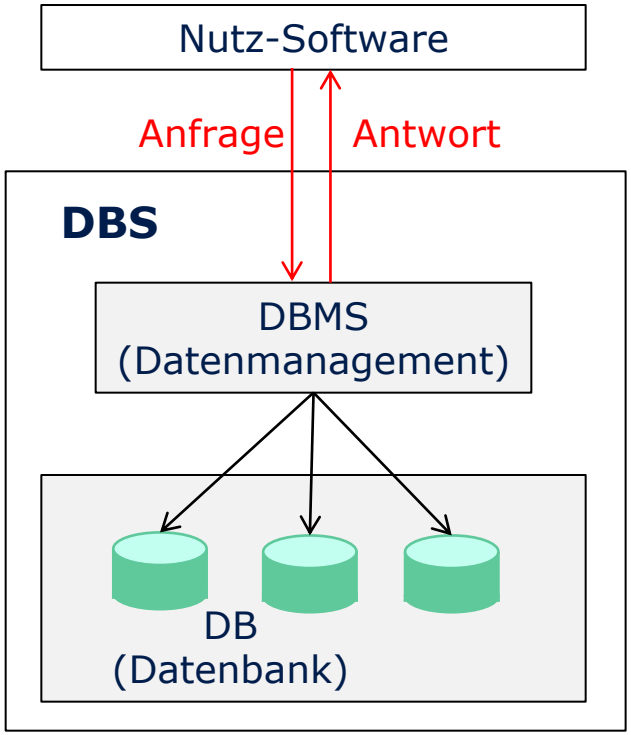
  z.B.
  durch farbige Markierungen (Fähnchen)

http://www.spitta.de/

Datenbanksysteme (DBS)

- bieten Computernutzern eine leistungsfähige Unterstützung bei Problemen der Haltung und Verarbeitung großer Datenmengen
- sind unverzichtbar in Ökonomie, Verwaltung, Wissenschaft, ...

Struktur



Datenbank-Managementsystem

zur Organisation des Datenzugriffs

strukturierte Datenbasis

Nutzen von Datenbanksystemen

zentrales Datenmanagement

- Daten besser strukturiert gegenüber Datenhaltung in Dateien
- weniger Redundanz
- Möglichkeit der flexiblen Datenverknüpfung

Effizienz

- kurze Bearbeitungszeiten
  wegen optimal implementierter Algorithmen für Sortierung, …
- geringer Speicherbedarf
- hohe Zuverlässigkeit

(i.a.) **Unterstützung des Mehrnutzerbetriebes**

- Datensicherheit durch Kontrolle der Zugriffsrechte
- Sicherung der Datenkonsistenz

komfortable Nutzerschnittstelle

DBMS

Aufgaben

- Aufbau und Strukturierung einer Datenbank
- Strukturveränderungen
- Informationsaktualisierung
  (Informationen eingeben, ändern, löschen)
- Anfragen zum Datenbankinhalt
  (Informationen auswählen, lesen, sortieren, verknüpfen)

Nutzungsbeispiel: Mitarbeiterverwaltung eines Betriebes

- Erzeugung von Listen,
  geordnet nach Mitarbeiternamen, Alter, Qualifikation, …
- Erzeugung von verknüpften Listen,
  z.B. Liste der Mitarbeiter mit Führerscheinen C1 und A1
- …

Datenabstraktionsebenen

Sichtebene

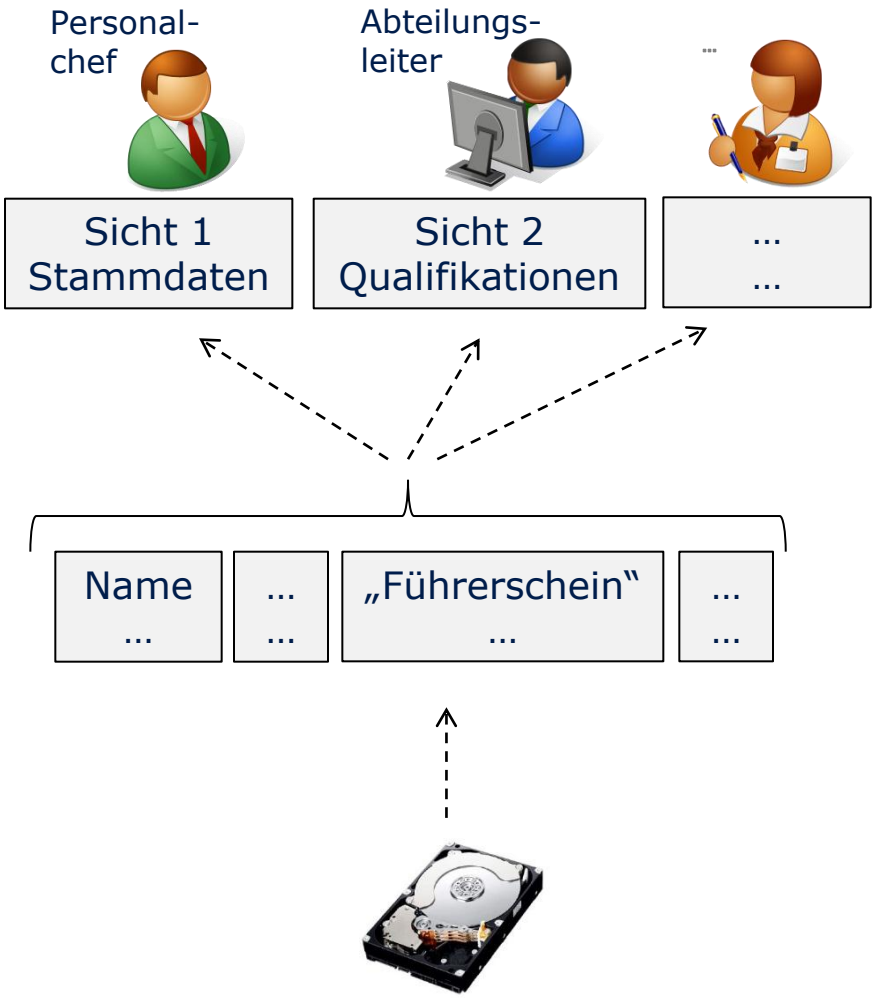
Teilmengen der Daten,
für einen Nutzertyp optimiert

logische/konzeptionelle Ebene

Festlegung
von Datentypen und –strukturen

physische Ebene

Beschreibung
der Datendarstellung in Dateien



Datenbankmodelle

Hierarchisches Datenbankmodell

- Datensätze in Baumstruktur verknüpft
  - realisiert z.B. bei Rechnernetzen (FTAM, MIB)
  - einfache Realisierung in der Sprache XML

Relationale Datenbanken

- 1970 theoretisches Konzept (relationale Algebra) von **Codd**
- vorherrschendes Modell für Datenbanksysteme, z.B. für
  - MS Access, HSQLDB
  - MS SQL-Server, IBM DB2, Oracle Database, MySQL

Objektorientierte Datenbanken

- aktuelle Entwicklungen,
  basieren auf Prinzipien der objektorientierten Programmierung

Relationale Datenbanken vs. klassische Karteien

Kartei	**relationale Datenbank**
Karteikästen, -karten, …	→ Tabellen
Karteikarte hat eine Ordnung (zu erfassende Informationen)	→ Tabellenkopfzeile
ausgefüllte Karteikarte	→ jeweils eine Tabellenzeile
sortierte Kartei	→ Sortierung über Tabellenspalte
aufwendige Handsortierung bei Ordnung nach anderen Merkmalen	→ Neusortierung nicht erforderlich (über Index-Tabellen effizient lösbar)
Verknüpfung von Merkmalen unmöglich mit einfachen Karteikarten (eingeschränkt mit Kerblochkarten)	→ Tabellenverknüpfung durch DBMS effizient möglich

Relationale Datenbanken (2)

Relationen

sind Beziehungen zwischen Attributen und Attributwerten,
darstellbar als **Tabelle** bzw. Verknüpfungen zwischen Tabellen

Spalten: **Attribute**
Spaltenanzahl entspricht Grad der Relation

Zeilen: **Tupel** bzw. Datensätze (evtl. einige Attribute unbelegt)
Zeilenanzahl entspricht Kardinalität der Relation

$Attribut_1$ (Name/Typ der Spaltenelemente)	…	…	$Attribut_n$
$Wert_{11}$	…	…	$Wert_{1n}$
…	…	…	NULL
$Wert_{m1}$	…	NULL	$Wert_{mn}$

Grad
n

Kardinalität
m

Eigenschaften von Relationen

Wertetupel müssen eindeutig sein.
(Alle Tabellenzeilen unterscheiden sich in mindestens einem Attributwert.)

Schlüssel (Menge von Attributen der Relation)
zur eindeutigen Identifikation der Wertetupel

- Superschlüssel: Menge aller möglichen Schlüssel
  - immer möglich: Menge aller Attribute
    (ineffizient für Bearbeitung)
  - oft möglich: weitere Schlüssel mit weniger Attributen
- Schlüsselkandidaten: Teilmenge der Superschlüssel
  mit minimaler Attributanzahl
  (optimal, wenn nur 1 Attribut reicht)
- Primärschlüssel: ausgewählter Schlüsselkandidat

Beispiel

Verwaltung der Mitarbeiter eines Bahn-Depots

Tabelle für alle Mitarbeiter und alle interessierenden Attribute

Mitarbeiter-name	Vorname	Geburts-datum	…	Qualifikation	…	Lokführer-schein	…	…
…	…	…	…	…	…			
Adam	Eva	01.10.92	…	B	…	Klasse 1		
…	…	…	…	…	…			
…	…	…	…	…	…			…
Mayer	Martin	12.11.77	…	C	…	Klasse 3		
Meier	Martin	21.08.91	…	B, E5	…		…	
…	…	…	…	…	…			
…	…	…	…	…	…			
Pech	Paul	31.05.60	…	C	…	Klasse 3		
…	…	…	…	…	…			

Tab_1

Beispiel - Schlüsselwahl

Diskussion

- Attribut „Mitarbeitername“ als Schlüssel

  Mitarbeiternamen sind nicht (garantiert) eindeutig
  (z.B. könnte 2x „Meier“ vorkommen)

- 2 Schlüsselattribute, zusätzlich „Vorname“
  Problem nicht vollständig gelöst, z.B. 2x „Martin Meier“ möglich

- 3 Schlüsselattribute, zusätzlich „Geburtsdatum“
  (ziemlich) sicher

- evtl. 2 Attribute „Mitarbeitername“ und „Geburtsdatum“
  „Mitarbeitername“ und „Anschrift“

…

- optimal Surrogatschlüssel (neues eindeutiges Attribut)

  Primärschlüssel: Attribut „Personalnummer“

Beispiel

Verwaltung der Mitarbeiter eines Bahn-Depots

Tabelle für alle Mitarbeiter und alle interessierenden Attribute

Personal-nummer	Mitarbeiter-name	Vorname	…	Qualifikation	…	Lokführer-schein	…	…
0001	…	…	…	…	…			
0002	Adam	Eva	…	B	…	Klasse 1		
…	…	…	…	…	…			
…	…	…	…	…	…			…
0221	Mayer	Martin	…	C	…	Klasse 3		
0222	Meier	Martin	…	B, E5	…		…	
…	…	…	…	…	…			
…	…	…	…	…	…			
0374	Pech	Paul	…	C	…	Klasse 3		
…	…	…	…	…	…			

Tab_2

Tabellenbereich voll
jeder Mitarbeiter besitzt alle Attribute

Tabellenbereich relativ leer
nicht alle Mitarbeiter besitzen diese Attribute

Verknüpfung von Tabellen (JOIN)

Schlüssel ermöglichen die Kombination von Daten aus mehreren Tabellen z.B.

Personal-nummer	Mitarbeiter-Name	Vorname	...
0001	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

Stammtabelle

Lokführer	Lokführer-schein
0002	Klasse 1
0221	Klasse 3
0374	Klasse 3

Tab_3
Tab_4

Tabelle
Lokführer

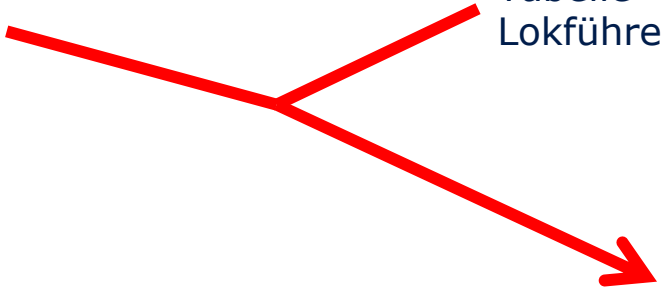


Tabelle
Kandidaten für Weiterbildungskurs

Vor.: Alter unter 50 Jahre
Lokführerschein Klasse 3

Kandidat
0221

Tab_5

Tabellenverknüpfungen (2)

Primärschlüssel besitzt jede Relation (Tabelle)
zur Sicherung der Eindeutigkeit der Tupel

Fremdschlüssel

- zeigen auf einen Schlüssel der fremden Relation und dienen der Verknüpfung der Tupel verschiedener Relationen
- Häufig wird der Primärschlüssel der anderen Relation als Fremdschlüssel benutzt.

  z.B. „Personalnummer" für „Lokführer" bzw. „Kandidat"

Vor- und Nachteile des Nutzens mehrerer Tabellen in einer Datenbank

- Speicherbedarf, Abfrageaufwand und Tabellenwartung sinken
- schwieriger Datenbankentwurf, evtl. Leistungseinbußen

Tabellensortierung

(Vor-)Sortierte Tabellen

- einmaliger Aufwand
- Zugriffsvorteile im Dauerbetrieb

- Vorgehen

  1. Auswahl eines Primärattributes (z.B. „Mitarbeitername“)
  2. Sortierung nach Primärattribut

     - Ordnung: „Alphabet“, „natürliche Zahlen“ oder …
     - Richtung: auf- oder absteigend

  3. Im Falle von Wertgleichheiten beim Primärattribut Sortierung nach weiteren Attributen (z.B. „Vorname“)

Problem Ordnung in einer Spalte bedeutet meist Unordnung in den anderen Spalten

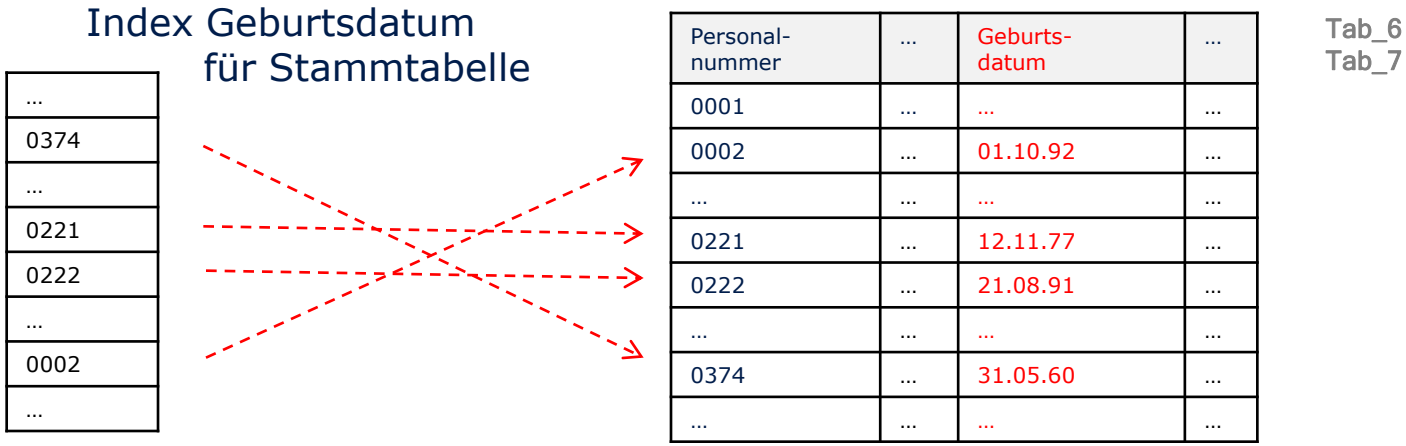
➔ Sekundärschlüssel für Zugriffsindexierung

Indexierung

Index

- Zeiger-Vektor
  Elemente verweisen auf Tabellenzeilen in geordneter Reihenfolge
- Schlüsselattribute (Primärschlüssel, Fremdschlüssel)
  erhalten automatisch einen Index
- Verbesserung der Suchzeiten in Datenbanken,
  aber mehr Aufwand für für Dateneingabe, -änderung, -löschung

Beispiel: Ordnung der Mitarbeiter nach „Alter absteigend"



Tab_6
Tab_7

Atomare und nichtatomare Attribute

Atomare Attribute besitzen in jedem Fall nur einen Attributwert pro Tupel, nichtatomare besitzen mehrere Werte pro Tupel.

- Nichtatomare A. erschweren Datenoperationen (Sortierung, …).

Beispiel: nichtatomares Attribut „Qualifikation"

- evtl. Tabelle umstellen,
  Attribut aufteilen in 2 Attribute „Qual_1" und „Qual_2"
  → unflexibel (weitere Qualifikationen in Zukunft ?)

- evtl. mehrere Tupel in Relation aufnehmen (s. 1. Normalform)
  → alter Primärschlüssel nicht mehr anwendbar (2-Attribut-Schlüssel)

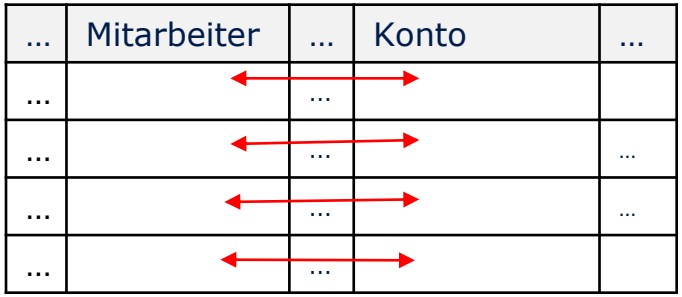
Personal-nummer	Mitarbeiter-name	Vorname	…	Qualifikation	…	Lokführer-schein	…	…
…	…	…	…	…	…			
0222	Meier	Martin	…	B	…		…	
0222	Meier	Martin	…	E5	…		…	
…	…	…	…	…	…			

Tab_8

Beziehungen

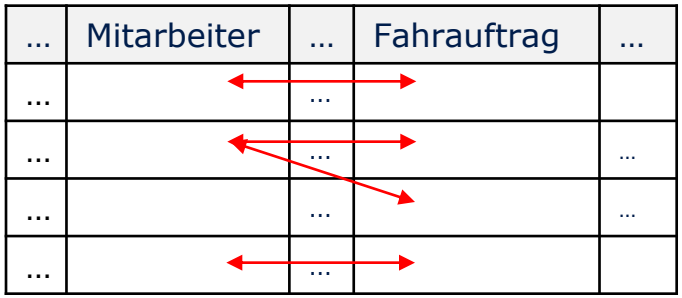
1:1-Beziehung (atomar)

- Jeder MA hat genau ein Konto.



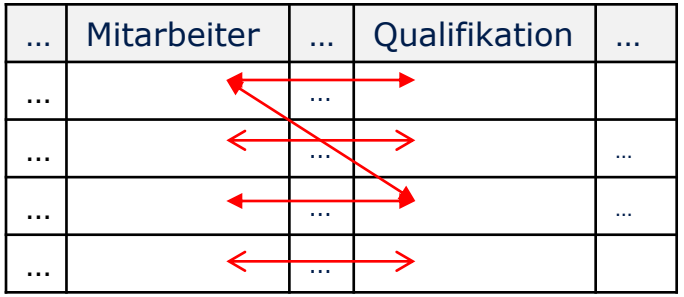
1:n-Beziehung

- Ein Mitarbeiter kann mehrere Fahraufträge ausführen.
- Jeder Fahrauftrag wird immer von genau einem Mitarbeiter ausgeführt.



n:m-Beziehung

- Ein Mitarbeiter kann mehrere Qualifikationen besitzen.
- Eine bestimmte Qualifikation können mehrere Mitarbeiter haben.



Normalisierung relationaler Datenbanken

Problem: Nichtatomare Attributsbeziehungen

- hoher Wartungsaufwand / Gefahr der Konsistenzverletzung

Lfd.Nr.	Personal-nummer	…	Wohnort	…
0001	…	…	…	…
…	0222	…	Dresden	…
…	0222	…	Dresden	…
…	…	…	…	…

Änderungsoperation

„Umzug Martin Meier
von Dresden nach Freital“

Tab_9

- Änderung nur 1x → inkonsistente Datenbank (Datenwidersprüche)
- Änderung in mehreren Tabellenzeilen → steigender Arbeitsaufwand

Normalisierung

Tabellen werden umgeordnet
mit dem Ziel einer besseren Strukturierung der Datenbank

1. Normalform (1. NF)

Bedingungen

- Es existiert ein Primärschlüssel (evtl. über mehrere Attribute).
- Zahl der Tabellenspalten ist fest.
- Alle Tabellenspalten enthalten atomare Inhalte.

ggf. sind Attribute aufzuteilen, z.B. Anschrift in „PLZ", „Ort", „…"
bzw. Tupel im mehrere Tupel aufzuspalten

Beispiele

1. NF erfüllt	Tab_8 mit Primärschlüssel{„Personalnummer", „Qualifikation"}
1. NF verletzt	Tab_1 und Tab_2 mehrere Attributwerte im Attribut „Qualifikation"

2. Normalform (2. NF)

Bedingungen

- Relation in 1. Normalform, zusätzlich
- keine Attribute abhängig vom Primärschlüssel

immer in 2. NF
- Relationen mit Einzelattribut als Primärschlüssel
- Relationen mit 2 Attributen

Beispiele

2. NF erfüllt	Tab_9 mit Primärschlüssel{„lfd. Nr.“}
2. NF verletzt	Tab_8 mit Primärschlüssel{„Personalnummer“, „Qualifikation“} Primärschlüssel-Teilattribut „Personalnummer“ identifiziert eindeutig Attribute „Mitarbeitername“, „…“

ggf. sind Tabellen geeignet aufzuteilen und zu verknüpfen,
z.B. in Tabelle „Stammdaten“ u.a. Tabellen

3. Normalform (3. NF)

Bedingungen

- Relation in 2. Normalform, zusätzlich
- keine (Nichtschlüssel-)Attribut
  abhängig von anderen (Nichtschlüssel-)Attributen

ggf. sind Tabellen geeignet aufzuteilen und zu verknüpfen

Beispiele

3. NF erfüllt	Tab_3
3. NF verletzt	Tab_9 mit Primärschlüssel{„lfd.Nr.“} Nichtschlüsselattribut „Personalnummer“ identifiziert eindeutig Attribute „Mitarbeitername“, „…“

Weitere Normalformen: 4./5. NF, BCNF (Boyce-Codd-NF)

DBMS - Transaktionen

Folge mehrerer Aktivitäten (Änderungen von Attributwerten),
die eine logische Einheit bilden

z.B.

Abnahme Handwerkerleistung
(Firma, Art der Arbeit, Arbeitsort, Zeitraum, Qualität, Vergütung, …)

DBMS-Aufgaben	eintragen	Auftragsende
	veranlassen	Nachfolgearbeiten
	erhöhen	Tagesumsatz
	veranlassen	steuerliche Kostenanrechnung
	veranlassen	Überweisung Rechnungsbetrag
	…	

Gefahr von Widersprüchen in der Datenbasis

- bei Absturz des DBMS-Rechners vor Transaktionsende
- bei (zeit-)parallelem Mehrnutzerzugriff

→ DBMS muss **Konsistenz sichern** (ACID-Prinzip)!

ACID – Pflichteigenschaften

- Atomicity (Atomarität)

  Änderungen an der Datenbank werden
  „ganz oder gar nicht“ ausgeführt.

- Consistency (Konsistenz)

  Daten in Datenbank müssen
  vor und nach einer Transaktion eindeutig konsistent sein.

- Isolation (Isolation)

  Die Datenkonsistenz muss auch bei
  zeitparallel stattfindenden Transaktionen erhalten bleiben.
  (ggf. muss DBMS den Zugriff sperren)

- Durability (Dauerhaftigkeit)

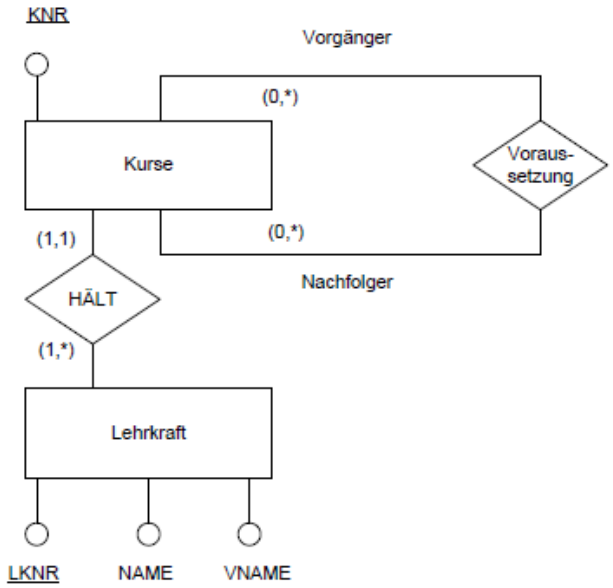
  nach Transaktion in die Datenbank geschriebene Daten müssen
  dort dauerhaft gespeichert bleiben; auch nach Systemabsturz, …

Datenbankentwurf

Methoden
zur grafischen Darstellung der Strukturierung einer Datenbasis

- ERM (Entity Relationship Diagram)
- …
- UML (Unified Modeling Language)

Beispiel:
Modellierung Schulungszentrum



Zitat:
Prof. Datenbanken (TUD/INF)

Datenbankoperationen

Verwaltungsoperationen

- Anlegen/Löschen von Tabellen und Indextabellen
- Datenimport und –export
- Tabellenwartung (z.B. logisch gelöschte Tupel physisch entfernen)
- Datensicherung (Replikation, Backup, Logging)

Abfrageoperationen

- Projektion - Relation auf Teilmenge beschränken (weniger Spalten)
- Selektion - Ausgabetupel einschränken über logische Bedingungen (z.B. „Alter über 50“)
- Verbund - Vereinigung von Attributen unterschiedlicher Relationen zu neuer Tabelle

Datenbank – Schnittstelle für Endnutzer

1. **Grafisches Interface**

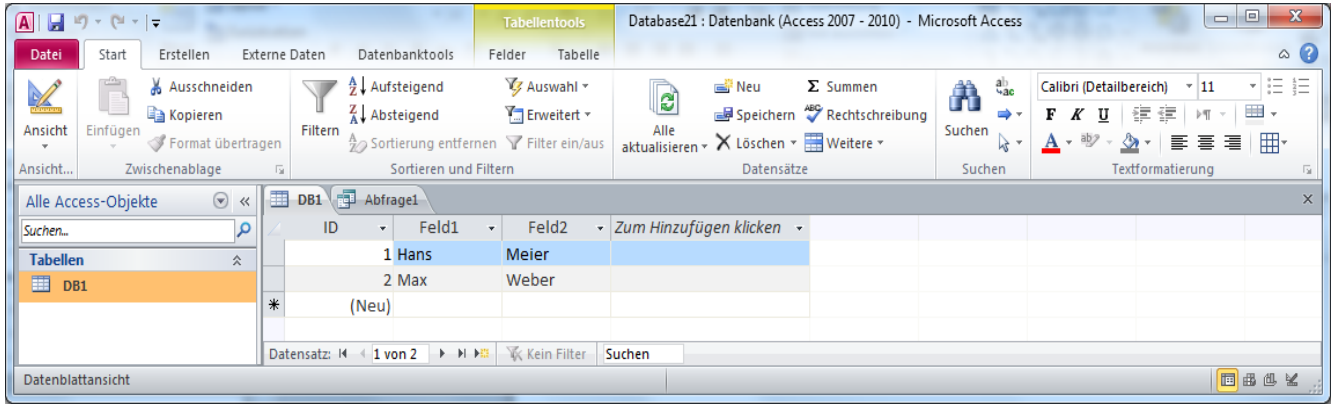
   - leicht erlernbar, komfortabler Zugriff über Formulare, …
     (Formularerstellung allerdings relativ aufwendig)
   - leider meist DBS-spezifisch

2. **Abfragesprache**, z.B. SQL (Structured Query Language)

   - ermöglicht DB-Einbindung in Verarbeitungsprogramme
     durch Übergabe von Anweisungen über spezielle Schnittstellen
     (z.B. ODBC, JDBC)
   - geeignet auch für Zugriff auf DB-Server in Rechnernetzen
   - interaktive Nutzung unkomfortabel

MS Access – DBS für einfache Büroarbeiten

komfortable grafische Nutzerschnittstelle



SQL-Schnittstelle (für Spezialisten)