

INHALTSVERZEICHNIS:

DATENBANKEN..... 3

GEGENÜBERSTELLUNG DATEISYSTEM – DATENBANKSYSTEM 3

DATENBANKSYSTEME 4

DATENBANK-MANAGEMENT-SYSTEM (DBMS)..... 4

DATENBANKMODELLE 5

DATENMODELL..... 5

RELATIONALE DATENBANKEN 5

PLANUNG EINER RELATIONALEN DATENBANK..... 5

GRAFISCHE DARSTELLUNG DES LOGISCHEN DB-ENTWURFS 5

ER-MODELL..... 6

RELATIONENSHEMA 6

BEZIEHUNGEN 7

BEZIEHUNGSARTEN 7

BEZIEHUNGEN – BEISPIELE 8

PRIMÄRSCHLÜSSEL 8

FREMDSCHLÜSSEL (ALTERNATIVSCHLÜSSEL) 8

ÜBUNG 1 9

LÖSUNG ÜBUNG 1 FRAU MANG..... 9

LÖSUNG ÜBUNG 3..... 10

LÖSUNG ÜBUNG 6 FR. MANG 11

ÜBUNG 9 11

LÖSUNG ÜBUNG 9	11
ACCESS	12
SQL	15
PARAMETERABFRAGE ODER FLEXIBLE ABFRAGE	22
ABFRAGE MIT BERECHNETEM FELD (EINZELPREIS * LAGERBESTAND)	22
AGGREGATFUNKTION	22
DATA MANIPULATION LANGUAGE (DML-BEFEHLE)	27
AUFGABEN	27
DATA DEFINITION LANGUAGE (DDL-BEFEHLE)	28
ÜBUNG 14	29
SICHTEN – VIEWS (DDL-BEFEHLE)	29
RECHTE (DCL-BEFEHLE)	31
WIEDERHOLUNGSFRAGEN	33
MAKROS IN ACCESS	36
GLOSSAR	36
WEITERE QUELLEN	36
BUCHEMPFEHLUNGEN	36

Datenbanken

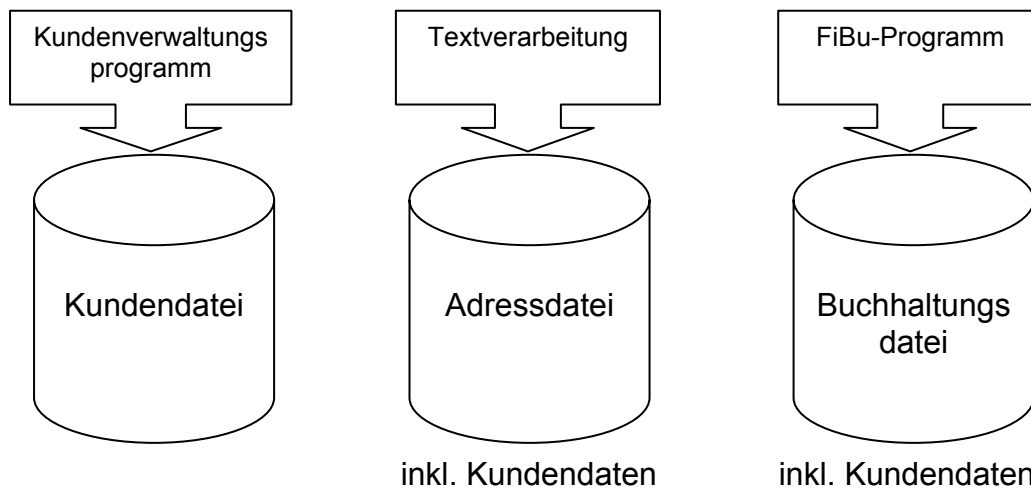
→ Eine Datenbank ist eine selbstständige und auf Dauer ausgelegte Datenorganisation, welche einen Datenbestand sicher und flexibel verwalten kann.

Aufgaben:

- 📖 Datenbanken sollen dem Benutzer den Zugriff auf gespeicherte Daten ermöglichen, ohne dass dieser wissen muss, wie die Daten gespeichert bzw. organisiert sind.
- 📖 Datenbanken sollen verhindern, dass ein Benutzer ohne entsprechende Rechte die Daten sichten und manipulieren kann.
- 📖 DB muss es ermöglichen, die interne Datenorganisation zu ändern, ohne dass dadurch die Benutzeroberfläche beeinflusst wird.

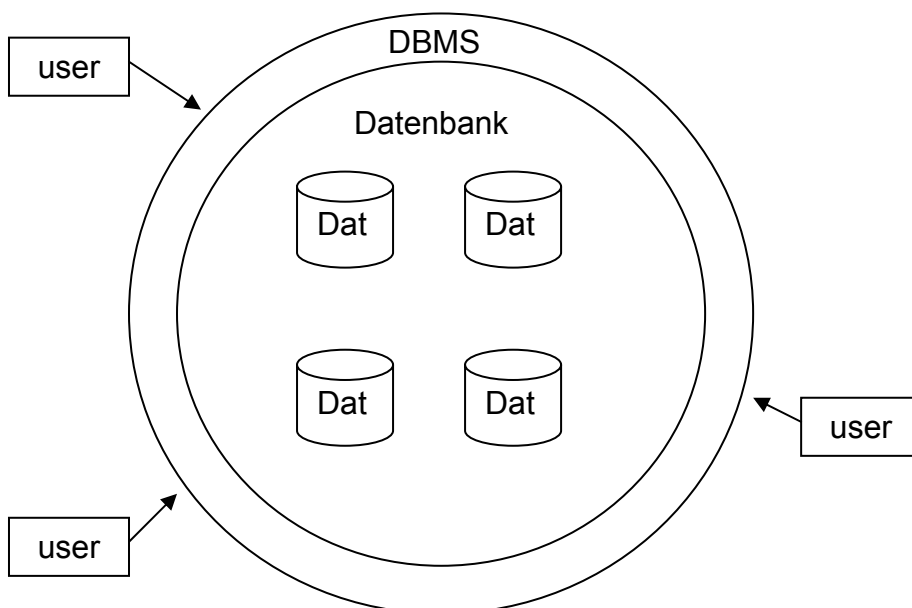
Gegenüberstellung Dateisystem – Datenbanksystem

- Dateisystem** → Daten werden in einzelnen Dateien gespeichert. Anwendungsprogramme greifen auf diese Dateien zu.
- **Probleme:**
 - Redundanzen
 - Jedes Programm hält seinen eigenen Bestand.
 - Führt zu Mehrfachspeicherung.
 - Redundanz → Mehrfaches Speichern ein und derselben Information
 - Inkonsistenz → gleiche Daten werden mit unterschiedlichen Werten gespeichert.
Tritt als Folge der Redundanz auf, wenn redundante Datenbestände nicht einheitlich aktualisiert werden.
 - Daten-Programm-Abhängigkeit
Programm enthält die Struktur der Datei. Wird die Dateistruktur geändert, ist zwangsläufig eine Änderung des Programms notwendig.
 - hoher Wartungsaufwand
 - inflexibel bei Weiterentwicklungen
 - keine zentrale Zugriffskontrolle
Programme sind nicht miteinander verknüpft
 - kaum Möglichkeit zur Einhaltung von Standards
z.B. Eingabeformate



Datenbanksysteme

- Entwickelt um die Probleme mit den Dateisystemen zu lösen
- Eigenschaften → Datenbank ist auf einen bestimmten Anwendungsbereich ausgerichtet
- Alle User benutzen die gleiche Datenbasis
- Daten sind weitgehend konsistent und redundanzfrei
- keine Daten-Programm-Abhängigkeit
- zentrale Zugriffskontrolle
- Flexibilität bei Wartung / Entwicklung
- Integritätskontrolle
 - Integrität = Korrektheit der Daten. Einhalten von DB-Regeln
- Werkzeuge zur Wiederherstellung von Daten nach Abstürzen
- Zugriffsschutz
- gemeinsame Sprache für Definition, Änderung und Erfassung von Daten



DBMS stellt die Daten so bereit wie sie von den Anwendungsprogrammen benötigt werden.

→ SW-System zur Organisation von DB

Datenbankmodelle:

hierarchisch = streng festgelegter Aufbau → bereits am Anfang müssen die Abfragen geplant werden

netzwerkmodelle= Verknüpfungen unter den Dateien beliebig möglich

objektrelational = hybrid aus relational und objektorientiert

Datenmodell

- Beschreibung der DB-Struktur
- Hierarchisch
- Netzwerk
- Relational (Standard)
- Objektorientiert (im kommen)

Relationale Datenbanken

Im relationalen Datenbankmodell werden die Daten in Tabellen (= Relationen) gespeichert. Zwischen den Tabellen bestehen Beziehungen.

Vorteile:

- 📖 hohe Flexibilität
- 📖 geringe Redundanzen
- 📖 weniger Inkonsistenzen

siehe dazu „Stärken und Grenzen relationaler DB“:

<http://www.bw.fh-deggendorf.de/kurse/db/skripten/skript12.pdf>

Planung einer relationalen Datenbank

Vorgehensweise:

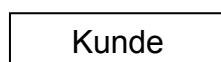
- 📖 Ist-Zustand feststellen
- 📖 Ermittlung welche Tabellen benötigt werden
- 📖 Festlegung der Schlüssel- bzw. Nichtschlüsselattribute (=Felder)
- 📖 Aufbau der Beziehungen
- 📖 Abschließende Prüfung auf Redundanzen

Grafische Darstellung des logischen DB-Entwurfs

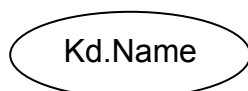
Hilfsmittel zur Planung einer DB

- Entity-Relationship-Modell
- zeigt die Tabellen (= Relationen, Entitätsmengen) mit ihren Attributen (= Eigenschaften, Felder) und den Beziehungen

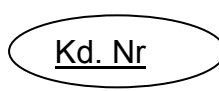
Symbole:



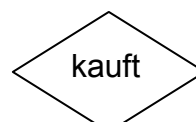
Tabelle



Attribut-Feld



Primär-Schlüssel



Beziehung

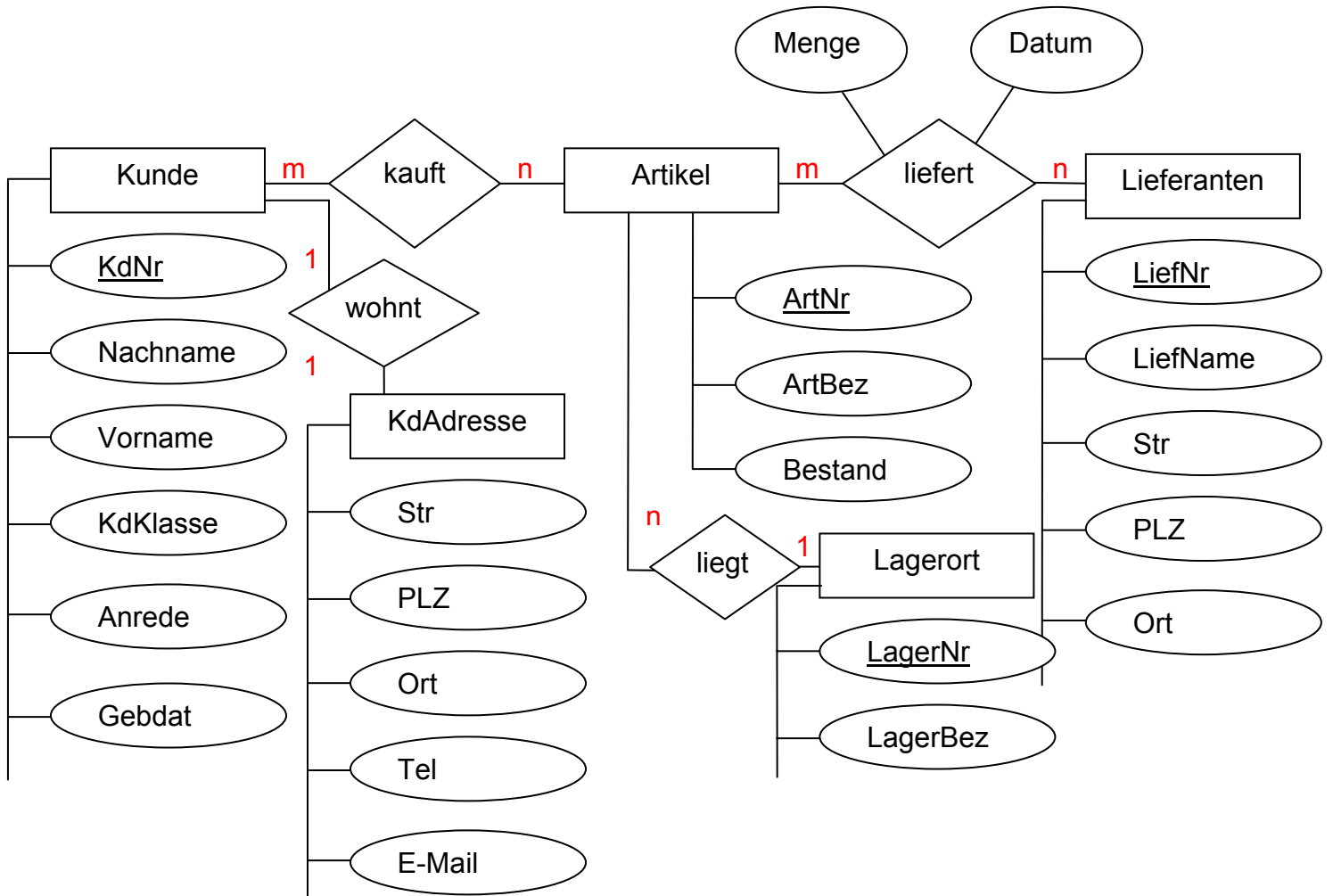


mehrwertiges
Attribut

Bsp.:

Datenbank enthält

- 📖 Kunden
 - 📖 Kunden kaufen Artikel
 - 📖 Artikel haben einen Lagerort
 - 📖 Lagerorte sind gekennzeichnet durch LagerNr + LagerBez
 - 📖 Lieferanten liefern Artikel
 - 📖 Kunden haben eine eindeutige Adresse
- Aus Gründen der Übersichtlichkeit ist eine eigene Adresstabelle aufzubauen.



(7 Tabellen)

Relationsschema:

Kunde (KdNr, Nachname, Vorname, KdKlasse, Anrede, Gebdat)

KdAdresse (Str, PLZ, Ort, Tel, E-Mail, KdNr)

Artikel (ArtNr, ArtBez, Bestand, LagerNr)

Lagerort (LagerNr, LagerBez)

Lieferanten (LiefNr, LiefName, Str, PLZ, Ort)

kauft (ArtNr, KdNr)

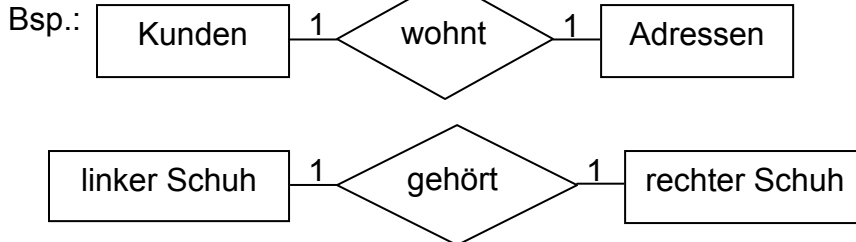
liefert (ArtNr, LiefNr, Menge, Datum, VorgangsNr)

Beziehungen

- notwendig um eine Verknüpfung zwischen Datensätzen herzustellen
- notwendig zur Vermeidung von Redundanzen
- werden hergestellt über Primär- und Fremdschlüssel

Beziehungsarten:

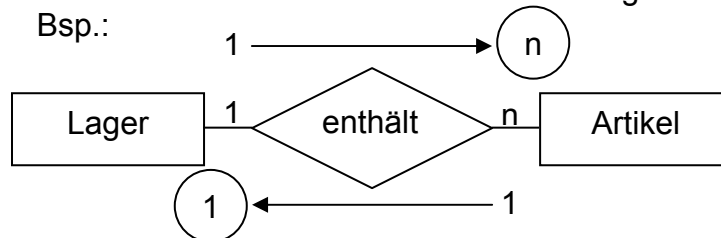
1:1 Ein Datensatz aus der 1. Tabelle steht in Beziehung zu genau einem Datensatz in der 2. Tabelle



Umsetzung der 1:1 Beziehung

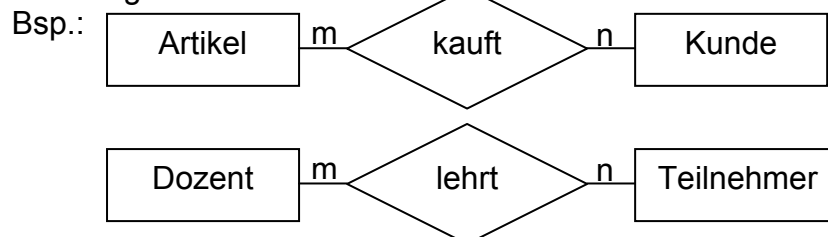
- Primärschlüssel aus einer der beiden Tabellen geht als Fremdschlüssel in die zweite Tabelle.
- Hat die zweite Tabelle keinen Primärschlüssel, kann Fremdschlüssel zum Primärschlüssel werden.
- Alternative: Haben beide Tabellen einen Primärschlüssel, kann eine neue Tabelle erzeugt werden. Diese Tabelle enthält dann die beiden Schlüssel.

1:n Beziehung → Ein Datensatz aus der 1. Tabelle (1-Seite) steht in Beziehung zu vielen Datensätzen aus der 2. Tabelle (n-Seite)
n:1 Ein Datensatz aus der 2. Tabelle (n-Seite) steht aber nur zu einem Datensatz aus der 1. Tabelle in Beziehung



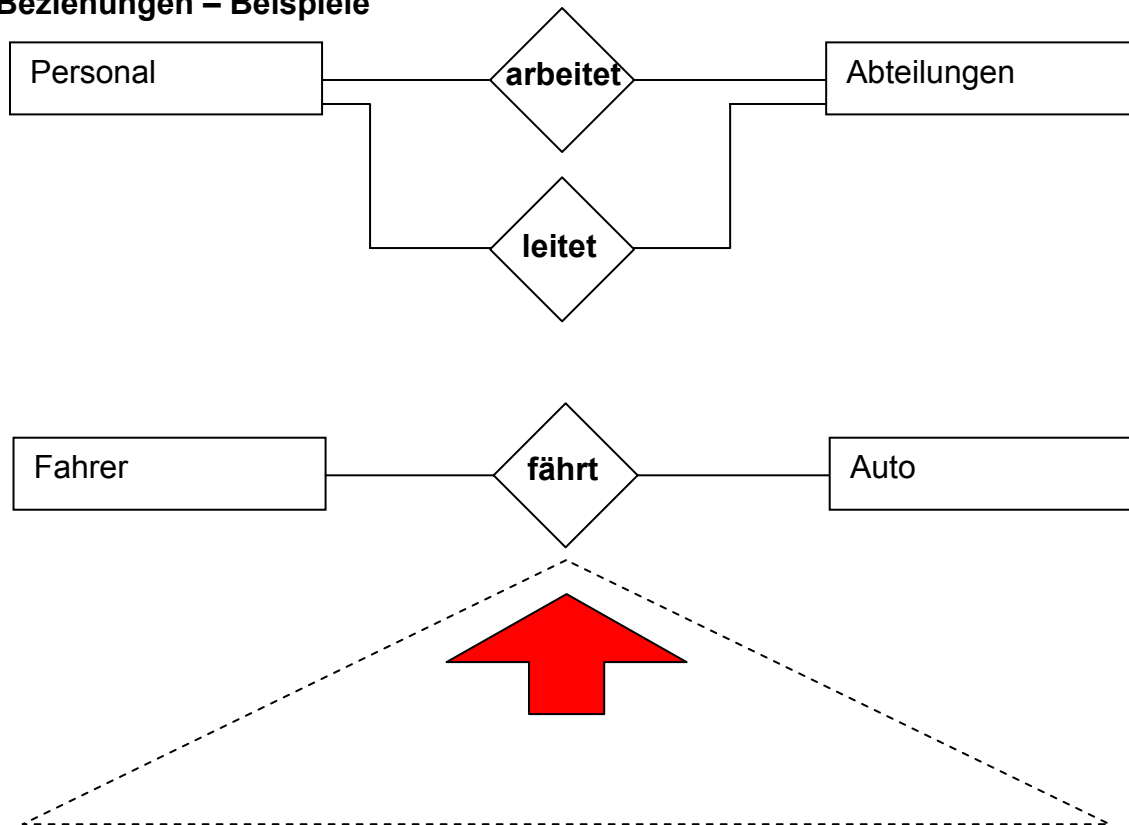
Umsetzung: Primärschlüssel der Tabelle auf der „1-Seite“ wird als Fremdschlüssel auf die „n-Seite“ übertragen.

m:n Beziehung → Ein Datensatz aus der 1. Tabelle (m-Seite) steht in Beziehung zu vielen Datensätzen auf der „n-Seite“.
Und umgekehrt!



Umsetzung: Aus m:n-Beziehung entsteht immer eine neue Tabelle. Diese enthält die beiden Primärschlüssel aus den verbundenen Tabellen (als Fremdschlüssel).

Beziehungen – Beispiele



Primärschlüssel

- identifiziert einen Datensatz eindeutig
- darf keine Nullwerte und Duplikate enthalten
- darf pro Tabelle nur einmal vorkommen
- kann sich aber aus mehreren Feldern zusammensetzen
- dient dem Aufbau von Beziehungen

zusammengesetzte Primärschlüssel

- Kombination der Feldinhalte muss eindeutig sein

Fremdschlüssel (Alternativschlüssel)

- bezieht sich auf den Primärschlüssel einer anderen Tabelle
- dient dem Aufbau von Beziehungen
- darf nur Werte enthalten, die auch in dem Primärschlüssel-Feld der verbundenen Tabelle enthalten sind

Artikel:

<u>ArtNr</u>	<u>ArtBez</u>	<u>LagNr</u>
A1	Hammer	L1
A2	Schraube	L3
A3	Säge	L3
A4	Zange	L5

Lager:

<u>LagNr</u>	<u>LagBez</u>
L1	Hochregal
L2	Zwischenregal
L3	Hof

Fremdschlüssel = doppelt unterstrichen Primärschlüssel = einmal unterstrichen

Übung 1:

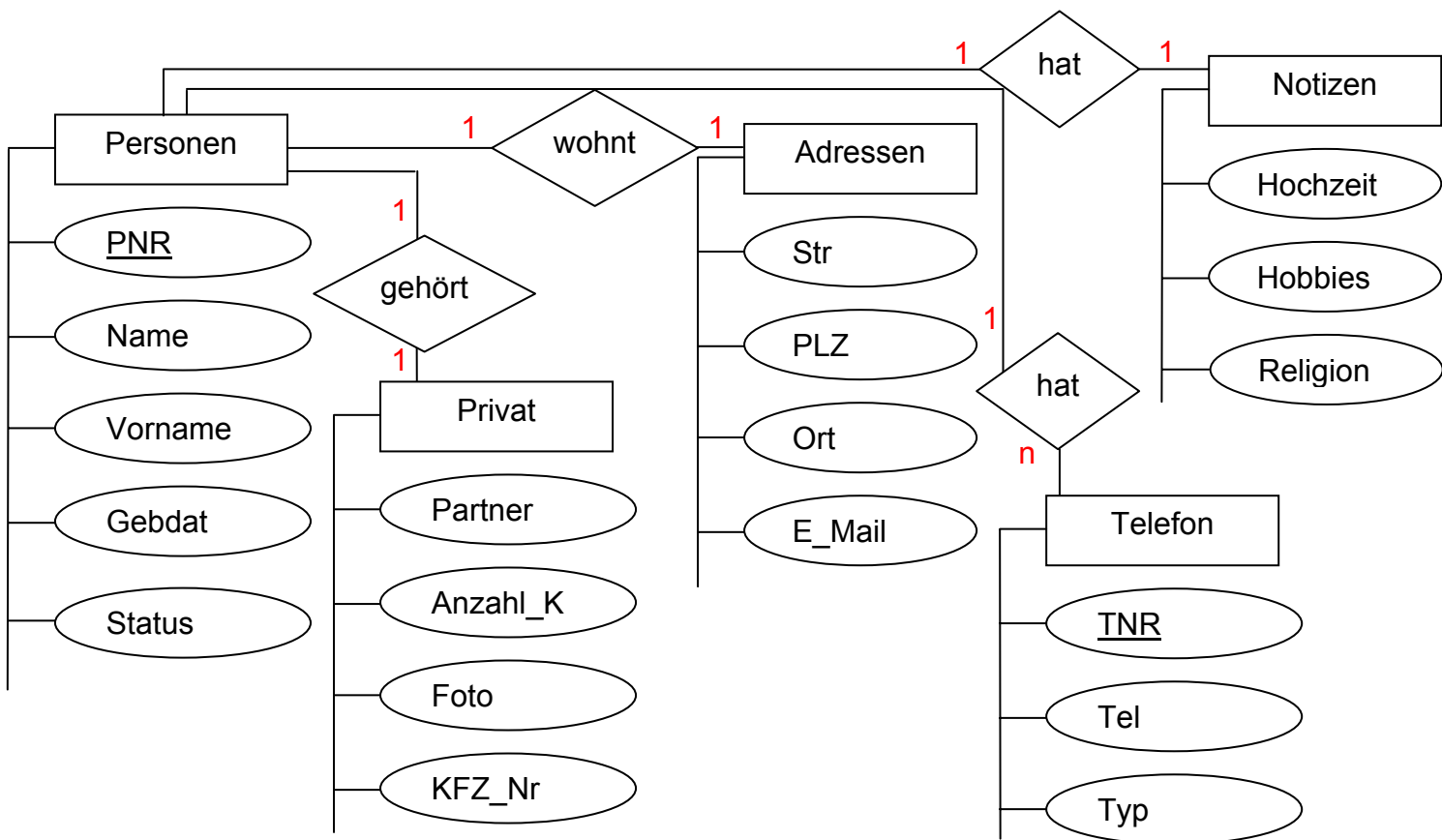
Lösung 1:

Tabellen: Personen (Nr, Name, Vorname, Gebdat, Tel, Fax, Status)
 Privat (Partner, Anzahl Kinder, Foto, KFZ-Nr)
 Adressen (Str, PLZ, Ort)
 Notizen (Hochzeit, Hobbies)
 Tel (Telefon, Handy)
 (5 Tabellen)

Lösung 2:

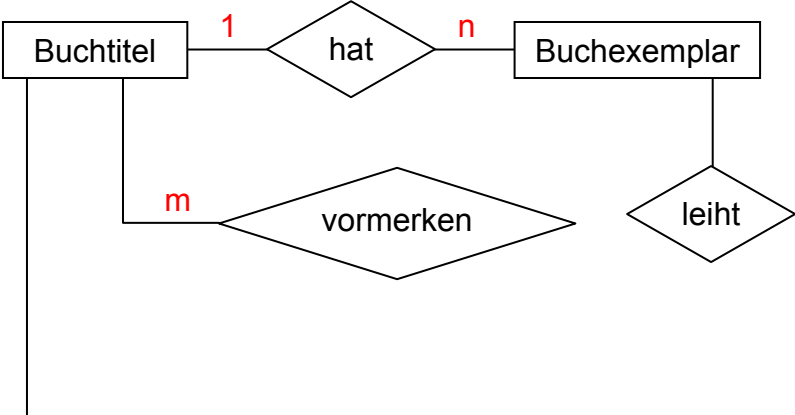
Tabellen: Personen (Nr, Name, Vorname, Gebdat, Tel, Fax, Status)
 Privat (Nr, PrNr, Partner, Anzahl Kinder, Foto, KFZ-Nr)
 Adressen (Nr, Anr, Str, PLZ, Ort)
 Notizen (Nr, Nnr, Hochzeit, Hobbies)
 Tel (Nr, Tnr, Telefon, Handy)
 (5 Tabellen)

Lösung Übung 1 Frau Mang:



Personen (PNR, Name, Vorname, Gebdat, Status)
 Privat (Partner, Foto, Anzahl_K, KFZ-Nr, PNR)
 Adressen (Str, PLZ, Ort, E-Mail, PNR)
 Notizen (Hochzeit, Hobbies, Religion, PNR)
 Telefon (TNR, Tel, Typ, PNR)

Lösung Übung 3:



Übung 6

Lösung Übung 6 Fr. Mang:

Relationsschema:

Kunde (KdNr, Name, VN, Str, KdNr, MNR)

Mitarbeiter (MNR, Name, VN, Str)

Vorgang (VNR, VDat, Rdat)

CD (CDNR, CDTitel, Preis, Anschaffdat)

CD_Inhalt (InhNr, Länge, CDNR, IntNr, TNR)

Interpret (IntNr, Name, VN)

Titel (TNR, Bez, Erschjahr)

Ausgabe (CDNR, VNR)

----- = Beziehungen

Übung 7

Lösung Übung 7 Fr. Mang:

Relationsschema:

Fahrlehrer (FINr, Name, VN, Gebdat)

Fahrschüler (FsNr, Name, VN, Gebdat)

Fahrzeug (FzNr, KFZ_Nr, Typ, Ans_Dat, Preis, HNR)

Händler (HNR, Str, PLZ, Ort)

Unterricht (UNr, Dat, Zeit, Typ, FINr, FsNr, FzNr)

Übung 8

Lösung Übung 8 Fr. Mang:

Relationsschema:

Übung 9

Lösung Übung 9:

Relationsschema:

Sportler (SNR, Name, Vorname, Str, PLZ, Ort)

Trainer (TNR, Name, Vorname, Str, PLZ, Ort)

Abteilung (ANR, Sportart)

nimmt teil (SNR, ANR)

trainiert (TNR, ANR)

ACCESS

→ Anwendungsprogramm, das auf einem DBMS basiert (= MS Jet Engine)

- Vorteile**
- „einfache“ Anwendungsprogrammierung
 - bietet einfache Möglichkeiten zur Gestaltung von Formularen und Berichten
 - ermöglicht schnelle Erfassung von Tabellen und Daten
 - ermöglicht einfaches Erstellen von Abfragen

- Nachteile**
- weniger geeignet für große Anwendungen
 - eingeschränkte SQL-Befehle

Übung (demo.mdb)

Kunde (KdNr, Name, Vorname, Gebdat, Anrede, KdKlasse)

Artikel (ArtNr, ArtBezeichnung, Bestand, LagerNr)

Lieferant (LiefNr, LiefName, LiefVorname, Str, PLZ, Ort)

kauft (KdNr, ArtNr)

liefert (LiefNr, ArtNr, Datum, Menge, VorgangsNr)

Lagerort (LagerNr, LagerBezeichnung)

KdAdresse (KdNr, Str, PLZ, Ort, Tel, Email)

KdNr = 4 Zeichen Bsp.: 12AB, 3474, 12CD

Leere Datenbank - Speichern als demo.mdb - Erstellt eine Tabelle in der Entwurfsansicht (neue Tabelle erstellen):

Speichern als Tabelle: Kunde (kann erst nach einer Dateneingabe gespeichert werden)

Feldname: KdNr; Felddatentyp: Text; Allgemein – Feldgröße: 4; Allgemein -

Eingabeformat: >00LL (> erzwingt, dass alle nachfolgenden Zeichen in Großbuchstaben umgewandelt werden; die ersten zwei Stellen stehen für Ziffern von 0-9; Stelle 3-4 steht für einen Buchstaben von A-Z → siehe Buch Seite 62 – ACCESS Datenbankentwickler),

Beschriftung: Kundennummer (optional: auf der Programmoberfläche wird dann anstatt der internen KdNr das Feld Kundennummer stehen → der Benutzer sieht anstatt KdNr → Kundennummer; in der Programmierung kann beides angesprochen werden)

KdNr mit Schlüsselssymbol als Primärschlüssel setzen → Eingabe ist immer erforderlich

Feldname: Name; Felddatentyp: Text; Allgemein - Feldgröße: 50; Allgemein - Eingabe erforderlich: ja

Feldname: Vorname; Felddatentyp: Text; Allgemein - Feldgröße: 50; Allgemein - Eingabe erforderlich: ja

Feldname: Geburtsdatum; Felddatentyp: Datum/Uhrzeit; Allgemein - Eingabeformat: 00/00/0000; Allgemein - Gültigkeitsregel: <=datum() (nicht für Kunden, die noch nicht geboren sind → anhand der Systemzeit); Gültigkeitsmeldung: Kein zukünftiges Datum eingeben;

Feldname: Anrede; Allgemein - Feldgröße: 10; Felddatentyp: Nachschlageassistent – Ich möchte selbst Werte in die Liste eingeben – Herr; Frau; Firma - oder unter Nachschlagen alles selbst eingeben (andere Lösung ohne DropDownMenü:

Felddatentyp: Text; Allgemein - Gültigkeitsregel: „Herr“ oder „Frau“ oder „Firma“);

Nachschlagen - Nur Listeneinträge: Ja

Feldname: KdKlasse; Felddatentyp: Text; Allgemein – Feldgröße: 2; Allgemein – Gültigkeitsregel: >=10 Und <=20 (Alternative: Gültigkeitsregel: Zwischen „10“ Und „20“);

Tabelle schließen (Datei – Schließen)

Erstellt eine Tabelle in der Entwurfsansicht (neue Tabelle erstellen)

Speichern als Tabelle: Artikel

Feldname: ArtNr; Felddatentyp: Text; Allgemein – Feldgröße: 3; Allgemein – Eingabeformat: 000 (nur Zifferneingabe)

ArtNr mit Schlüsselssymbol als Primärschlüssel setzen → Eingabe ist immer erforderlich

Feldname: ArtBezeichnung; Felddatentyp: Text

Feldname: Bestand; Felddatentyp: Zahl; Allgemein – Feldgröße: Single (siehe Buch Seite 57 – ACCESS 2000 Datenbankentwickler); Allgemein – Format: Standardzahl;

Allgemein – Dezimalstellenanzeige: 2

Feldname: LagerNr; Felddatentyp: Text; Allgemein – Feldgröße: 5; Allgemein – Eingabeformat: 00000

Tabelle schließen (Datei – Schließen)

Erstellt eine Tabelle in der Entwurfsansicht (neue Tabelle erstellen)

Speichern als Tabelle: Lieferant

Feldname: LiefNr; Felddatentyp: Text; Allgemein – Feldgröße: 5; Allgemein – Eingabeformat: >LLLLL (LiefNr 5 Stellen nur Großbuchstaben)

LiefNr mit Schlüsselssymbol als Primärschlüssel setzen → Eingabe ist immer erforderlich

Feldname: LiefName; Felddatentyp: Text

Feldname: LiefVorname; Felddatentyp: Text

Feldname: Strasse; Felddatentyp: Text

Feldname: PLZ; Felddatentyp: Text; Allgemein – Feldgröße: 5; Allgemein – Eingabeformat: 00000

Feldname: Ort; Felddatentyp: Text

Tabelle schließen (Datei – Schließen)

Erstellt eine Tabelle in der Entwurfsansicht (neue Tabelle erstellen)

Speichern als Tabelle: Lagerort

Feldname: LagerNr; Felddatentyp: Text; Allgemein – Feldgröße: 5; Allgemein – Eingabeformat: 00000 (LagerNr 5 Stellen nur Ziffern)

LagerNr mit Schlüsselssymbol als Primärschlüssel setzen → Eingabe ist immer erforderlich

Feldname: LagerBezeichnung; Felddatentyp: Text

Rest (kauft, liefert, KdAdresse) wie oben anlegen

VorgangNr 6 Ziffern Text

Formatvorlagen, wie oben anlegen, z.B. KdNr mit Feldgröße 4, etc.

Fremdschlüssel anlegen: Extras – Beziehungen oder Symbol: Beziehungen – alle Tabellen markieren – Hinzufügen – Schließen = Beziehungsfenster (kein ER-Modell!!)
Mit referentieller Integrität anklicken = Prüfung der Gültigkeit zwischen zwei Tabellen (siehe auch Hilfe in Access → Mit referentieller Integrität anklicken – F1 drücken)

Beziehungen mit referentieller Integrität

→ Referentielle Integrität prüft ständig die Gültigkeit einer Beziehung zwischen zwei Tabellen.

Beziehung ist nicht mehr gültig, wenn der Fremdschlüssel in der einen Tabelle andere Werte enthält, als im dazugehörigen Primärschlüsselfeld der anderen Tabelle vorhanden sind.

Gefahr besteht bei Änderungen im Primärschlüssel- bzw. Fremdschlüsselfeld, sowie beim Einfügen von Sätzen in die untergeordnete Tabelle und beim löschen von Sätzen in der übergeordneten Tabelle.

Aktualisierungsweitergabe

Änderungen im Primärschlüsselfeld der übergeordneten Tabelle werden automatisch an das dazugehörige Fremdschlüsselfeld der untergeordneten Tabelle weitergegeben.

Löschweitergabe

Löschen von Datensätzen in der übergeordneten Tabelle bewirkt automatisch das Löschen der verbundenen Datensätze in der untergeordneten Tabelle!

SQL

- Structured Query Language
- Sprache der 4. Generation
 - deskriptiv (nicht mehr beschreibend → ich sage nicht, wie etwas zu machen ist, sondern nur noch, was zu machen ist)
 - mengenorientiert
 - nicht prozedurale Sprache

SQL

- QL → Query Language (Abfragesprache) → Select
- DML → Data Manipulation Language → Insert, Update, Delete
- DDL → Data Definition Language
- DCL → Data Control Language → Grant Revoke → Rechteentzug

Select

- Auswahlabfragen
- ```
Select Feld1, Feld2...
From Tabelle;
Bsp.: Select * } liefert alle Felder und alle Sätze aus
 From Artikel; } der Tabelle Artikel
```

**Sortierung** → Standard aufsteigend

Absteigende Sortierung → DESC → Descending (siehe Glossar) → Z - A

Aufsteigende Sortierung → ASC → Ascending → A - Z

Bei aufsteigender Sortierung muss ASC nicht angegeben werden.

```
Bsp.: Select *
 From Kunden
 Order by Ort DESC;
```

## **Übung (foodware.mdb)**

### **einfache Abfrage:**

```
SELECT Artikel_Nr, Artikelname
FROM Artikel;
```

### **Sortierung (Standard: aufsteigend) nach dem Lagerbestand:**

```
SELECT Artikel_Nr, Artikelname, Lagerbestand
FROM Artikel
ORDER BY Lagerbestand;
```

### **zwei Sortierungen: erst absteigend nach Lagerbestand dann aufsteigend nach Einzelpreis:**

```
SELECT Artikel_Nr, Artikelname, Lagerbestand
FROM Artikel
ORDER BY Lagerbestand DESC, Einzelpreis;
```

**zwei Sortierungen: Sortiert nach Ort aufsteigend innerhalb gleicher Orte absteigend nach der Firma:**

```
SELECT *
FROM Kunden
ORDER BY Ort, Firma DESC;
```

**Filtern: Alle Kunden, die aus London kommen, werden angezeigt:**

```
SELECT *
FROM Kunden
WHERE Ort = "London";
```

**Filtern: Alle Kunden, die aus London oder Madrid kommen, werden angezeigt:**

```
SELECT *
FROM Kunden
WHERE Ort = "London" or Ort = "Madrid";
```

**Sätze, deren Bestand über 100 liegt, sortiert nach Artikelname**

```
SELECT Artikel_Nr, Artikelname, Lagerbestand, Einzelpreis
FROM Artikel
WHERE Lagerbestand > 100
ORDER BY Artikelname;
```

**Sätze, deren Bestand über 100 liegt, sortiert nach Artikelname und Kategoriennummer 4 lautet**

```
SELECT Artikel_Nr, Artikelname, Lagerbestand, Einzelpreis
FROM Artikel
WHERE Lagerbestand > 100 and Kategorie_Nr = 4
ORDER BY Artikelname;
```

**Alle Sätze aus Tabelle Personal mit Ort London, Seattle und Kent**

```
SELECT *
FROM Personal
WHERE Ort = "London" or Ort = "Seattle" or Ort = "Kent";
```

**Verkürzte Form (nur bei Oder-Verknüpfungen)**

```
SELECT *
FROM Personal
WHERE Ort IN ("London", "Seattle", "Kent");
```

**Bei = müssen die Werte exakt stimmen.**

**Für eine Suche nach Stra\* (Suche mit Platzhaltern, wie \* oder ?) muss anstatt dem = der Befehl LIKE benutzt werden:**

```
SELECT *
FROM Personal
WHERE Ort LIKE "S*g";
```

**Platzhalter**

\* = beliebig viele Zeichen } ACCESS SQL  
? = 1 Zeichen

% = beliebig viele Zeichen } andere SQL Dialekte  
- = 1 Zeichen

### Übereinstimmung aus zwei Tabellen (siehe Beziehungen bezüglich Fremdschlüssel in WHERE)

```
SELECT Artikel_Nr, Artikelname, Lieferanten.Lieferanten_Nr, Firma
FROM Artikel, Lieferanten
WHERE Artikel.Lieferanten_Nr = Lieferanten.Lieferanten_Nr;
```

#### Abfrage 8:

Tabelle Personal: Personal\_Nr, Nachname, Ort  
Tabelle Bestellungen: Bestell\_Nr, Bestelldatum  
Nur Sätze, deren Kunden\_Nr mit „M“ beginnt

```
SELECT Personal.Personal_Nr, Personal.Nachname, Personal.Ort,
Bestellungen.Bestell_Nr, Bestellungen.Bestelldatum
FROM Personal, Bestellungen
WHERE Kunden_Nr Like "M*" AND Personal.Personal_Nr =
Bestellungen.Personal_Nr;
```

#### Abfrage 9:

Tabelle Kunden: Kunden\_Nr, Firma  
Tabelle Personal: Personal\_Nr, Nachname  
Tabelle Bestellungen: Bestell\_Nr, Bestelldatum  
Tabelle Bestelldetails: Artikel\_Nr, Anzahl

```
SELECT Kunden.Kunden_Nr, Kunden.Firma, Personal.Personal_Nr,
Personal.Nachname, Bestellungen.Bestell_Nr, Bestellungen.Bestelldatum,
Bestelldetails.Artikel_Nr, Bestelldetails.Anzahl
FROM Kunden, Personal, Bestellungen, Bestelldetails
WHERE Kunden.Kunden_Nr = Bestellungen.Kunden_Nr AND Bestellungen.Bestell_Nr
= Bestelldetails.Bestell_Nr AND Personal.Personal_Nr = Bestellungen.Personal_Nr;
```

#### Abfrage 10:

Alle Felder aus Artikel aber nur die Artikel, die vom Lieferanten mit dem Namen „Exotic Liquids“ geliefert werden.

```
SELECT *
FROM Artikel, Lieferanten
WHERE Lieferanten.Firma = "Exotic Liquids" AND Lieferanten.Lieferanten_Nr =
Artikel.Lieferanten_Nr;
```

#### Hash

```
SELECT *
FROM Bestellungen
WHERE Bestelldatum > #01/01/2000#;
```

## Übung 10:

### Aufgabe 1

```
SELECT Artikel.Lagerbestand, Artikel.Artikel_Nr, Artikel.Artikelname,
Artikel.Einzelpreis, Artikel.Auslaufartikel
FROM Artikel
ORDER BY Artikel.Lagerbestand DESC;
```

### Aufgabe 2

```
SELECT Bestellungen.Kunden_Nr, Bestellungen.Empfänger, Bestellungen.Ort,
Bestellungen.Bestimmungsland, Bestellungen.Lieferdatum
FROM Bestellungen
ORDER BY Bestellungen.Lieferdatum;
```

### Aufgabe 3

```
SELECT Bestellungen.Bestelldatum, Bestellungen.Kunden_Nr,
Bestellungen.Empfänger, Bestellungen.Ort, Bestellungen.PLZ
FROM Bestellungen
ORDER BY Bestellungen.Bestelldatum;
```

### Aufgabe 4

```
SELECT Kunden.Land, Kunden.Ort, Kunden.Firma
FROM Kunden
ORDER BY Kunden.Land, Kunden.Ort;
```

### Aufgabe 5

```
SELECT Lieferanten.Land, *
FROM Lieferanten
ORDER BY Lieferanten.Land;
```

### Aufgabe 6

```
SELECT Personal.LAND, Personal.ORT, Personal.VORNAME, Personal.NACHNAME,
Personal.GEBURTSDATUM
FROM Personal
ORDER BY Personal.LAND, Personal.ORT;
```

## Übung 11

### Aufgabe 1

```
SELECT Artikel.Lagerbestand, Artikel.Artikel_Nr, Artikel.Artikelname,
Artikel.Auslaufartikel
FROM Artikel
WHERE Artikel.Auslaufartikel = yes
ORDER BY Artikel.Lagerbestand;
```

### Aufgabe 2

```
SELECT Artikel.Artikel_Nr, Artikel.Artikelname, Artikel.Liefereinheit,
Artikel.Lagerbestand, Artikel.Bestellte_Einheiten
FROM Artikel
WHERE Artikel.Bestellte_Einheiten > 50;
```

### **Aufgabe 3**

```
SELECT Artikel.Bestellte_Einheiten, Artikel.Artikel_Nr, Artikel.Artikelname,
Artikel.Mindestbestand
FROM Artikel
WHERE Artikel.Bestellte_Einheiten = 0;
```

### **Aufgabe 4**

```
SELECT Artikel.Einzelpreis, Artikel.Artikelname
FROM Artikel
WHERE Artikel.Einzelpreis <= 5;
```

### **Aufgabe 5**

```
SELECT Bestelldetails.Bestell_Nr, Bestelldetails.Artikel_Nr, Bestelldetails.Einzelpreis,
Bestelldetails.Anzahl
FROM Bestelldetails
WHERE Bestelldetails.Rabatt > 0.20;
```

### **Aufgabe 6**

```
SELECT Bestellungen.Personal_Nr, Bestellungen.Empfänger, Bestellungen.Ort,
Bestellungen.Bestelldatum
FROM Bestellungen
WHERE Bestellungen.Personal_Nr = 4
ORDER BY Bestellungen.Kunden_Nr;
```

### **Aufgabe 7**

```
SELECT *
FROM Bestellungen
WHERE Bestellungen.Kunden_Nr = "ALFKI"
ORDER BY Bestellungen.Versanddatum DESC;
```

### **Aufgabe 8**

```
SELECT Bestellungen.Bestell_Nr, Bestellungen.Empfänger, Bestellungen.Ort,
Bestellungen.Bestimmungsland, Bestellungen.Lieferdatum
FROM Bestellungen
WHERE Bestellungen.Bestimmungsland = "USA"
ORDER BY Bestellungen.Lieferdatum;
```

### **Aufgabe 9**

```
SELECT Kunden.Firma, Kunden.Strasse, Kunden.PLZ, Kunden.Ort,
Kunden.Kontaktperson, Kunden.Position, Kunden.Telefon
FROM Kunden
WHERE Kunden.Ort = "London"
ORDER BY Kunden.PLZ;
```

### **Aufgabe 10**

```
SELECT Bestellungen.VFirma_Nr, Bestellungen.Versanddatum,
Bestellungen.Empfänger, Bestellungen.Strasse, Bestellungen.Ort,
Bestellungen.Bestimmungsland
FROM Bestellungen
WHERE Bestellungen.Versanddatum = #12/12/1994#
ORDER BY Bestellungen.VFirma_Nr;
```

### **Aufgabe 11**

```
SELECT Bestellungen.Bestell_Nr, Bestellungen.Bestelldatum
FROM Bestellungen
```

```
WHERE Bestellungen.Bestelldatum LIKE >= #01/01/2001#
```

```
AND Bestelldatum <= #31/12/2001#;
```

*oder*

```
SELECT Bestellungen.Bestell_Nr, Bestellungen.Bestelldatum
FROM Bestellungen
```

```
WHERE Bestellungen.Bestelldatum LIKE "*2001";
```

### **Aufgabe 12**

```
SELECT Personal.Personal_Nr, Personal.Vorname, Personal.Nachname,
Personal.Ort, Personal.Land, Personal.Einstelldatum
```

```
FROM Personal
```

```
WHERE Personal.Einstelldatum < #01/01/1994#
```

```
ORDER BY Personal.Einstelldatum;
```

### **Aufgabe 13**

```
SELECT *
```

```
FROM Personal
```

```
WHERE Personal.Geburtsdatum > #31/12/1960#
```

```
ORDER BY Personal.Geburtsdatum;
```

### **Aufgabe 14**

```
SELECT Bestellungen.Empfänger, Bestellungen.Ort, Bestellungen.Bestimmungsland,
Bestellungen.VFirma_Nr
```

```
FROM Bestellungen
```

```
WHERE Bestellungen.Frachtkosten > 500;
```

### **Aufgabe 15**

```
SELECT *
```

```
FROM Bestelldetails
```

```
WHERE Bestelldetails.Artikel_Nr = 41
```

```
ORDER BY Bestelldetails.Anzahl DESC;
```

### **Aufgabe 16**

```
SELECT Kunden.Firma, Kunden.Kontaktperson, Kunden.Position, Kunden.Ort
```

```
FROM Kunden
```

```
WHERE Kunden.Position LIKE "Inhaber*";
```

### **Aufgabe 17**

```
SELECT Artikel.Artikel_Nr, Artikel.Artikelname, Artikel.Liefereinheit
```

```
FROM Artikel
```

```
WHERE Artikel.Liefereinheit LIKE "*Kart*";
```

### **Aufgabe 18**

```
SELECT Kunden.Kunden_Nr, Kunden.Firma, Kunden.Strasse, Kunden.Ort
```

```
FROM Kunden
```

```
WHERE Kunden.Kunden_Nr LIKE "?????";
```

**Aufgabe 19**

```
SELECT Artikel.Liefereinheit, Artikel.Artikelname, Artikel.Lagerbestand
FROM Artikel
WHERE Artikel.Liefereinheit LIKE "**Flasche**";
```

**Aufgabe 20**

```
SELECT Lieferanten.Lieferanten_Nr, Lieferanten.Firma, Artikel.Artikel_Nr,
Artikel.Artikelname
FROM Lieferanten, Artikel
WHERE Artikel.Lieferanten_Nr = Lieferanten.Lieferanten_Nr;
```

**Aufgabe 21**

```
SELECT Kunden.Kunden_Nr, Kunden.Firma, Kunden.Ort, Bestellungen.Bestell_Nr,
Bestellungen.Bestelldatum
FROM Kunden, Bestellungen
WHERE Kunden.Kunden_Nr = Bestellungen.Kunden_Nr
ORDER BY Bestellungen.Bestelldatum;
```

**Aufgabe 22**

```
SELECT Personal.Personal_Nr, Personal.Nachname, Personal.Vorname,
Bestellungen.Bestell_Nr, Bestellungen.Empfänger, Bestellungen.Ort
FROM Personal, Bestellungen
WHERE Personal.Personal_Nr = Bestellungen.Personal_Nr;
```

**Aufgabe 23**

```
SELECT Versandfirmen.*, Bestellungen.Bestell_Nr, Bestellungen.Bestimmungsland
FROM Versandfirmen, Bestellungen
WHERE Versandfirmen.VFirma_Nr = Bestellungen.VFirma_Nr;
```

**Aufgabe 24**

```
SELECT *
FROM Artikel
WHERE Artikel.Kategorie_Nr = 3 AND Artikel.Lagerbestand > 30;
```

**Aufgabe 25**

```
SELECT Kunden.Firma, Kunden.Kontaktperson, Kunden.Position, Kunden.Strasse,
Kunden.Ort, Kunden.Land, Kunden.Telefon, Kunden.Telefax
FROM Kunden
WHERE Kunden.Position LIKE "Inhaber**";
```

**Aufgabe 26**

```
SELECT Kunden.Kunden_Nr, Kunden.Firma, Kunden.Kontaktperson, Kunden.Position,
Kunden.Ort, Kunden.Telefon
FROM Kunden
WHERE Kunden.Position NOT LIKE "Inhaber**";
```

### Aufgabe 27

**SELECT** Bestellungen.Bestell\_Nr, Bestellungen.Kunden\_Nr, Bestellungen.Empfänger,  
Bestellungen.Frachtkosten

**FROM** Bestellungen

**WHERE** Bestellungen.Bestell\_Nr between 10100 **AND** 10200;

*oder*

**SELECT** Bestellungen.Bestell\_Nr, Bestellungen.Kunden\_Nr, Bestellungen.Empfänger,  
Bestellungen.Frachtkosten

**FROM** Bestellungen

**WHERE** Bestellungen.Bestell\_Nr >= 10100 **AND** Bestellungen.Bestell\_Nr <= 10200;

### Parameterabfrage oder flexible Abfrage

**SELECT** \*

**FROM** Kunden

**WHERE** Ort = [Ort eingeben];

*\*\*\*Text in der eckigen Klammer wird in einem Dialogfenster angezeigt.*

*Alternativ kann auch LIKE verwendet werden. Anwender gibt dann im Dialogfenster die entsprechenden Platzhalter ein.*

### Abfrage mit berechnetem Feld (Einzelpreis \* Lagerbestand).

**SELECT** Artikel\_Nr, Artikelname, Einzelpreis \* Lagerbestand **AS** Lagerwert,  
Einzelpreis, Lagerbestand

**FROM** Artikel;

*\*\*\*Einzelpreis \* Lagerbestand berechnen und ausgeben in einer neuen Tabelle namens Lagerwert.*

*Lagerwert ist der Alias-Name für dieses Feld. Das berechnete Feld existiert nur in dieser Abfrage.*

### Aggregatfunktion

→ liefern einen Wert aus einer Gruppe von Datensätzen.

MIN()

MAX()

SUM()

COUNT()

AVG() → Average → Durchschnitt

**SELECT** MAX(Lagerbestand) **AS** Größter\_Wert

**FROM** Artikel;

*\*\*\*Diese Abfrage liefert den größten Lagerbestand aus der Tabelle Artikel*

Aggregatfunktionen (Max, Min, Sum, Count, Avg) sind nur in der SELECT-Anweisung zulässig. Sie dürfen nicht in der WHERE-Klausel stehen!!!

Bei Einsatz einer Aggregatfunktion darf die SELECT-Anweisung nur noch Felder enthalten, die Bestandteil einer Aggregatfunktion sind.

Ausnahme: Die Felder sind Teil einer Gruppierungsanweisung.

Bsp.: Nicht zulässige Abfrage

**SELECT** Artikel\_Nr, MAX(Lagerbestand)

**FROM** Artikel;

*\*\*\*Fehlermeldung: Artikel\_Nr ist nicht Bestandteil einer Aggregatfunktion*



```

SELECT Artikel_Nr, Artikelname
FROM Artikel
WHERE Lieferanten_Nr = (SELECT Lieferanten_Nr
 FROM Lieferanten
 WHERE Firma = „Exotic Liquids“);

```

\*\*\*Alternative zur vorherigen Abfrage mit zwei Tabellen. Subselect kann hier eingesetzt werden, da nur aus einer Tabelle Felder angezeigt werden sollen.

Artikel\_Nr, Artikelname  
 der Artikel mit dem kleinsten Einzelpreis aber nur aus Kategorie „Getränke“

```

SELECT Artikel_Nr, Artikelname, Einzelpreis
FROM Artikel, Kategorien
WHERE Artikel.Kategorie_Nr = Kategorien.Kategorie_Nr
AND (Einzelpreis = (SELECT MIN(Einzelpreis) FROM
Artikel WHERE Kategorie_Nr = (SELECT Kategorie_Nr
FROM Kategorien WHERE Kategorie_Bez =
"Getränke")) AND Kategorie_Bez = "Getränke"
OR Einzelpreis = (SELECT MIN(Einzelpreis) FROM
Artikel WHERE Kategorie_Nr = (SELECT Kategorie_Nr FROM Kategorien
WHERE Kategorie_Bez = "Gewürze")) AND Kategorie_Bez = "Gewürze");

```

\*\*\*Abfrage liefert Artikel\_Nr, Artikelname, Einzelpreis der Artikel die den kleinsten Einzelpreis aus der Kategorie „Getränke“ bzw. „Gewürze“ haben.

Bestell\_Nr, Kunden\_Nr, Firma (aus Tabelle Kunden), Bestelldatum  
 von den Sätzen mit überdurchschnittlichen Frachtkosten  
 SELECT Bestellungen.Bestell\_Nr, Kunden.Kunden\_Nr, Kunden.Firma,  
 Bestellungen.Bestelldatum  
 FROM Kunden, Bestellungen  
 WHERE Bestellungen.Kunden\_Nr = Kunden.Kunden\_Nr AND Frachtkosten >  
 (SELECT AVG(Frachtkosten) FROM Bestellungen);

\*\*\*Abfrage liefert Bestell\_Nr, Kunden\_Nr, Firma und Bestelldatum der Bestellungen mit überdurchschnittlichen Frachtkosten.

## Übung 12

1.

| Name   | Vorname | Alter | Gehalt | Abt |
|--------|---------|-------|--------|-----|
| Bauer  | Gustav  | 33    | 5000   | A20 |
| Meier  | Gustav  | 26    | 5400   | A30 |
| Müller | Paula   | 37    | 4700   | A10 |

| Name  | Vorname | Alter |
|-------|---------|-------|
| Bauer | Gustav  | 33    |
|       |         |       |
| Huber | Hugo    | 41    |

$$26 + 33 = 59 / 2 = 29$$

| Name   | Abt | Gehalt | AX | CX |
|--------|-----|--------|----|----|
| Bauer  | A20 | 5000   | 11 | 7  |
| Meier  | A30 | 5400   | 10 | 7  |
| Müller | A10 | 4700   | 17 | 9  |

| Gesamtgehalt |
|--------------|
| 10400        |
| 4900         |
| 5200         |
| 4700         |

| Vorname |
|---------|
| Paula   |
| Gustav  |
| Gustav  |

| Name  | Vorname | Alter |
|-------|---------|-------|
| Meier | Gustav  | 26    |

**2. a)**

```
SELECT AVG(CX) AS Durchschnittswert
FROM Tab 2
GROUP BY BX;
```

**b)**

```
SELECT Abt, SUM(Gehalt) AS Gesamtgehalt
FROM Tab1
GROUP BY Abt
ORDER BY Abt;
```

**c)**

```
SELECT *
FROM Tab1, Tab2
WHERE BX = Name
AND Name = "Bauer";
```

## Übung 13

### Aufgabe 1

```
SELECT Versandfirmen.VFirma_Nr, Versandfirmen.Firma, SUM(Frachtkosten) AS
Gesamtkosten
FROM Bestellungen, Versandfirmen
WHERE Versandfirmen.VFirma_Nr = Bestellungen.VFirma_Nr
GROUP BY Versandfirmen.Firma, Versandfirmen.VFirma_Nr;
```

### Aufgabe 2

```
SELECT Einzelpreis * Anzahl * 1.16 AS Endpreis, *
FROM Bestelldetails
WHERE Anzahl <= [Bestellmenge eingeben];
```

### Aufgabe 3

```
SELECT *
FROM Kategorien
WHERE Kategorie_Nr = [Kategorie Nummer eingeben];
```

### Aufgabe 4

*oder*

*Funktioniert hier aber nur, weil es keine wirklichen Gruppierungen gibt:*

```
SELECT Artikel_Nr, Artikelname, Lagerbestand, Bestellte_Einheiten,
SUM(Lagerbestand) + SUM(Bestellte_Einheiten) AS Neubestand
FROM Artikel
GROUP BY Artikel_Nr, Artikelname, Lagerbestand, Bestellte_Einheiten;
```

### Aufgabe 5

## Data Manipulation Language (DML-Befehle)

### → Löschen eines Datensatzes

|                  |                          |
|------------------|--------------------------|
| Delete           | Delete                   |
| From Tabelle     | From Bestelldetails      |
| Where Bedingung; | Where Bestell_Nr = 11077 |

### → löscht ganze Datensätze

### → Ändern eines Datensatzes

|                     |                              |
|---------------------|------------------------------|
| Update Tabelle      | Update Personal              |
| Set Feld = Ausdruck | Set Nachname = „Hinterhuber“ |
| Where Bedingung     | Where                        |

### → ändert Feldinhalte

### → Einfügen eines Datensatzes

- |                                                                       |   |                                                                                                   |
|-----------------------------------------------------------------------|---|---------------------------------------------------------------------------------------------------|
| 1. Variante                                                           | } | Alle Felder des Datensatzes werden mit Werten belegt                                              |
| Insert into Tabelle<br>Values (Wert1, Wert2, ...)                     |   |                                                                                                   |
| 2. Variante                                                           | } | Reihenfolge der Felder in der Tabelle ist nicht bekannt, oder es werden nicht alle Felder belegt! |
| Insert into Tabelle (Feld1, Feld2, ...)<br>Values (Wert1, Wert2, ...) |   |                                                                                                   |

### → fügt ganze Datensätze ein!

## Aufgaben:

### **Einfügen von Sätzen**

#### Kunden

|         |                |
|---------|----------------|
| KdNr    | VORHU          |
| Firma   | Vorderhuber KG |
| Str     | Dorfweg 14     |
| PLZ     | 86167          |
| Ort     | Augsburg       |
| Region  | BAY            |
| Land    | Deutschland    |
| Telefon | 0821/223344    |
| Telefax | 0821/556677    |

INSERT INTO Kunden ( Kunden\_Nr, Firma, Strasse, PLZ, Ort, Region, Land, Telefon, Telefax )

VALUES ("VORHU", "Vorderhuber KG", "Dorfweg 14", 86167, "Augsburg", "BAY", "Deutschland", "0821/223344", "0821/556677");

oder

INSERT INTO Kunden

VALUES ("VORHU", "Vorderhuber KG", NULL, NULL, "Dorfweg 14", "Augsburg", "BAY", 86167, "Deutschland", "0821/223344", "0821/556677");

*\*\*\*Reihenfolge der Werte entspricht der Reihenfolge der Felder in der Tabelle- Die beiden nicht belegten Felder werden mit NULL-Werten gefüllt.*

## Ändern des obigen Datensatzes

Kontaktperson Mittelhuber  
Position Manager

```
UPDATE Kunden
SET Kontaktperson = "Mittelhuber", Position = "Manager"
WHERE Kunden_Nr = "VORHU";
```

*\*\*\*Die fehlenden Werte der vorhergehenden Insert-Anweisung werden mit der Update-Anweisung nachträglich hinzugefügt.*

### Aufgabe:

Mitarbeiter mit Nr 4 teilt mit, dass seine private Telefonnummer zu löschen ist.

```
UPDATE Personal
SET Telefon_Priv = NULL
WHERE Personal_Nr = 4;
```

*\*\*\*Der Inhalt des Feldes Telefon\_Priv im Datensatz mit der Personal\_Nr 4 wird gelöscht. D.h. mit einem NULL-Wert überschrieben.*

## Data Definition Language (DDL-Befehle)

### → Erstellen einer Tabelle

optional  
Create table Tabelle (Feld Datentyp not null unique primary key, Feld Datentyp ..., ...,Feld Datentyp References Tabelle (Feld));  
Tabelle (Fekd));

→ erzeugt Tabelle mit Primärschlüssel und Fremdschlüssel (incl. ref. Integ.)  
not null = Pflichtfeld → darf nicht leer sein  
unique = muss eindeutig sein, darf keine Duplikate enthalten

### Aufgabe:

Tabelle Auftrag  
Anr Text 3 Zeichen → Primärschlüssel  
Abez Text 40 Zeichen → Pflichtfeld  
Awert Nachkommastellen möglich  
Adatum Datumsfeld  
Bearbeiter → Feld Personal\_Nr aus Tabelle Personal

```
CREATE TABLE Auftrag(ANr CHAR(3) PRIMARY KEY, ABez CHAR(40) NOT NULL,
AWert DOUBLE, ADatum DATE, Bearbeiter DOUBLE REFERENCES Personal
(Personal_Nr));
```

### Aufgabe:

```
Tabelle Auftrag löschen
DROP TABLE Auftrag;
```

## Übung 14

### Aufgabe 1

```
SELECT Nachname, Gehalt
FROM Personal
WHERE Taetigkeit = „Manager“;
```

### Aufgabe 2

```
SELECT Persnr
FROM Personal
WHERE SEX = „w“;
```

### Aufgabe 3

```
SELECT
```

## Sichten – Views (DDL-Befehle)

Sichten ist eine besondere Darstellungsform von Daten.  
Stellt die Daten dem Anwender so zurecht, wie er diese benötigt.  
Wird in MS ACCESS nicht unterstützt.

- „Virtuelle“ Tabelle
- Beim Erstellen einer Sicht werden die Felder angegeben, die der Anwender benötigt.  
Für den Anwender erscheint die Sicht wie eine Tabelle.
- Es wird aber keine Kopie der Daten erzeugt.  
Gespeichert wird nur die Struktur der Sicht.  
Beim Ausführen der Sicht wird der aktuelle Datenbestand aus den betreffenden Tabellen ausgewählt.

### Erzeugen einer Sicht

- Create View Viewname  
As  
Select Feld, ...  
From Tabelle

Bsp.:

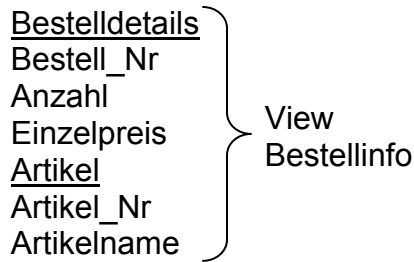
```
Create View Artikel_Kategorie
As
Select Artikel_Nr, Artikelname, Kategorien.Kategorie_Nr, Kategorie_Bez
From Artikel, Kategorien
Where Artikel.Kategorie_Nr = Kategorien.Kategorie_Nr;
```

### Zugriff auf die View:

Bsp.:

```
Select Artikel_Nr, Artikelname, Kategorie_Bez
From Artikel_Kategorie;
```

----- = Viewname



Create View Bestellinfo

As

Select Bestell\_Nr, Anzahl, Bestelldetails.Einzelpreis, Artikel, Artikel.Artikel\_Nr, Artikelname

From Bestelldetails, Artikel

Where Artikel.Artikel\_Nr = Bestelldetails.Artikel\_Nr;

**Aus der ACCESS-Hilfe:**

### **CREATE VIEW-Anweisung**

Erstellt eine neue [Ansicht](#).

### **Ansicht**

Ansicht ist die ANSI-Definition einer virtuellen Tabelle. Eine Ansicht ist ein Synonym für eine gespeicherte Abfrage unter Access, die aus einer SELECT-Anweisung ohne Parameter besteht. Eine Ansicht ist eine gespeicherte Definition, die bei Ausführung realisiert wird. Der Unterschied zu einer Tabelle besteht darin, dass physikalisch keine Daten gespeichert, sondern Daten nur zurückgegeben werden.

Eine Ansicht kann mit der neuen ANSI SQL CREATE VIEW-Syntax erstellt werden, auf die über Microsoft® OLE DB Provider for Jet zugegriffen werden kann. Alternativ kann auch eine Prozedur erstellt werden, die SQL DML-Anweisungen und SELECT-Anweisungen mit Parametern unterstützt. Dies wird mit der neuen CREATE PROCEDURE-Syntax realisiert, auf die über Microsoft OLE DB Provider for Jet zugegriffen werden kann.

## Rechte (DCL-Befehle)

Wird in MS ACCESS nicht unterstützt.

### **Grant:**

Gewährt einem vorhandenen Benutzer oder einer Gruppe bestimmte Berechtigungen.

### Rechtevergabe (Grant)

Objektberechtigungen

- Select
- Update
- Delete
- Insert

Grant Recht, Recht ...  
On Tabelle | View  
To User, User ...

#### **Bsp.:**

Grant Select, Delete  
On Kunden  
to Hugo, Berta;

#### ***Alle Befehle für eine Person***

Grant all  
On Personal  
to Berta

#### ***Bestimmte Befehle für alle***

Grant Select, Delete  
On Kunden  
to Public

### ***Gustav darf das Recht für den Select-Befehl an andere weitergeben:***

Grant Select  
On Artikel  
to Gustav  
with grant option

Grant update, delete  
On Artikel(Lagerbestand)  
to Paula;

### Rechteentzug (Revoke)

Revoke Recht, Recht ...  
On Tabelle | View  
From User, User ...

Revoke update  
On Artikel  
From Paula;

### ***Rechteentzug für Berta für die Select-Anweisung***

Revoke grant option for select  
On Artikel  
From Berta;

### Aufgabe:

In einer Schulbibliotheks-DB erteilt dem DBA dem Lehrer\_A das Recht, die Tabelle Ausleihe zu lesen.

Befehl: Grant Select on Ausleihe to Lehrer\_A  
With grant option;

Lehrer\_A gibt seine Rechte an Lehrer\_B weiter.

Lehrer\_B gibt das Select-Recht an die Schüler X, Y, Z weiter.

Wer hat welche Rechte?

| Person   | Rechte             |
|----------|--------------------|
| Lehrer_A | Select, Weitergabe |
| Lehrer_B | Select, Weitergabe |
| X, Y, Z  | Select             |
| DBA      | alle               |

Darf Lehrer\_A das Update-Recht weitergeben? Begründen Sie Ihre Entscheidung:  
Nein, er hat das Update-Recht nicht.

### **Übung 15:**

#### **Aufgabe I:**

1.

Create View Inventureingabe

As

Select Artikel\_Nr, Artikelname, Auslaufartikel, Mindestbestand, Lagerbestand  
From Artikel;

2.

Create View KundenGB

As

Select \*

From Kunden

Where Land Like "Großbritannien\*"

3.

Create View Nachbestellung

As

Select Lieferanten.Firma, Artikel.\*

From Lieferanten, Artikel

Where Lieferanten.Lieferanten\_Nr = Artikel.Lieferanten\_Nr

And Artikel.Lagerbestand < Artikel.Minderbestand;

4.

Create View VersandSpeedy

As

Select Bestellungen.\*

From Versandfirmen, Bestellungen

Where Bestellungen.VFirma\_Nr = Versandfirmen.VFirma\_Nr

And Versandfirmen.Firma Like „Speedy Express“;

#### **Aufgabe II:**

1.

Grant Select

On Artikel

To Davolio;

2.

Grant Select

On KundenGB

To public;

3.

Grant all

On Inventureingabe

To Fuller;

4.

Grant Select, Delete

On Versandfirmen

To King

With grand option;

**Aufgabe III:**

Create Table Hardware(HardwareNr CHAR(4) primary key, Bezeichnung CHAR(50), Anschaffungsdatum DATE, Kaufpreis DOUBLE, Bemerkung CHAR(50));

**Aufgabe IV:**

Revoke all  
On Inventureingabe  
From Fuller;

**Wiederholungsfragen**

Definieren Sie **KURZ** die folgenden Begriffe:

**Datenbank:**

Organisation von Daten, die auf eine bestimmte Weise gespeichert und organisiert sind.

**Dateisystem (Vergleich mit Datenbanksystem):**

Bei Änderung der Struktur muss Datei und Programm geändert werden → Programmabhängigkeit.

Daten werden in Dateien gespeichert und von Anwendungsprogrammen verwaltet.

- Redundanzen
- Gefahr von Inkonsistenzen
- Daten-Programm-Abhängigkeit
- Inflexibilität
- Keine zentrale Zugriffskontrolle
- Keine gemeinsame Datenbasis

**Datenbanksystem (DBS) (Nennen Sie einige Eigenschaften/Aufgaben)**

Datenbanksystem ist Datenbank-Management-System und Datenbank.

Eigenschaften: Seite 4

**Datenbankmanagementsystem (DBMS)**

Schnittstelle zwischen Datenbank und Anwendungsprogramm.

**Redundanz**

Wenn ein und dieselbe Information mehrfach gespeichert wird.

**Inkonsistenz**

Ein und dieselbe Information wird mit unterschiedlichen Werten gespeichert.

**Konsistenz**

Einheitlichkeit der Daten. Widerspruchsfreie Daten

**Integrität**

Korrektheit / Vollständigkeit der Daten.

**Integritätsbedingung**

Gewährleistet die Integrität, wie z.B. Gültigkeitsbedingungen - Datenbankregeln.

**Sicht**

Benutzerdefinierte Darstellung der Daten – virtuelle Tabelle. Dient dem Komfort und dem Datenschutz.

### **Daten-Programm-Abhängigkeit**

Datei arbeitet mit dem bestimmten Anwendungsprogramm zusammen.  
Anwendungsprogramm greift direkt auf die Daten zu / erzeugt direkt die Daten.  
Die Struktur der Datei ist im Anwendungsprogramm direkt hinterlegt.  
Folgen: wenig flexibel, hoher Aufwand bei Änderungen

### **Feld**

Teil eines Datensatzes; besteht aus einem oder mehreren Zeichen

### **Satz**

besteht aus einem oder mehreren Feldern; Teil einer Datei; beschreibt ein Objekt

### **Datei**

Strukturierte Sammlung von Daten.

### **Relationales Modell**

Speicherung der Daten in Form von Tabellen. Zwischen den Tabellen bestehen Beziehungen. Zeichnet sich aus durch hohe Flexibilität, Ausbaufähigkeit, leichtes Erstellen von Abfragen, etc.

### **ER-Modell**

Entity Relationship Model – grafische Form des logischen Datenbankentwurfs

### **Relation**

Eine Tabelle in einer relationalen Datenbank.

Jede Tabelle hat einen eindeutigen Namen.

### **Tupel**

Datensatz, Zeile, Reihe

Beschreibt eine Entität oder ein reales Objekt. Kann aber auch eine Transaktion bzw. einen Vorgang beschreiben.

### **Attribut**

Eigenschaft eines Objektes oder einer Entität –

Feld oder Spalte in einer Tabelle

### **Attributswert**

Der Feldinhalt einer Eigenschaft, z.B. Attribut: Strasse; Attributswert: Dorfweg 17

### **Primärschlüssel**

Identifiziert einen Datensatz eindeutig.

Dient der Herstellung oder Beziehung zwischen zwei Tabellen.

### **Fremdschlüssel**

Dient dem herstellen von Beziehungen zwischen dem Primärschlüssel auf eine andere Tabelle. Darf nur Werte des betreffenden Primärschlüsselfeldes enthalten.

### **Beziehung**

Verknüpfung / logische Verbindung zwischen zwei Tabellen.

### **1:1 – Beziehung**

Ein Datensatz in der einen Tabelle steht in Beziehung zu einem Datensatz in der anderen Tabelle.

### **1:n – Beziehung**

Ein Datensatz in der einen Tabelle steht in Beziehung zu vielen Datensätzen in der anderen Tabelle.

### **m:n – Beziehung**

Jeder Datensatz in der Tabelle steht vielen Datensätzen in der anderen Tabelle gegenüber.

### **referentielle Integrität**

Prüfung der Gültigkeit der Beziehung zwischen zwei Tabellen.

Nicht mehr gültig: wenn das Fremdschlüsselfeld andere Werte beinhaltet, als das übergeordnete Primärschlüsselfeld – z.B. beim Löschen von Datensätzen in der übergeordneten Tabelle → Löschanomalien.

### **Aktualisierungsweitergabe**

Änderungen im Primärschlüsselfeld bewirken automatisch eine Änderung der Fremdschlüsselwerte in der untergeordneten Tabelle.

### **Löschweitergabe**

Löschen eines Datensatzes in der übergeordneten Tabelle bewirken automatisch das Löschen der in Beziehung stehenden Datensätze der untergeordneten Tabelle.

### **Relationenschema**

Darstellung der Tabellen mit allen Primärschlüsseln, Fremdschlüsseln, Feldern, realisierte Beziehungen.

### **Aggregatfunktion**

Liefert einen Wert aus einer Gruppe von Datensätzen.

### **Gruppierung**

Zusammenfassung von Datensätzen nach einem bestimmten Gruppenkriterium.

### **Abfragen**

Gezielter Auszug von Daten.

### **SQL**

Abfragesprache für relationale Datenbanken.

### **DDL**

Befehle zum Definieren von Datensätzen.

### **DML**

Befehle zum Manipulieren von Datensätzen.

### **DCL**

Befehle zum kontrollierten Zugriff auf die Datenbank.

## Normalisierung - Normalformen

**Ziel:** Redundanzen und unerwünschte Abhängigkeiten aus einer Datenbank (Tabellen) weitgehend zu entfernen!

Für Praxis wichtig: 1. – 3. Normalform

### 1. Normalform

- - Tabelle enthält Felder und Reihen
- alle Felder sind atomar (d.h. sie weisen keine innere Struktur auf)

Tabelle befindet sich nicht in der 1. Normalform:

Personal (PNr, Name, Vorname, Adresse, Tel)

Tabelle in der 1. Normalform:

Personal (PNr, Name, Vorname, Str, PLZ, Ort, Tel)



### 2. Normalform

- 1. Normalform ist erfüllt
- Alle Nicht-Schlüselfelder sind ausschließlich vom kompletten Primärschlüssel abhängig, und nicht nur von einem Teil des Schlüssels.

2. Normalform nicht erfüllt:

Projekt\_Personal (ProjNr, PersNr, Beginn, Ende, Pers\_Name, ProjBezeichnung)

2. Normalform erfüllt:

Projekt\_Personal (ProjNr, PersNr, Beginn, Ende)

Personal (PersNr, Pers\_Name)

Projekt (ProjNr, ProjBez)

### 3. Normalform

- 1. + 2. Normalform nicht erfüllt
- Alle Nicht-Schlüselfelder sind ausschließlich vom Primärschlüssel abhängig und nicht von einem anderen Nicht-Schlüsselfeld

3. Normalform nicht erfüllt:

Personal (PNr, Name, Vorname, Str, PLZ, Ort, AbtNr, Abtname)

3. Normalform erfüllt:

Personal (PNr, Name, Vorname, Str, PLZ, Ort, AbtNr)

Abteilung (AbtNr, Abtname)

## Makros in Access

Makros werden in der Praxis selten angewendet!!!  
Das wird in der Regel selbst programmiert.

Access – Formulare – Erstellt ein Formular unter Verwendung des Assistenten –  
Tabelle: Artikel – Alle Felder auswählen – Fertigstellen

Datei schließen

Berichte – Erstellt einen Bericht unter Verwendung des Assistenten – Tabelle: Artikel –  
Ausgewählte Felder: Artikel\_Nr, Artikelname, Lagerbestand – Weiter – Weiter – Layout:  
einspaltig – Bei „Feldbreite so anpassen, dass alle Felder auf eine Seite passen“ das  
Häckchen herausnehmen, da er dann einfach alles abschneidet, um es auf eine Seite  
zu bekommen!!!! – Fertigstellen – Schließen

Formulare – Entwurfsansicht von Tabelle: Artikel –  
In der Toolbox Zauberstab ausklicken (Steuerelement-Assistenten), da sonst immer der  
Assistent angezeigt wird. – Befehlsschaltfläche auswählen – Rechte Maustaste –  
Eigenschaften – Registerblatt: Format – Beschriftung: Bericht aufrufen – Registerblatt:  
Andere – Name: cmdBericht – Registerblatt: Ereignis – Beim Klicken: ... anklicken –  
Makro-Generator – Makroname: Bericht aufrufen – Aktion: ÖffnenBericht –  
Aktionsargumente - Berichtsname: Artikel – Ansicht: Seitenansicht - Bedingung:  
Artikel\_Nr = Formulare![Artikel]![Artikel\_Nr] → Holt sich aus der Tabelle: Artikel die  
Sätze heraus, die gerade angezeigt werden und zeigt den Artikel mit der  
Artikelnummer an – Schließen

Auf Ansicht wechseln – Bericht aufrufen -

Wieder zum Makro-Generator wechseln – Neue Zeile am Anfang einfügen – Aktion:  
Meldung – Aktionsargumente – Meldung: Achtung! Bericht wird nicht sofort gedruckt. –  
Typ: Information – Titel: Hinweis

Formulare – Neue Befehlsschaltfläche – Eigenschaften – Registerblatt: Format -  
Beschriftung: Nächster Satz – Registerblatt: Andere – Name: cmdNext – Registerblatt:  
Ereignis – Beim Klicken: ... anklicken – Makro-Generator – Aktion: GeheZuDatensatz –  
Ansicht – Bedingungen – Aktion: Meldung – Aktionsargumente – Meldung: Achtung  
nachbestellen – Typ: Kritisch – Titel: Bestellen!!! - Bedingung:  
[Lagerbestand]<[Mindestbestand]

### **Für eigene Anwendungen:**

Extras – Start – Einstellen, was der Benutzer sehen / nicht sehen soll  
SHIFT gedrückt halten und MDB-Datei anklicken, um wieder in die Entwickler-Fassung  
zu kommen, wenn Startoptionen eingerichtet wurden!!





## Glossar:

### **Datenbank**

(Datenbank)

Bezeichnung für Programme, mit denen Informationen wie z. B. Adressen oder Warenbestände erfaßt, verwaltet und selektiv gesucht werden können.

Informationen werden in Datenbanken nach fester Struktur geordnet. Dabei werden die Daten in Datensätze (z. B. eine Adresse) zusammengefaßt, die durch Felder (z. B. Nachname) gekennzeichnet und gegliedert sind.

Beim am weitesten verbreiteten Typ der relationalen Datenbank werden die Felder und Datensätze in Tabellen geordnet, die außerdem miteinander verknüpft werden können. Relationale Datenbanken haben durch diese Struktur daher auch Ähnlichkeit mit Tabellenkalkulationen. Durch dieses Prinzip können verschiedenen Datendateien auch einfach miteinander verknüpft werden.

Datenbanken dienen vor allem zur schnellen Abfrage aus großen

Informationsbeständen in Form von Reports oder Berichten. Diese Abfragen werden mittlerweile meist über standardisierte Protokolle wie SQL oder ODBC durchgeführt, die auch einen Zugriff auf die Daten von anderen Programmen wie

z. B. Textverarbeitungen erlauben. Bekannte Datenbanken sind dBase von Borland, FoxPro, MS-Access und Lotus-Approach, Clipper von Nantucket, Bestandteil von integrierten Programmpaketen

### **Datenbank**, die; *Subst.* (data bank, database)

Im weiteren Sinn jede wesentliche Datensammlung.

Im engeren Sinn eine Datei, die aus Datensätzen besteht, die jeweils aus Feldern aufgebaut ist. Zu einer Datenbank gehören weiterhin Operationen zum Suchen, Sortieren, Bilden neuer Kombinationen und andere Funktionen.

### **Datensatz**, der; *Subst.* (data record, record)

Eine Datenstruktur, die eine Sammlung von Feldern (Elementen) darstellt, von denen jedes einen eigenen Namen und Typ aufweist. Im Gegensatz zu einem Array, dessen Elemente alle dem gleichen Datentyp angehören und über einen Index angesprochen werden, repräsentieren die Elemente eines Datensatzes verschiedene Datentypen, und der Zugriff erfolgt über ihre Namen. Auf einen Datensatz kann man sowohl in seiner Gesamtheit als auch durch Referenzierung einzelner Elemente zugreifen.

### **hierarchische Datenbank**, die; *Subst.* (hierarchical database)

Eine Datenbank, in der die Anordnung der Datensätze eine verzweigte, baumartige Struktur bildet. Diese Form, die am häufigsten bei Datenbanken für größere Computer verwendet wird, eignet sich besonders für die Organisation von Informationen, bei denen sich eine logische Untergliederung in sukzessive größere Detailebenen anbietet. Dabei sollte die Organisation der Datensätze im Hinblick auf die gebräuchlichsten oder zeitkritischsten Arten des erwarteten Zugriffs erfolgen.

### **objektorientierte Datenbank**, die; *Subst.* (object-oriented database)

Ein flexibler Datenbanktyp, der den Einsatz von abstrakten Datentypen, Objekten sowie Klassen unterstützt und eine Vielzahl unterschiedlicher Datenarten speichern kann, neben Texten und Zahlen auch Klänge, Videos und Grafiken. Einige objektorientierte Datenbanken erlauben es, Datenrückgewinnungsprozeduren und Datenverarbeitungsregeln zusammen mit den Daten oder anstelle der Daten zu speichern. Auf diese Weise können Daten außerhalb der physikalischen Datenbank untergebracht werden, was häufig wünschenswert ist, wenn die Dateien sehr groß werden, beispielsweise in Verbindung mit Videodateien.

### **Relation**

(*Computer, Datenbank*)

Lateinisch und englisch für Beziehung, Verhältnis, Verwandtschaft. Allgemeiner Begriff für alle Formen von Beziehungen und Abhängigkeiten von Objekten oder Daten zueinander. Spielt eine besondere Rolle bei den sogenannten ['relationalen Datenbanken'](#).

Dort können die in Form von Spalten und Zeilen (Tabellen) gespeicherten Daten untereinander verknüpft werden. Das ermöglicht es, neue kombinierte Datensätze mit Informationen aus verschiedenen Tabellen (i.d.R. auch Dateien) zu erstellen, eine Neueingabe der bereits vorhandenen Daten entfällt dabei. Verknüpft werden nicht die ganzen Tabellen, sondern nur jeweils einzelne Felder der Spalten. Beispielsweise könnte eine Auftragsverwaltung aus einer Kundentabelle (Adreßdaten) und einer Auftragsstabelle bestehen, in der jeder Kunde mit seiner Kundennummer gekennzeichnet ist.

Die Spalte 'Kundennummer' kommt in beiden Dateien vor und erlaubt daher eine Relation der beiden Tabellen. Durch diese Verknüpfung kann ein neuer Datensatz als Rechnungsformular erstellt werden, der die Daten aus beiden Tabellen automatisch vereinigt.

Bei dieser Form von Relation unterscheidet man immer zwischen einer 'Ursprungstabelle' (Kundendatei) und einer 'Detailtabelle' (Auftragsliste), zu der die Verknüpfung aufgebaut wird. Man verwendet dabei insbesondere zwei verschiedene Relations-Typen: Bei den 1:1-Relationen hat jeder Datensatz aus der Ursprungstabelle nur eine Verknüpfung zu einem Datensatz der Detailtabelle. Sind mehr als eine Verknüpfung erlaubt, so spricht man von einer 1:N-Relation. Da jeder Kunde mehrere Bestellungen durchführen kann, würde hier eine 1:N-Relation angewendet werden.

### **Relation**, die; *Subst.* (relation)

Eine Struktur des relationalen Datenbankmodells, die sich aus Attributen und Tupeln aufbaut. Relationen werden in relationalen Datenbank-Managementsystemen als Tabellen gespeichert. Attribute (Spalten) sind individuelle Kennzeichen, und Tupel (Zeilen) bilden die ungeordneten Kennzeichensätze, die eine bestimmte Entität (beispielsweise einen Kunden) beschreiben. Innerhalb einer Relation können Tupel nicht wiederholt werden – sie müssen eineindeutig sein. Weiterhin sind Tupel innerhalb einer Relation ungeordnet. Der Austausch zweier Tupel ändert nicht die Relation. Wenn schließlich die relationale Theorie anwendbar sein soll, muss die Domäne jedes Attributes atomisch sein - d. h. strukturierte Domänen (Arrays, Datensätze usw.) sind nicht erlaubt. Eine Relation, in der die Domänen aller Attribute atomisch sind, charakterisiert man als normalisiert oder in der ersten Normalform.

### **relationale Datenbank**, die; *Subst.* (relational database)

Die Organisation einer Datenbank oder eines Datenbank-Managementsystem nach dem relationalen Modell. Danach sind die Informationen in Tabellen - Daten in Zeilen und Spalten - gespeichert. Für Suchoperationen verwendet man Daten in spezifizierten Spalten einer Tabelle, um zusätzliche Daten in einer anderen Tabelle zu ermitteln. In einer relationalen Datenbank stellen die Zeilen einer Tabelle die Datensätze (Sammlungen von Informationen über separate Elemente) und die Spalten die Felder (besondere Attribute eines Datensatzes) dar. Bei Suchoperationen vergleicht eine relationale Datenbank die Informationen eines Feldes in der einen Tabelle mit Informationen in einem korrespondierenden Feld einer anderen Tabelle, um eine dritte Tabelle zu produzieren, die die angeforderten Daten aus beiden Tabellen kombiniert. Enthält eine Tabelle z. B. die Felder PERSONALNUMMER, NACHNAME, VORNAME und EINSTELLUNGSDATUM und eine andere die Felder ABTEILUNG, PERSONALNUMMER und GEHALT, dann kann eine relationale Datenbank die Felder PERSONALNUMMER in beiden Tabellen vergleichen, um solche Informationen wie die Namen aller Beschäftigten mit einem bestimmten Gehalt oder die Abteilungen aller Beschäftigten mit einem bestimmten Einstellungsdatum zu finden. Eine relationale Datenbank verwendet also übereinstimmende Werte in zwei Tabellen, um die Informationen einer Tabelle mit den Informationen in der anderen in Verbindung zu bringen. Bei einem Großteil der gegenwärtig angebotenen Datenbankprodukte für Mikrocomputer handelt es sich um relationale Datenbanken.

### **Relationale Datenbank** (*Datenbank*)

[Datenbank](#), die Daten in verknüpfbaren Tabellen speichert.

**Redundanz** = Ein und dieselbe Information wird mehrfach gespeichert.

(Computer)

Anderes Wort für Überfluss, zusätzlich vorhandene und nicht notwendige Information. Redundanz spielt bei der Datensicherheit eine große Rolle, da zusätzliche Information beispielsweise die Fehlertoleranz eines Systems erhöhen kann. Im Computer als Bitmuster gespeicherte Daten sind normalerweise nicht redundant, so dass jede Veränderung nur eines Bits die Veränderung der dazugehörigen Information bedeuten würde. Aus diesem Grund werden bei kritischen Prozessen wie z. B. der Datenfernübertragung mit Absicht redundante Informationen eingefügt (Prüfbits, no parity), die so eine Fehlerkorrektur ermöglichen.

### **Datenbanksystem**

(Datenbank)

Gesamtheit der Programme zur Abfrage, Änderung und Speicherung von Daten (Datenmanipulation) sowie zur Beschreibung der Datenstruktur und zum Aufbau der nötigen Dateien (Datendefinition).

Datenbank-Systeme sind heute für alle gängigen Größenklassen von Computern erhältlich.

**Tupel** = Datensatz

**Tupel**, das; *Subst.* (tuple)

In einer Datenbanktabelle (Relation) ein Satz von zusammengehörigen Werten, die jeweils ein Attribut (Spalte) repräsentieren. Ein Tupel wird in einem relationalen Datenbank-Managementsystem als Zeile gespeichert und ist mit einem Datensatz in einer nicht relationalen Datei vergleichbar.

**Datenmodell**, das; *Subst.* (data model)

Eine Sammlung aufeinander bezogener Objekttypen, Operatoren und Integritätsregeln, die die vom Datenbank-Managementsystem (DBMS) unterstützte Entität bilden. In Abhängigkeit vom jeweils implementierten Datenmodell spricht man daher von einem relationalen DBMS oder einem Netzwerk-DBS usw. Im Allgemeinen unterstützt ein DBMS mehr aus praktischen als aus theoretischen Einschränkungen nur ein Datenmodell.

**Integrität**

(Computer, Datenbank)

Vorhandene Integrität sagt aus, dass alle zu einem System gehörenden Komponenten bzw. die Hardware einwandfrei funktionieren und auftretende Fehler entdeckt werden können. Im Bereich von Daten gewährleistet die Daten-Integrität, dass die Daten dem Zustand, den sie beschreiben, entsprechen und auch während ihrer Verarbeitung diesen Status behalten. So sorgen z. B. bei Datenbanken entsprechende Sicherheitsmaßnahmen (Datensicherheit) dafür, dass, wenn Fehler bei der Bearbeitung auftreten, die bearbeiteten Daten wieder in den Zustand vor der Bearbeitung zurückversetzt werden können.

**Integrität**, die; *Subst.* (integrity)

Die Vollständigkeit und Korrektheit der in einem Computer gespeicherten Daten, insbesondere nachdem sie in einer beliebigen Form manipuliert wurden.

**Datenbank-Managementsystem**, das; *Subst.* (database management system)

Abgekürzt DBMS. Eine Softwareebene zwischen der Datenbank und dem Benutzer. Ein Datenbank-Managementsystem handhabt Anforderungen von Benutzern für Datenbankaktionen und ermöglicht die Kontrolle hinsichtlich Sicherheit und Datenintegrität.

**DBMS**

(*Abk, Netzwerk*)

(**Datenbank-Management-System**); Es handelt sich hierbei um Serverkonzepte, die sich außer der reinen Bereitstellung von Daten auch um sekundäre Verwaltungsaufgaben kümmern ([Zugriffsberechtigungen](#), Transaktionssicherheit, usw.).

**Primärschlüssel**

(*Computer, Datenbank*)

Eine Datenmenge, Tabelle oder Datenbank, kann nach bestimmten Kriterien sortiert werden, man spricht in diesem Zusammenhang auch von Sortierschlüsseln. Der Primärschlüssel ist das erste Sortierkriterium, z. B. könnte eine Datenbank nach Nachnamen in alphabetisch aufsteigender Reihenfolge sortiert werden.

**Primärschlüssel**, der; *Subst.* (primary key)

Auch als Hauptschlüssel bezeichnet. In Datenbanken das Schlüsselfeld, das als eindeutiger Bezeichner eines bestimmten Tupels (Zeile) in einer Relation (Datenbanktabelle) verwendet wird.

**Sekundärschlüssel**, der; *Subst.* (candidate key, secondary key)

Ein eindeutiger Kennzeichner für einen Datensatz (Tupel) in einer Relation (Datenbanktabelle). Der Sekundärschlüssel kann entweder einfach (ein einzelnes Attribut) oder zusammengesetzt (zwei oder mehr Attribute) sein. Per Definition muss jede Relation zumindest über einen Sekundärschlüssel verfügen, wobei aber auch mehrere Sekundärschlüssel vorhanden sein können. Wenn es nur einen Sekundärschlüssel gibt, wird er automatisch zum Primärschlüssel der Relation. Sind mehrere Sekundärschlüssel vorhanden, muss der Entwickler einen davon als Primärschlüssel bestimmen. Jeder Sekundärschlüssel, der nicht als Primärschlüssel festgelegt wurde, stellt einen alternativen Schlüssel dar.

**Schlüssel**, der; *Subst.*

In der Datenbankverwaltung stellt ein Schlüssel einen Bezeichner für einen Datensatz oder eine Gruppe von Datensätzen in einer Datendatei dar.

**Alternativschlüssel, Fremdschlüssel**, der; *Subst.* (alternate key)

Jeder alternative Schlüssel in einer Datenbank, der nicht als Primärschlüssel vorgesehen ist.

**SQL, strukturierte Abfragesprache**, die; *Subst.* (structured query language)

Eine Datenbanksprache zur Abfrage, Aktualisierung und Verwaltung relationaler Datenbanken - eine De-facto-Norm in Datenbankprodukten.

**SQL**

(*Abk., Datenbank*)

(**Structured Query Language**), Deutsch: strukturierte Abfragesprache. Eine von [IBM](#) entwickelte Abfragesprache für relationale [Datenbanksysteme](#). Komplexe Datenbankmanipulationen sind bei SQL durch simple Schlüsselworte unabhängig vom Dateninhalt möglich.

**Bericht**, der; *Subst.* (report)

Die Präsentation von Informationen über ein gegebenes Thema - meist in gedruckter Form. Per Computer mit entsprechender Software aufbereitete Berichte können Text, Grafiken und Diagramme umfassen.

Datenbankprogramme enthalten meist spezielle Funktionen für die Erstellung von Berichtformularen und die Erzeugung von Berichten. Mit Software für Desktop Publishing sowie Laserdruckern bzw. Satzmaschinen lassen sich Ausgaben in veröffentlichungsreifer Druckqualität produzieren.

**Index**, der; *Subst.* (index)

Eine Liste von Schlüsselwörtern mit zugehörigen Daten, die auf die Speicherstellen mit umfangreicheren Informationen zeigen. Dazu gehören z. B. Dateien und Datensätze auf einer Diskette bzw. Schlüsseldatensätze in einer Datenbank.

In der Programmierung bezeichnet »Index« einen Skalarwert für den direkten Zugriff auf eine mehrelementige Datenstruktur (z. B. ein Array), ohne eine sequentielle Suche durch die Sammlung von Elementen durchführen zu müssen.

**Datenmanipulationssprache**, die; *Subst.* (data manipulation language)

Abgekürzt DML. Eine Sprache, die in der Regel Bestandteil eines Datenbank-Managementsystems ist und dazu verwendet wird, Daten einzufügen, zu aktualisieren und eine Datenbank abzufragen. Datenmanipulationssprachen können häufig mathematische und statistische Berechnungen ausführen, die das Generieren von Berichten erleichtern.

**Datendefinitionssprache**, die; *Subst.* (data definition language)

Abgekürzt DDL. Eine Sprache, die für die Definition aller Attribute und Eigenschaften einer Datenbank verwendet wird - insbesondere für Datensatzlayouts, Felddefinitionen, Schlüsselfelder, Dateistandorte und Speicherstrategien.

**descend** [*dɪ'send*] **I. vi** **1.** herab-, hinabsteigen, herabkommen; hinunterfahren; heruntergehen **2.** abstammen (**from** von) **3.** (*Eigentum*) übergehen (**from** von) (**to** auf), vererbt werden **4.** herfallen (**on, upon** über), überfallen (**on s.o.** jdn), hereinbrechen (**on** über) **5.** sich erniedrigen, herablassen (**to** zu) **descend to details** in die Einzelheiten gehen **II. vt** (*Treppe*) hinuntergehen, -steigen

Siehe DESC

**hash**

Ein Befehl in zahlreichen FTP-Clientanwendungsprogrammen, der den FTP-Client anweist, immer dann ein Nummernzeichen (#) anzuzeigen, wenn ein Datenblock gesendet bzw. empfangen wird.

Als »hashen« wird außerdem der Vorgang bezeichnet, bei dem ein Element durch eine Umwandlung als numerischer Wert abgebildet wird (Hashingfunktion). Hashing wird dann verwendet, wenn ein Identifizierer oder Schlüssel, der für den Benutzer relevant ist, in einen Wert für die Position der entsprechenden Daten in einer Struktur (z. B. eine Tabelle) umgewandelt werden soll. Wenn z. B. der Schlüssel MOUSE und eine Hashingfunktion, die den ASCII-Werten der Zeichen hinzugefügt wurde, die Summe durch 127 teilt und den Rest abzieht, erhält MOUSE durch Hashing den Wert 12. Die Daten, die von MOUSE identifiziert werden, befinden sich unter den Elementen in Eintrag 12 der Tabelle.

### **Referentielle Integrität**

Referentielle Integrität ist ein Regelsystem, das Microsoft Access verwendet, um sicherzustellen, dass Beziehungen zwischen Datensätzen in verknüpften Tabellen gültig sind und verknüpfte Daten nicht versehentlich gelöscht oder geändert werden. Aktivieren Sie dieses Kontrollkästchen, wenn Sie für diese Beziehung referentielle Integrität festlegen möchten. Aktivieren Sie es jedoch nur, wenn alle der folgenden Bedingungen vorliegen: das übereinstimmende Feld aus der Primärtabelle ist ein Primärschlüssel oder verfügt über einen eindeutigen Index, die verknüpften Felder haben denselben Datentyp und beide Tabellen sind in derselben Access-Datenbank gespeichert.

### **Aktualisierungsweitergabe**

Wählen Sie „Mit referentieller Integrität“ und dann „Aktualisierungsweitergabe an Detailfeld“, wenn Sie bei jedem Ändern eines Primärschlüsselwerts in der Primärtabelle automatisch die entsprechenden Werte in der verknüpften Tabelle aktualisieren möchten.

Wählen Sie „Mit referentieller Integrität“, und deaktivieren Sie dann „Aktualisierungsweitergabe an Detailfeld“, wenn Sie verhindern möchten, dass Änderungen an einem Primärschlüsselfeld in der Primärtabelle vorgenommen werden, wenn es in der verknüpften Tabelle verknüpfte Datensätze gibt.

### **Löschweitergabe**

Wählen Sie „Mit referentieller Integrität“ und dann „Löschweitergabe an Detaildatensatz“, wenn Sie bei jedem Löschen eines Datensatzes in der Primärtabelle automatisch die verknüpften Datensätze in der verknüpften Tabelle löschen möchten.

Wählen Sie „Mit referentieller Integrität“, und deaktivieren Sie dann „Löschweitergabe an Detaildatensatz“, wenn Sie verhindern möchten, dass Datensätze aus der Primärtabelle gelöscht werden, wenn es in der verknüpften Tabelle verknüpfte Datensätze gibt.

### **Transaktion**, die; *Subst.* (transaction)

Eine in sich abgeschlossene Aktivität innerhalb eines Computersystems, z. B. die Erfassung einer Kundenbestellung oder die Aktualisierung einer Bestandsposition. Transaktionen beziehen sich in der Regel auf Systeme zur Datenbankverwaltung oder Auftragserfassung sowie andere Onlinesysteme.

Die Unterstützung von Transaktionen umfasst dabei Funktionen wie "Commit", "Rollback" und "Crash Recovery", die für eine vollständige Konsistenz der Daten garantieren sollen.

### **normalisieren** *Vb.* (normalize)

In der Programmierung das Anpassen der Festkomma- und Exponentialbestandteile von Gleitkommazahlen, um die Festkommanteile in einen festgelegten Bereich zu bringen.

In der Datenbankverwaltung bezeichnet »normalisieren« die Anwendung einer Reihe von Methoden auf eine relationale Datenbank, um doppelt vorhandene Informationen zu minimieren. Normalisieren vereinfacht in starkem Maße die Behandlung von Abfragen und Aktualisierungen, wozu auch Aspekte der Sicherheit und Integrität zählen. Das Normalisieren wird allerdings mit einer größeren Anzahl von Tabellen erkauft.

**Normalform**, die; *Subst.* (normal form)

In einer relationalen Datenbank eine Methode zur Strukturierung von Informationen. Normalformen vermeiden Redundanz und Inkonsistenz und fördern effiziente Verwaltung, Speicherung und Aktualisierung von Informationen. Dabei werden verschiedene »Regeln« oder Ebenen der Normalisierung gemeinhin anerkannt - jede stellt eine Verfeinerung der vorangegangenen dar. Von diesen werden Formen werden in der Praxis meist drei verwendet: die erste Normalform (1NF), die zweite Normalform (2NF) und die dritte Normalform (3NF). Die ersten Normalformen sind am wenigsten strukturiert und stellen Gruppen von Datensätzen dar (z. B. Angestelltenlisten), in denen jedes Feld (Spalte) eindeutige, also nicht wiederholende Daten enthält. Die zweiten und dritten Normalformen gliedern die ersten Normalformen auf, indem sie diese in unterschiedliche Tabellen aufteilen, wobei der Reihe nach feinere Wechselbeziehungen zwischen den Feldern definiert werden. Die zweiten Normalformen enthalten dabei nur Felder, die Untermengen von Feldern mit Primärschlüssel darstellen. Beispielsweise kann eine zweite Normalform, die einen Schlüssel auf den Namen eines Angestellten aufweist, nicht gleichzeitig den Dienstgrad und den Stundentarif enthalten, wenn die Bezahlung vom Dienstgrad abhängig ist. Die dritten Normalformen enthalten keine Felder, die Informationen über Felder außer dem Schlüsselfeld zur Verfügung stellen. Eine dritte Normalform, die einen Schlüssel auf den Angestelltenamen aufweist, kann z. B. nicht den Projektnamen, die Arbeitsgruppennummer und den Gruppenleiter enthalten, solange die Arbeitsgruppennummer und der Gruppenleiter nur dem Projekt zugewiesen sind, an dem der Angestellte arbeitet. Weitere Verfeinerungen für Normalformen sind die Boyce-Codd-Normalform (BCNF), die vierte Normalform (4NF) und Projektions-Kombinationsnormalform (englisch »projection-join«, Abkürzung: PJ/NF). Letztere wird auch als »fünfte Normalform« (5NF) bezeichnet. Diese Ebenen sind jedoch weniger gebräuchlich wie die ersten, zweiten und dritten Normalformen.

In der Programmierung versteht man unter »Normalform« die - manchmal als Backus-Normalform (Backus-Naur-Form) bezeichnete - Metasprache, die für die Beschreibung der Syntax anderer Sprachen verwendet wird - speziell ALGOL 60, für die sie geschaffen wurde.

## Weitere Quellen:

| <b>Internet-Seiten</b>                                                                | <b>Anmerkungen:</b>                                                           |
|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| <a href="#">Microsoft Press Computer-Fachlexikon</a>                                  | mit Fachwörterbuch (deutsch-englisch/englisch-deutsch)                        |
| <a href="#">Computer-Lexikon</a>                                                      |                                                                               |
| <a href="#">SQL in 21 Tagen</a>                                                       | eBook (Markt & Technik)                                                       |
| <a href="#">MySQL Datenbankhandbuch</a>                                               | eBook                                                                         |
| <a href="#">Datenbanken und SQL</a>                                                   | eBook                                                                         |
| <a href="#">SQL eBooks</a>                                                            | eBooks (English)                                                              |
| <a href="#">IT-Kompaktkurs</a>                                                        | Skripte, Folien zum runterladen                                               |
| <a href="#">SQL Tutorial</a>                                                          | (Englisch)                                                                    |
| <a href="#">Interaktiver SQL-Kurs</a>                                                 | (Englisch)                                                                    |
| <a href="#">Datenbanken/SQL-Tutorials</a>                                             |                                                                               |
| <a href="#">c't Artikel: Datenbanken: Wie man Informationen am Besten organisiert</a> |                                                                               |
| <a href="#">Datenmodellierung I</a>                                                   | Wirtschaftsinformatik Unterlagen                                              |
| <a href="#">Datenmodellierung II</a>                                                  | Wirtschaftsinformatik Unterlagen                                              |
| <a href="#">kostenlose Schulung zu PHP mit MySQL</a>                                  | alphatec<br>Standort: München                                                 |
| <a href="#">ACCESS-FAQ</a>                                                            | Fragen und Antworten zu MS ACCESS                                             |
| <a href="#">Entwickler-Forum</a>                                                      |                                                                               |
| <a href="#">ACCESS Portal</a>                                                         |                                                                               |
| <a href="#">EditPlus</a>                                                              | Editor mit Syntax-Highlighting für alle Sprachen (SQL, HTML, C++, JAVA, etc.) |
| <a href="#">zusätzliche PlugIns für andere Sprachen</a>                               |                                                                               |
| <a href="#">Access-Grundlagen</a>                                                     | Es werden auch Begriffe, wie DDL erklärt.                                     |
| <a href="#">Microsoft Office XP Developer</a>                                         | Access-Datenbanken als eigene Anwendungen weitergeben                         |
|                                                                                       |                                                                               |
| <b>Kostenlose Datenbanken:</b>                                                        |                                                                               |
| <a href="#">MySQL</a>                                                                 |                                                                               |
| <a href="#">PostgreSQL</a>                                                            |                                                                               |
| <a href="#">FireBird</a>                                                              | basierend auf Borland InterBase                                               |
| <a href="#">Oracle 9i Personal</a>                                                    | Einzelplatz-/Notebookversion                                                  |

## Buchempfehlungen:

| Internet-Seiten / Bücher:             | Anmerkungen:                                                                                                                                                                    |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Das große Buch SQL</a>    |                                                                                                                                                                                 |
| <a href="#">Das grosse MySQL-Buch</a> |                                                                                                                                                                                 |
| <a href="#">SQL in a Nutshell</a>     | Die englische Originalausgabe soll besser sein!<br>Dafür sind in der Deutschen Fassung zusätzliche Aktualisierungen enthalten → kommt dafür aber immer um einiges später heraus |
|                                       |                                                                                                                                                                                 |