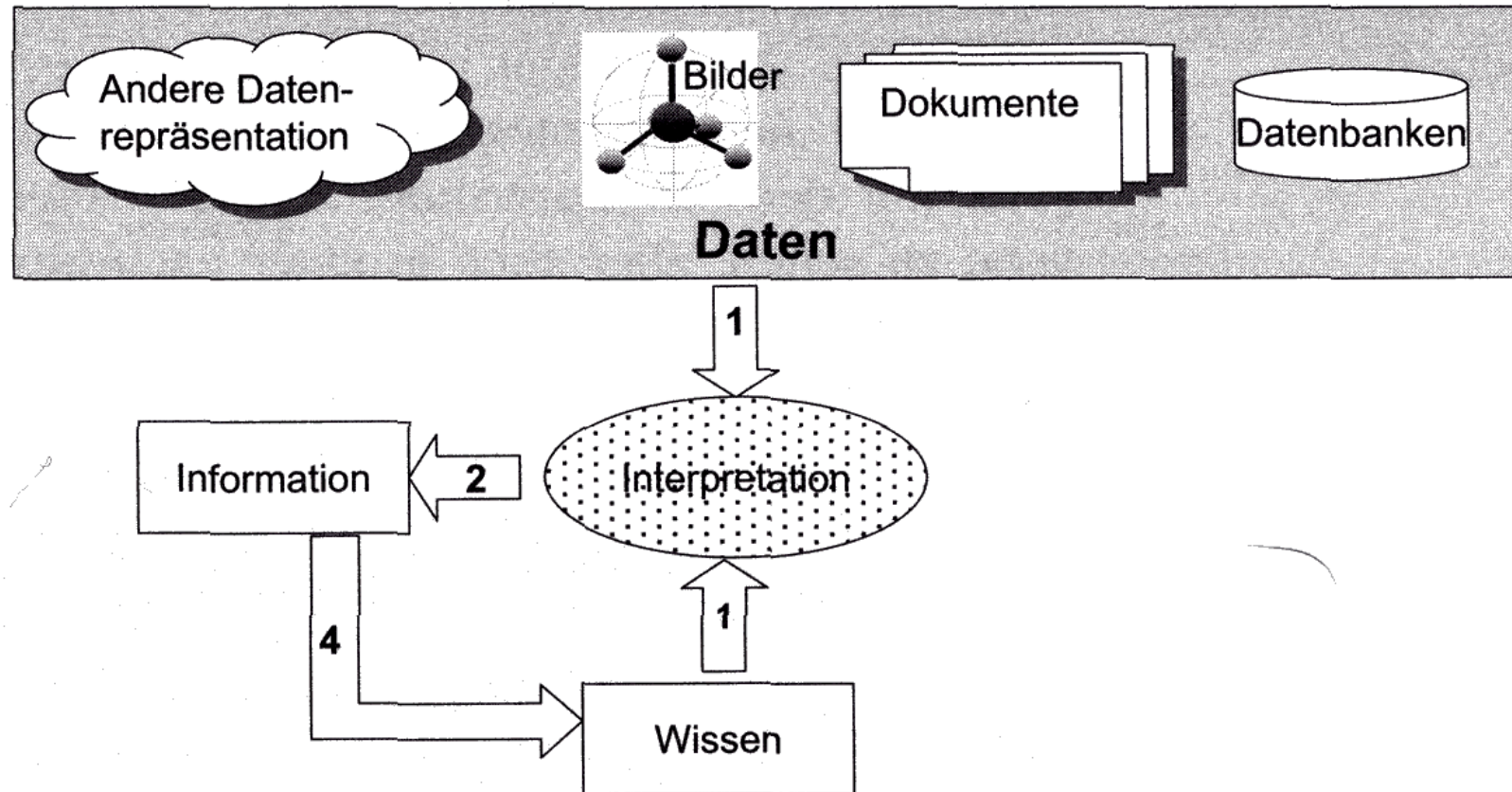


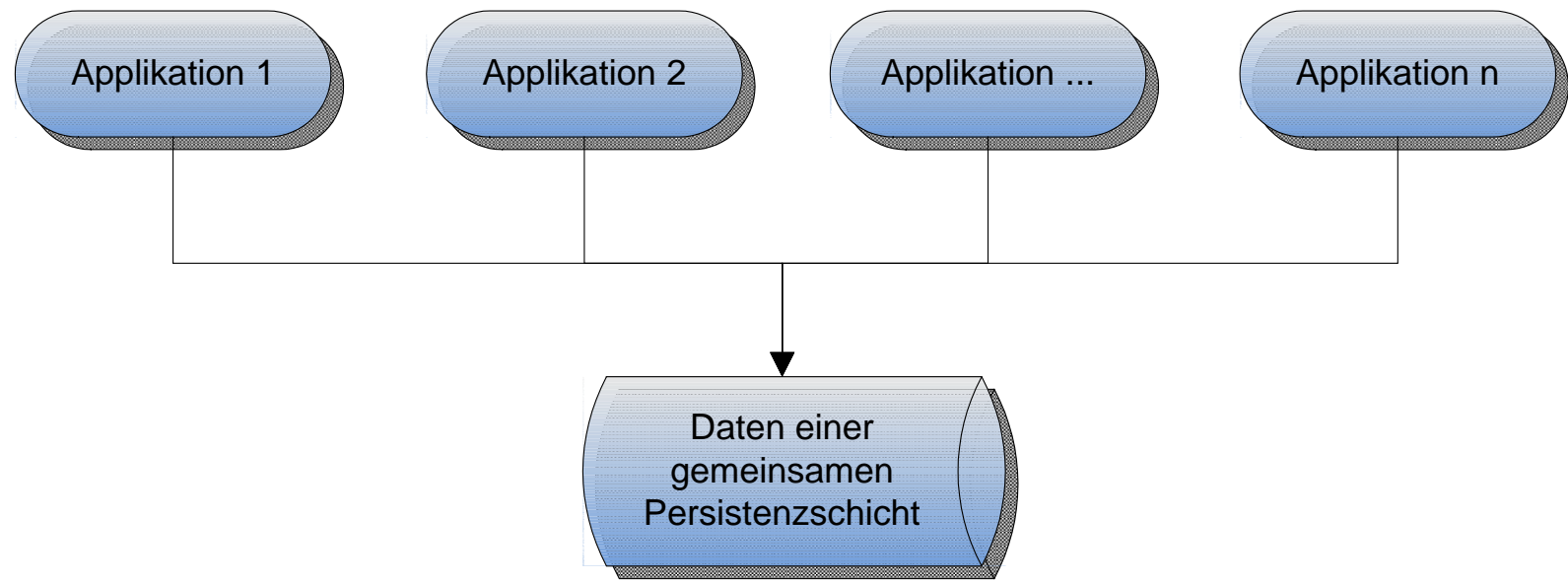
The diagram features a horizontal light blue bar with a gradient, wider on the left and tapering to the right. The word "Datenbanken" is centered on this bar. To the left, a blue circle is partially visible, with a thick black bracket spanning its width. To the right, a blue closing bracket is positioned. The entire composition is set against a light gray background.

Datenbanken

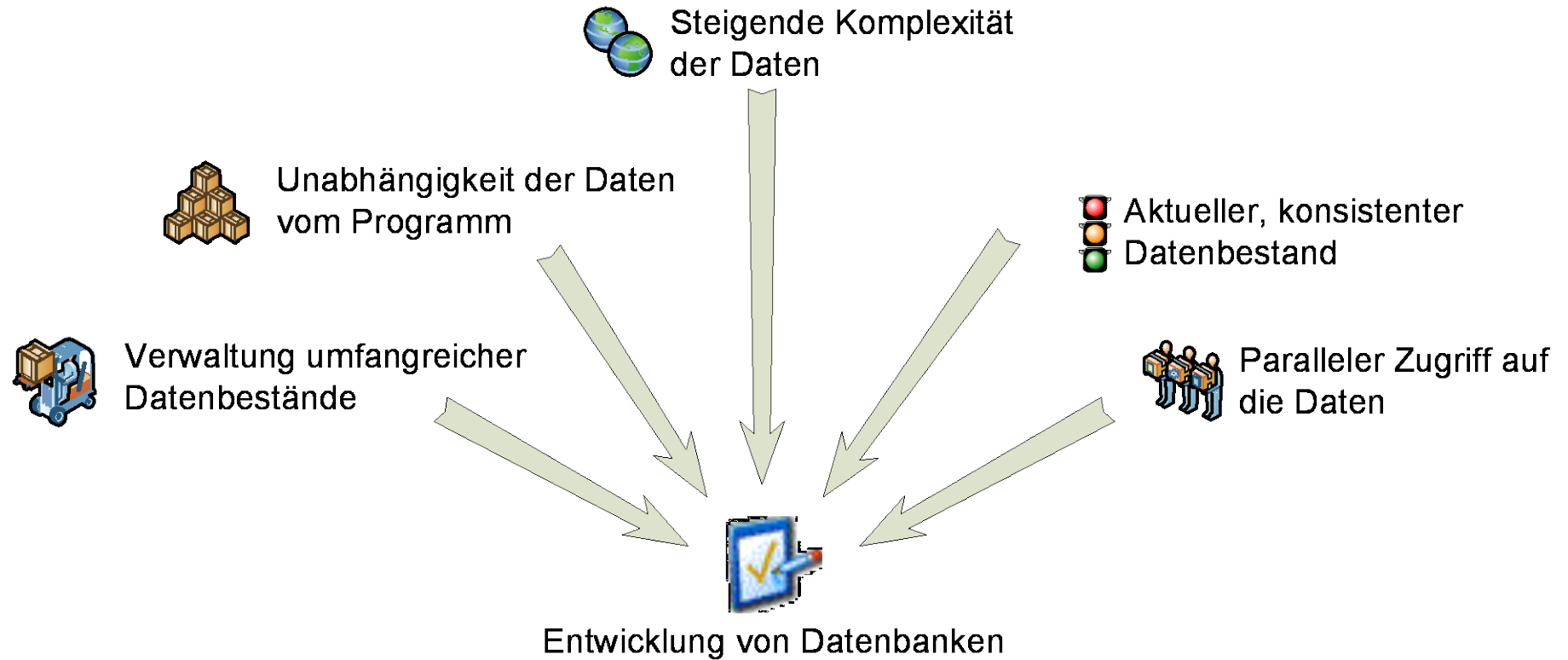


# Anforderungen an verteilte Software

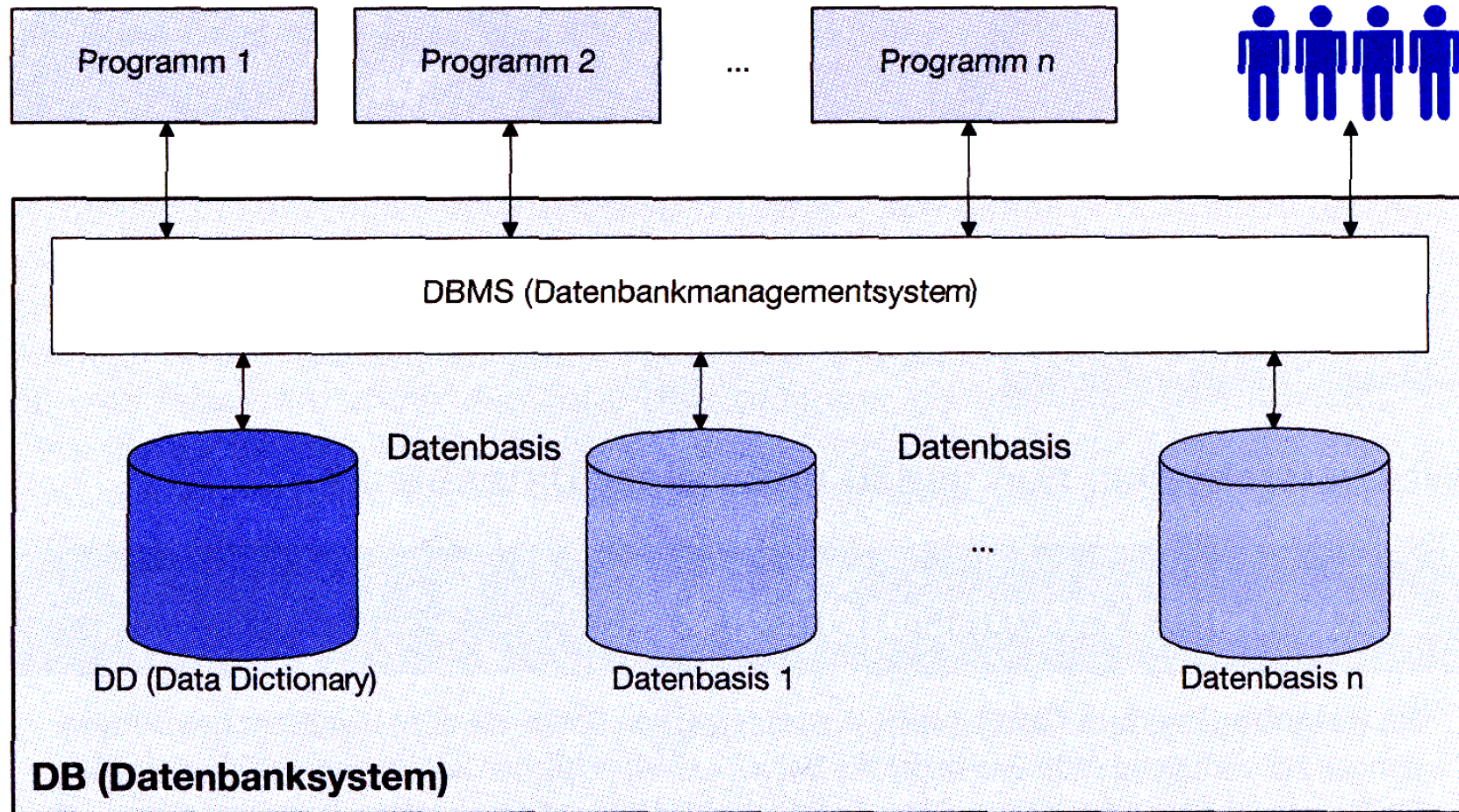
- Welche Voraussetzungen müssen gegeben sein, damit verschiedene Applikationen auf eine gemeinsame Persistenzschicht zugreifen können?
- Welche Anforderungen lassen sich dadurch für die Persistenzschicht festlegen?
- Welche Applikationen entsprechen dem dargestellten Schema?



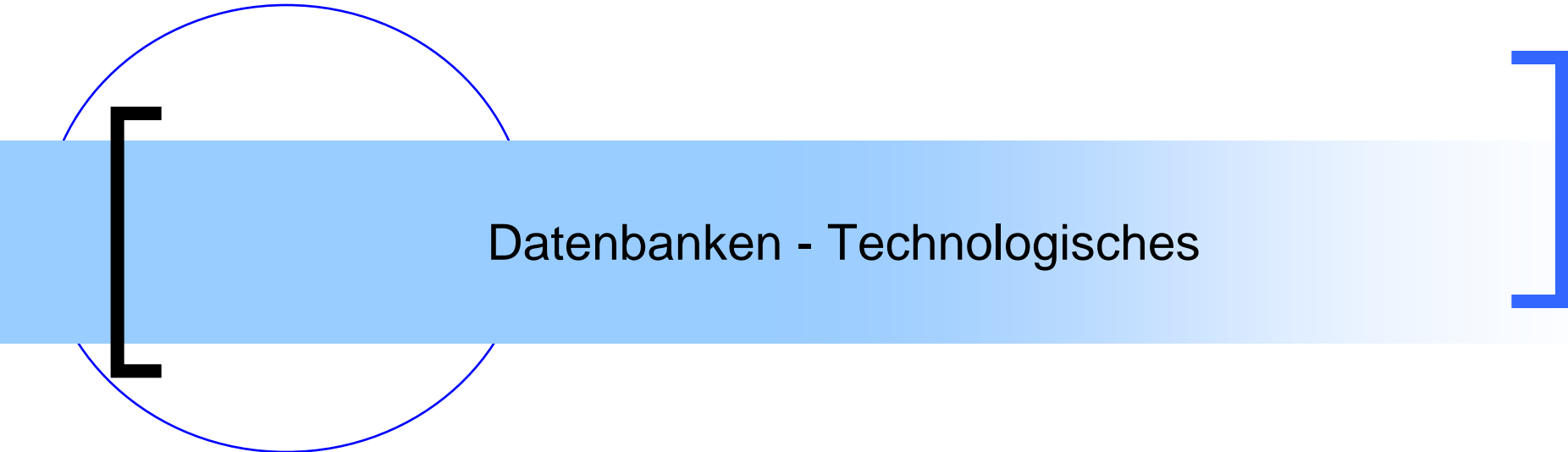
# Gründe für die Entstehung von Datenbanken



# Bestandteile einer Datenbank



Quelle: Faeskorn et.al.



# Datenbanken - Technologisches

- **Datenbasis**
  - Dies sind die eigentlichen Daten einer Datenbank, in der Regel repräsentiert durch verschiedene Dateien für die Anwendungs- und Metadaten
- **Data Dictionary**
  - Beschreibung der Datenbasis anhand von Metadaten (z.B. die Beschreibung der Spalten einer Tabelle hinsichtlich der Typen und Restriktionen)
- **Datenbankmanagementsystem (DBMS, data base management system)**
  - Das DBMS ist eine Applikation, die den Zugriff auf die Datenbasis regelt. Hier werden Dienste bereitgestellt, die es erlauben, die Datenbasis zu modifizieren, abzufragen oder auch zu optimieren. Weitere Funktionen erlauben die Sicherung der Daten, die Steuerung des Zugriffs oder die Interaktion mit anderen DBMS
- **Datenmodell**
  - Im Datenmodell wird festgelegt, in welcher Form ein Ausschnitt der Realität modelliert wird, dies betrifft in erster Form die Datentypen und deren Relationen
- **Datenbankschema**
  - Ausprägung eines konkreten Datenmodells



- ERP-Systeme (SAP, PSI, Oracle, Baan, Navision (Dynamics NAV), Abas, KHK, ...)
  - Lagerhaltung
  - Personalwesen
  - Stücklisten und Fertigungsaufträge
- Multimedia – Software
  - Media-Player
  - Foto-Alben
- Geoinformations (GIS) - Systeme
  - Kataster- und Liegenschaftsverwaltung
  - Kabel und Rohrleitungspläne
  - Routenplaner
- Internet – Anwendungen
  - Homepages, z.B. typo3
  - eCommerce – Shops, eBanking, Versicherungsapplikationen
  - Buchungssysteme für Reisen, Konzerte, ...
  - Gästebücher, Weblogs, Newsgroups
  - Suchmaschinen
  - Wikis

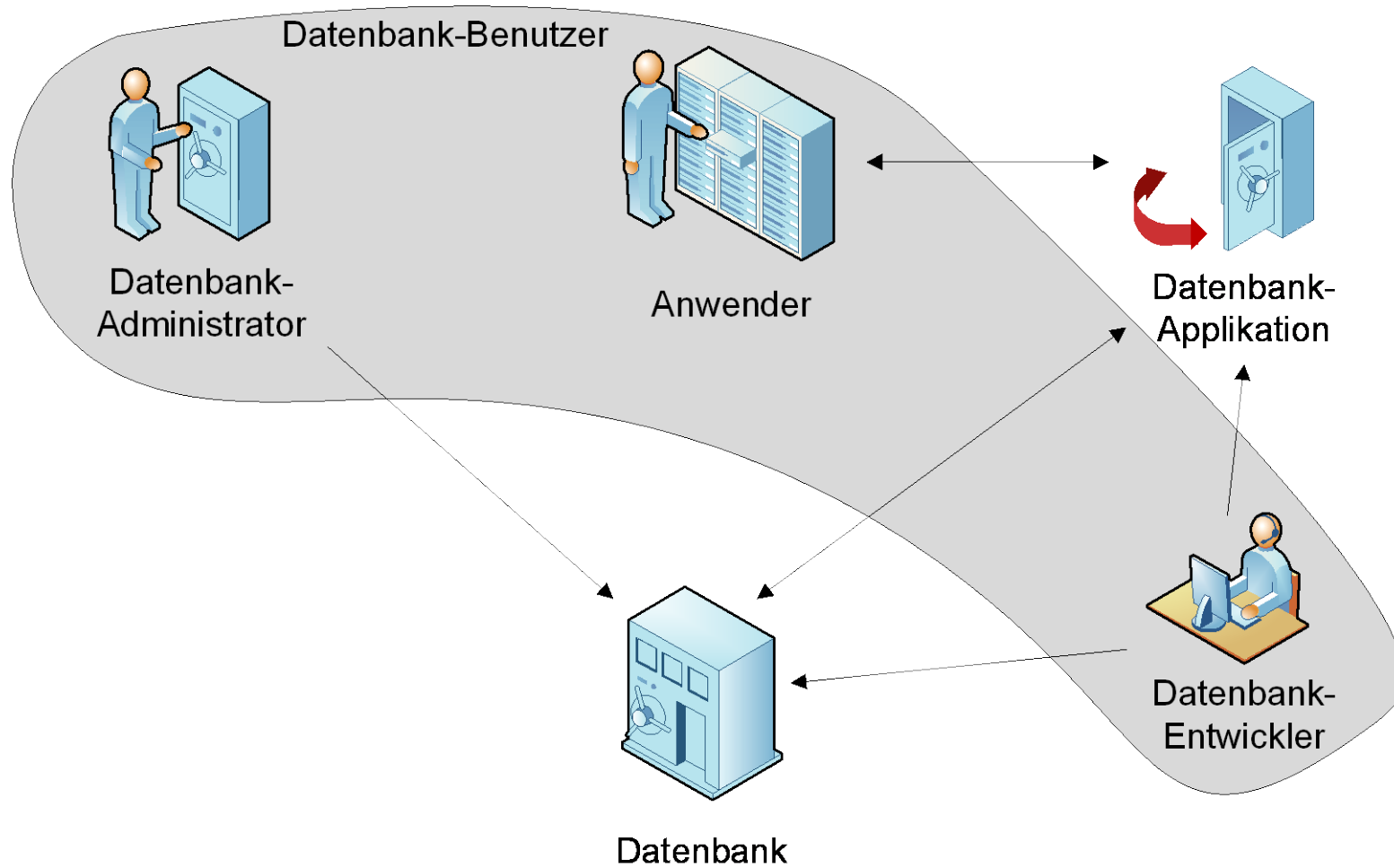


- Die Abbildung einer Information erfolgt durch nur einen Wert, dies trifft für die Metadaten und die Anwendungsdaten zu, keine Redundanz
- Jedes Feld der Datenbank ist durch eine Kombination von Relation, Primärschlüssel und Spaltenname ansprechbar
- Fehlende Werte in einer relationalen Datenbank werden durch Nullwerte abgebildet, der Leerstring ist kein Nullwert
- Das Data Dictionary unterscheidet sich in der Abbildung nicht von den Anwendungsdaten
- Ein RDBS unterstützt eine Sprache (in der Regel eine 4GL) zur Modifikation und Administrierung der Datenbank
- Ein RDBS erlaubt die Definition von Sichten (Views), Sichten sind das Ergebnis einer Suche
- Das Einfügen, Ändern und Löschen von Daten ist möglich
- Wenn sich die interne Speicherstruktur oder der Datenzugriff innerhalb der Datenbank ändert, bleibt das ohne inhaltlichen Einfluß auf die Anwendungsprogramme, ebenso bleiben diese Programme unverändert, wenn sich Datenstrukturen ändern, die nicht von Anwendungsprogrammen verwendet werden – Anwendungsprogramme, die auf die gleiche Datenbank zugreifen, agieren unabhängig voneinander

Quelle: nach Faeskorn et.al.

- Sofern im DBMS Integritätsbedingungen (z.B. mit Hilfe der Datenbanksprache) definiert worden sind, sollten diese auch dort gehandhabt werden und nicht noch einmal im Anwendungsprogramm abgebildet werden
- Wird die Datenbank auf mehrere Rechner verteilt, so bleibt dennoch das Anwendungsprogramm davon unbehelligt, da das DBMS nach außen hin unverändert gleiche Datenstrukturen anbietet
- Unterstützt das DBMS neben der Datenbanksprache auch eine 3rd Generation Language (3GL) wie C++ oder Java, so dürfen damit nicht die oben definierten Eigenschaften der Datenbanken unterwandert werden können – die Datenbank kann als nach außen gekapseltes, konsistentes System betrachtet werden
- Unterstützung eines Multi-User-Konzeptes
- Unterstützung von Transaktionen, Transaktionen sind zusammengefaßte Datenbankoperationen, bei denen die Gesamtheit der Transaktionen am Ende rückgängig gemacht werden können (Rollback) oder endgültig werden können (Commit) – nur durch Transaktionen sind komplexere Abläufe konsistent zu gestalten (Bankbuchung als einfaches Beispiel)

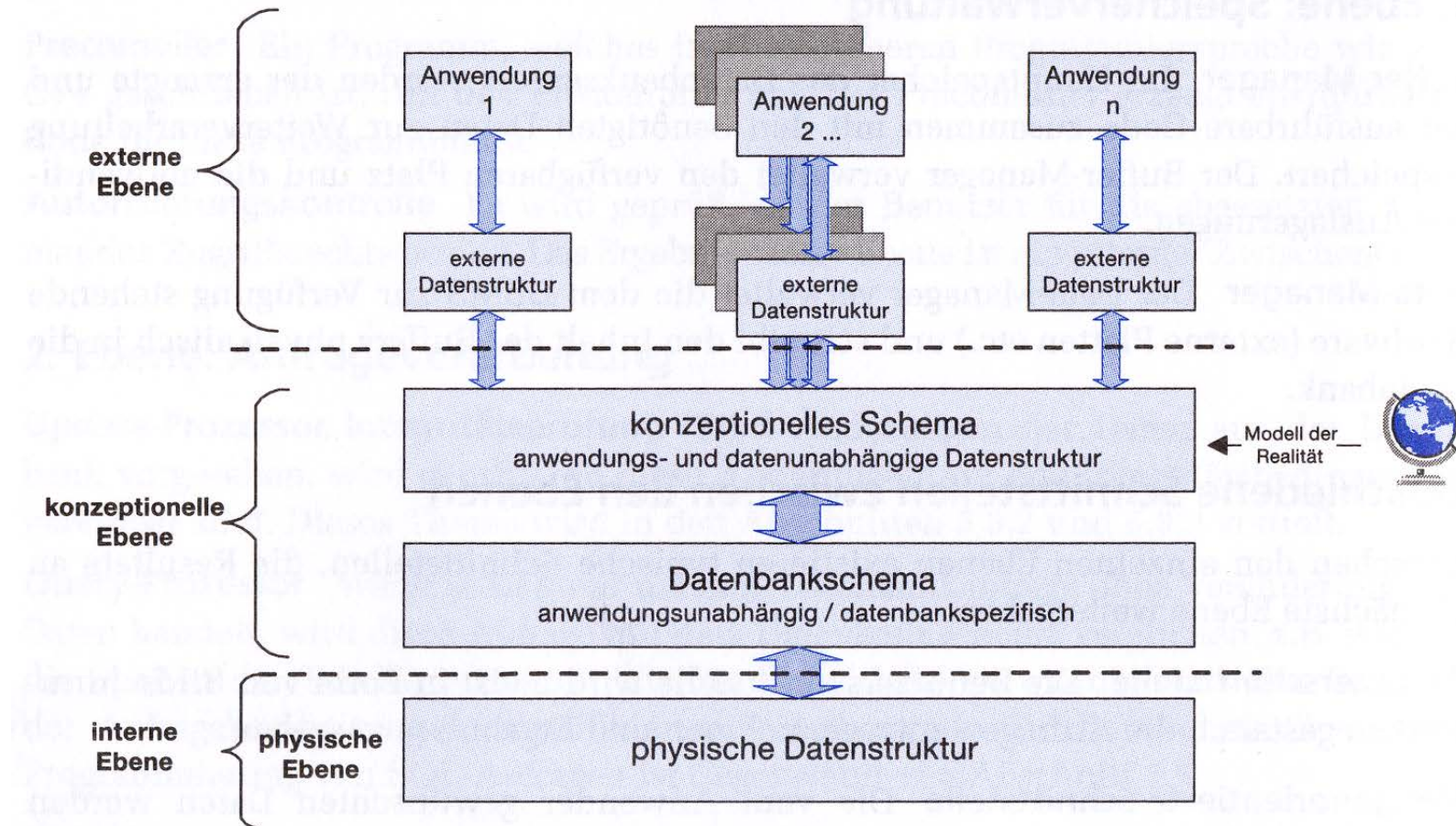
# Nutzer von Datenbanken



- Planung der logischen und physischen Struktur der Datenbank
  - Tabellen, Views und Indizes anlegen
  - Verteilung der Datenbank über mehrere Standorte
  - Replikation verteilter Datenbestände
- Softwareinstallation und Wartung
  - Installation der Datenbank
  - Einspielen von Updates und Patches
  - Datenübernahme aus Altversionen
  - Organisation des Plattenbedarfs
  - Reorganisation von Datenbeständen
  - Sicherstellung einer hohen Performance
- Sicherheit
  - Benutzerverwaltung und Zugriffsrechte
  - Schutz vor Fremdzugriff
- Backup und Recovery
  - Erstellen von Sicherungen der Datenbank
  - Einspielen von Backups

- Systemanalyse
  - Erstellung eines Datenbankmodells zur Abbildung realer Anwendungsfälle
  - Beschreibung des Datenbankmodells mit Hilfe eines ER- oder eines UML-Modells
  - Erarbeitung von Datenmodellen für Applikationen oder Datenmengen
- Erstellung individueller Abfragen
  - Nutzen einer 4GL-Sprache wie SQL zur Extraktion von Informationen aus Datenbanken, die durch vorhandene Applikationen nicht zur Verfügung stehen
  - Verknüpfen der Daten aus verschiedenen Datenbeständen zur Informationsgewinnung
- Anwendungsentwicklung
  - Erstellen von Datenbankanwendungen für verschiedene Aufgabengebiete
  - Einsatz einer 3GL Sprache für Aufgaben wie GUI und Teile der Logikschicht
  - Zugriff auf die Datenbank über die 4GL-Sprache zum Erzeugen, Ändern und Löschen von Daten bzw. zur Erstellung von Datensichten bzw. Suche
  - Nutzen der Datenbankfunktionen zur Realisierung von Replikationsmechanismen, Datenextraktion bzw. Datenex- und import

# ANSI – 3-Ebenen Modell



Quelle: Faeskorn et.al.

## ■ Administration

- Front-End vom Datenbankhersteller
- durch Log-In geschützt
- grafische Schnittstelle zur Administration mit den angebotenen Diensten
  - Benutzerrechte
  - Tabellenstrukturen und –Inhalte
  - Relationen
  - Backup und Recovery, ...

## ■ Programmierung

- neutrale Schnittstelle, z.B. ODBC-Schnittstelle (Open Data Base Connectivity), in der Regel vom Hersteller angeboten
- Direkte Programmierschnittstellen in Entwicklungsumgebungen wie z.B. in Visual Studio oleDB (oder alte DAO bzw. ADO) oder in php über entsprechende Packages für verschiedene Datenbanken

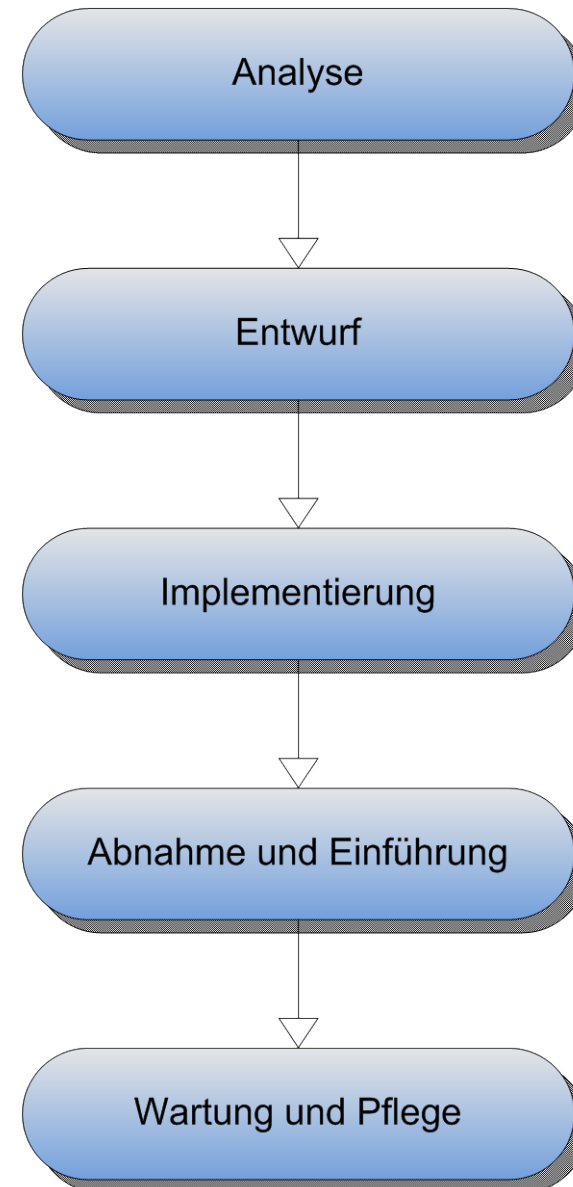
## ■ Auswahlkriterien

- Daten: wie viel, wie schnell, wo, wie viele User, Altdatenübernahme, ...
- Kosten: Lizenzen, Hardware, Wartungsverträge, Administrationsaufwand, ...
- Skalierbarkeit: Hardware, zukünftige Datenmengen, User, Transaktionen, Netzwerklast, ...



# Vorgehensmodell bei der datenbankbasierten Softwareentwicklung

- Ziel der Analysephase
  - Intern beim Auftraggeber: Lastenheft
  - Extern beim Auftragnehmer: Pflichtenheft
- Entwurfsphase (nach dem 3-Ebenen Modell)
  - Anwendungsfälle ausmodellieren
  - Erarbeitung eines konzeptionellen Schemas in Form eines UML bzw. ER-Diagramms
  - Erstellung des Datenbankschemas
  - Festlegung der physischen Datenstruktur
- Implementierung
  - Umsetzung des Entwurfes auf einer konkreten Datenbank
  - Implementierung der Anwendung
- Abnahme und Einführung
  - Schulungsmaßnahmen
  - Abnahme anhand Lastenheft/Pflichtenheft
- Wartung und Pflege
  - Fehlerbeseitigung
  - Leistungssteigerung
  - Weiterentwicklung





# SQL – ein Überblick

- Ein View ist eine Sicht auf eine (oder mehrere) Tabellen
- Durch eine Sicht werden
  - Daten gefiltert bzw. ausgeblendet
  - Daten in neuartiger Weise zusammengestellt
- Sichten werden in der Datenbank abgelegt, sie beinhalten im wesentlichen folgende Elemente
  - Auswahl der anzuzeigenden Spalten und deren Gruppierung
  - Auswahl der Tabellen oder Sichten, die in der Abfrage verwendet werden sollen
  - Verknüpfungseigenschaften zwischen den Tabellen
  - Einschränkung auf Datensätze, die bestimmten Kriterien genügen
  - Sortierung der Ergebnisse
  - `SELECT <spalten> FROM <tabelle, sicht> [WHERE <bedingung>] [ORDER BY <spalten>] [DESC|ASC] [GROUP BY <spalten>] [HAVING <bedingung>]`
- Beispiele
  - `SELECT * FROM Kunden`
  - `SELECT * FROM Kunden WHERE Umsatz>1000`
  - `SELECT Umsatz FROM Kunden WHERE ID=99`
  - `SELECT * FROM Kunden WHERE Name LIKE 'P*' ORDER BY Umsatz DESC`

- Vergleichsoperatoren
  - =, >, >=, <, <=, <> (Access) oder !=
- Logische Operatoren
  - AND, OR, NOT, XOR (nur eine von beiden wahr)
- SQL-Operatoren
  - [NOT] BETWEEN <untergrenze> AND <obergrenze>
  - [NOT] IN (<Kommagetrennte Werteliste>)
    - Trifft ein Wert der Liste zu
  - [NOT] LIKE
    - nur bei Texten, hier können auch Suchmuster verwendet werden (z.B. P\*)
    - \* (bei Access), % bei anderen Datenbanken ist Suchmuster für beliebige Zeichen (keines, eins oder mehrere)
    - ? (bei Access), \_ bei anderen Datenbanken ist Suchmuster für genau ein beliebiges Zeichen
    - # (bei Access) ist Suchmuster für eine Zahl, so kann mit ##.##.#### ein Suchmuster für ein Datum gebildet werden
  - IS [NOT] NULL
    - Ist der Feldinhalt leer, also kein Wert eingetragen, es gilt zu beachten, daß NULL != 0, NULL != ""

- LOWER und UPPER (<string>) konvertieren den String in Groß- bzw. Kleinbuchstaben
- LENGTH (<string>) gibt die Länge eines Strings zurück
- SUBSTR(<string>, <m> [, <n>]) gibt beginnend ab der Stelle m einen String der Länge n bzw. den Reststring zurück
- CONCAT (<string1>, <string2> ,...) fügt Strings zusammen
- ASCII (<zeichen>) konvertiert Zeichen in ASCII-Code
- CHR (<zahl>) konvertiert Zahl in ASCII-Zeichen
- ROUND (<zahl>, <n>) rundet Zahl auf n Nachkommastellen
- MOD (<zahl1>, <zahl2>) ergibt den Rest der Division von zahl1/zahl2
- TRUNC (<zahl>, <n>) schneidet die Zahl nach n Nachkommastellen ab
- Mathematische Operationen wie ABS, SIN, COS, TAN, LOG, LN, POWER, SQRT, ... werden unterstützt
- Datumsfunktionen (häufig DB-abhängig), um Zeitspannen, Tage usw. zu ermitteln
- sonstige Datenbankspezifische Funktionen

- COUNT ermittelt die Anzahl der Werte in einer Spalte
  - SUM ermittelt die Summe der Werte in einer Spalte
  - AVG gibt den Mittelwert einer Spalte zurück
  - MAX den größten Wert in einer Spalte
  - MIN den kleinsten Wert
- 
- CURRENT\_DATE ermittelt das aktuelle Systemdatum
  - CURRENT\_TIME die aktuelle Systemzeit
  - CURRENT\_USER den momentan eingeloggten User
  - weitere Systemvariablen je nach Datenbankfunktionalität

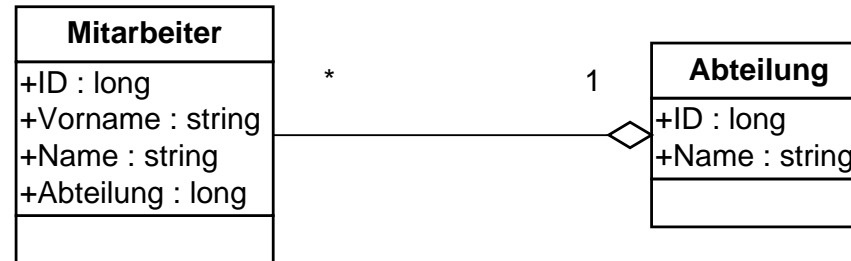
- Excel-Tabelle und Datenbank-Tabelle - wo liegen Gemeinsamkeiten, welches sind die Unterschiede?
  - Was kann eine Excel-Tabelle abbilden und welche Funktionalität wird angeboten
  - Welche Funktionalität besitzt die Tabelle einer Datenbank
- Wo gibt Gemeinsamkeiten zwischen einer Datenbank-Tabelle und einem Feld in C#?
  - Wie wird eine Datenbank-Tabelle in einfachster Weise abgebildet
- Wozu dient ein Index für eine Tabelle – wie funktioniert ein binärer Baum?
  - Welche Strategien zur schnelle Suche in Tabellen können verwendet werden



- Für das Erzeugen von Datensätzen stellt SQL das Sprachelement INSERT zur Verfügung
- Ein INSERT-Befehl ist wie folgt aufgebaut
  - `INSERT INTO <tabelle> [(<spalten>)] {VALUES (<werte>)|SELECT-Abfrage}`
  - die Werte (Values) müssen mit den Spalten korrespondieren, d.h. die Reihenfolge der Spalten muß der Reihenfolge der Werte entsprechen, bzw. bei Entfall von Spaltenangaben müssen die Angaben für alle Spaltenwerte in der korrekten Reihenfolge erfolgen
  - Die Werte können alternativ auch aus einer SELECT-Anfrage stammen, dann ohne VALUES-Angabe
- Beispiele für Inserts:
  - `INSERT INTO Mitarbeiter ( Vorname, Name ) VALUES ("Paul", "Baumann")`  
Hier müssen die VALUES vom Typ her den Spalten entsprechen und auch in der gleichen Reihenfolge aufgezählt werden, die Anzahl der zu füllenden Spalten ist beliebig zu wählen
  - `INSERT INTO Abteilung (Name) (SELECT namen FROM othertable WHERE ...)`  
Hier werden die VALUES für das INSERT aus einer Abfrage gewonnen, dabei muß die Anzahl der Spalten und deren Reihenfolge mit den Spalten des INTO-Teils übereinstimmen

- Das Löschen entspricht von der SQL-Syntax her der SELECT-Anweisung - die Datensätze, die beim SELECT angezeigt würden, werden durch DELETE gelöscht.
  - DELETE FROM <tabelle> [WHERE <Bedingung>]
- Beispiel
  - DELETE FROM Angestellte WHERE Abteilung=77;
- Das Ändern von Datensätzen erfolgt durch den UPDATE-Befehl
  - UPDATE <tabelle> SET <spalte = Wert> [, <spalte = Wert>] WHERE <bedingung>
  - Wert kann Null, eine explizite Angabe, ein SQL-Ausdruck oder eine SELECT-Abfrage mit einem Ergebnis sein
- Beispiele
  - UPDATE Kunden SET LetzterKontakt = CURRENT\_DATE WHERE ID=77;
  - UPDATE Preise SET PreisInEuro = PreisInDM / 1.95583;
  - UPDATE Kunden SET Gesamtumsatz = (SELECT SUM(Rechnungssumme) FROM Rechnungen WHERE (Rechnungen.KundenID = Kunden.ID));

- In einer relationalen Datenbank können zwischen zwei Tabellen Relationen aufgebaut werden, z.B. zwischen der Tabelle Mitarbeiter und der Tabelle Abteilung



- Jeder Mitarbeiter hat ein Feld, um die ID einer Abteilung zu hinterlegen. Damit kann die Zugehörigkeit zu einer Abteilung ausgedrückt werden, ohne alle Abteilungsdaten noch einmal redundant am Mitarbeiter zu hinterlegen, ändert sich der Abteilungsname, erfolgt diese Änderung an nur einer Stelle
- Abfragetypen:
  - `SELECT * FROM Mitarbeiter, Abteilung, dies entspricht`
  - `SELECT Mitarbeiter CROSS JOIN Abteilung)`
  - `SELECT * FROM Mitarbeiter, Abteilung WHERE Mitarbeiter.Abteilung = Abteilung.ID, dies entspricht`
  - `SELECT * FROM Mitarbeiter INNER JOIN ON Mitarbeiter.Abteilung = Abteilung.ID`
  - `SELECT * FROM Mitarbeiter LEFT OUTER JOIN Abteilung ON Mitarbeiter.Abteilung = Abteilung.ID` (alle Daten der Tabelle Mitarbeiter und nur die der Tabelle Abteilung, die damit in Beziehung stehen)
  - `SELECT * FROM Mitarbeiter RIGHT OUTER JOIN Abteilung ON Mitarbeiter.Abteilung = Abteilung.ID`
  - `SELECT * FROM Mitarbeiter FULL OUTER JOIN Abteilung ON Mitarbeiter.Abteilung = Abteilung.ID`
  - (WHERE-Kriterien können jeweils folgen)

- Analog zum Bearbeiten von Tabelleneinträgen steht SQL-Funktionalität auch für die Bearbeitung der Tabellen selbst zur Verfügung
  - Für die Bearbeitung der Tabellenstrukturen müssen entsprechende Rechte bestehen
  - CREATE TABLE ist die Anweisung zur Anlage einer Tabelle
  - CREATE INDEX erzeugt einen Spaltenindex
  - CREATE VIEW Abfragestatements können direkt als VIEW in der Tabelle angelegt werden
  - ALTER TABLE ändert vorhandene Tabellen
  - DROP TABLE löscht eine Tabelle
  - RENAME TABLE benennt eine Tabelle um