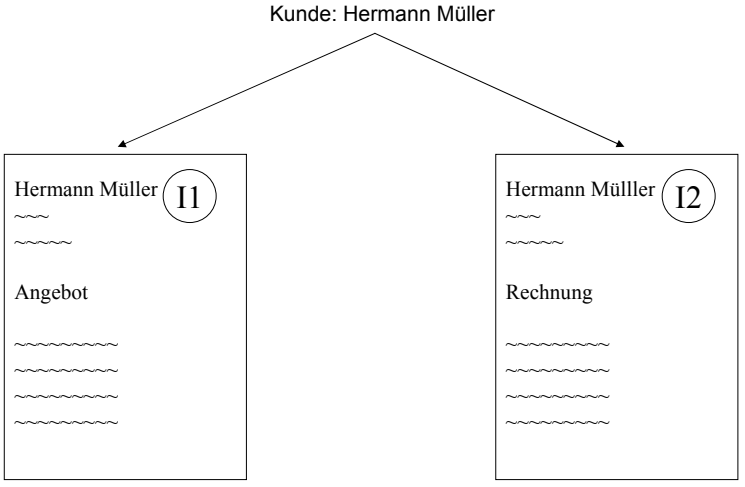
Datenorganisation: (Daten)Datei versus Datenbank

Grundsätzlich gilt:

Daten können in

- **(Daten)Dateien** oder
- **in Datenbanken** organisiert werden.

Datenorganisation in Dateien

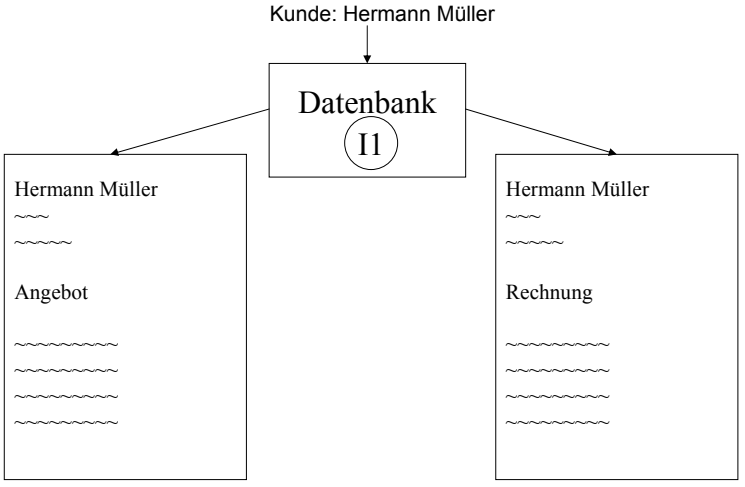


Datenorganisation in Dateien

- Vorteile von (Daten)dateien:
  - Schnell zu erstellen. (Keine Planung erforderlich).
  - Einfach zu erstellen. (Kein besonderes Wissen erforderlich, außer Programmbedienung. D.h., der Mitarbeiter hat die Bedienung in der Hand).
  - Jedes beliebige Programm verwendbar (freie Auswahl beim Erstellen).

- Nachteile von (Daten)Dateien:
  - Datendateien sind an die Software gebunden, mit denen sie erstellt wurden.
  - Speicherung von Daten in Datendateien geht mit redundanten Daten einher.
    Redundanz = Gleiche Informationen werden an unterschiedlichen stellen mehrfach gespeichert.
    Information 1 (I1) und Information 2 (I2) beinhalten beide die gleiche Anschrift. Diese wird jedoch zwei Mal in unterschiedlichen Dateien gespeichert, so dass in I2 ein Eingabefehler im Nachnamen die Folge ist.
  - Dadurch sehr aufwendige Datenpflege mit hoher Fehlerwahrscheinlichkeit.
  - Datendateien können schlecht ausgewertet werden.

Datenorganisation in Datenbanken



Datenorganisation in Datenbanken

- Vorteile von Datenbanken:
  - Redundante Daten können ausgeschlossen werden, da die Information 1 (I1) nur einmal in der Datenbank gespeichert wird.
  - Daten können von anderen Programmen benutzt werden (z.B. Ausgabe eines Serienbriefs in Word).
  - Daten können von mehreren Mitarbeitern gleichzeitig benutzt werden.
  - Eingabeoberfläche ist nicht von einer Anwendung (Textverarbeitung etc.) abhängig. Folge: Nur eine Eingabeoberfläche für alle Anwendungen.

- Nachteile von Datenbanken:
  - Datenausgabe ist **bei einmaliger Verwendung** sehr zeitaufwändig.
  - Kosten für Anschaffung, Wartung und Anpassungen an neue Aufgaben.
  - Gestaltung der Datenbank erfordert i.d.R. einen Programmierer.
  - Eingabefehler werden an alle Anwendungen weiter gegeben!

Allgemeine Grundlagen

Bestandteile einer Datenbank

Eine Datenbank besteht aus:

- einem Datenbankverwaltungssystem
  - Wird als **DBMS = Database Management System** bezeichnet.
  - Stellt alle strukturellen Bestandteile zur Datenorganisation bereit. Z.B. Benutzerrechte, Sprache zur Datenmanipulation, Speichervorgänge etc.

- miteinander verknüpften Dateien (Tabellen), in denen die Daten enthalten sind.
  - Wird als **DB = Datenbank** bezeichnet.
  - Je nach Hersteller, werden diese Dateien (Tabellen) innerhalb des DBMS, oder extern in real existierenden Einzeldateien gespeichert.

Als **Datenbanksystem (DBS)** bezeichnet man die Einheit von DBMS und DB.
DBS = DBMS + DB

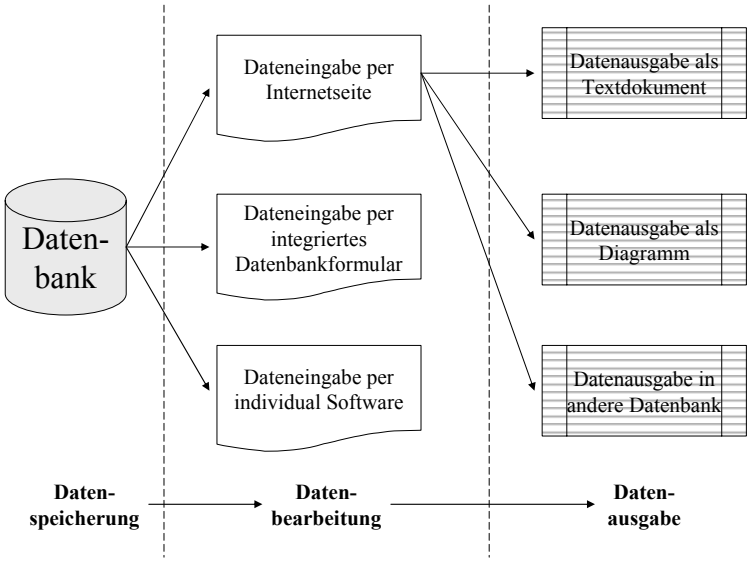
DBS: Client-Server-Prinzip

Aufteilung in:
1. Backend (eigentliche DBS)
2. Frontend (Benutzeroberfläche)
3. Anwendung zur Datenausgabe (z.B. Word, Excel etc.)

Backend:
- Dient der Datenspeicherung.

Frontend:
- Dient grundsätzlich der Dateneingabe, -änderung und dem Löschen von Daten.
- Einige Datenbanksysteme ermöglichen im Frontend auch die Datenausgabe in Form von druckbaren Berichten.
- **Speicherung von Befehlen** zur Datenmanipulation (siehe auch DML), nicht jedoch deren Ausführung!!!
- Bereitstellung individueller Funktionen (z.B. Weitergabe der Daten an die Textverarbeitung; Datensicherung in ein anderes Dateiformat etc.).

DBS: Client-Server-Prinzip (Beispiel)



SQL – Die Sprache des Datenbanksystems

Die (Programmier)Sprache SQL (Standard Query Language) wird benötigt, um Zustandsänderungen in einem DBS vorzunehmen.

SQL ist eine Zusammenstellung mehrerer Sprachen mit unterschiedlichen Aufgaben. Die wichtigsten sind:

- DDL (Data Definition Language)
  - Tabellen erstellen, ändern, löschen
  - Beziehungen erstellen, ändern, löschen
  - Index erstellen, ändern, löschen
- DML (Data Manipulation Language)
  - Daten hinzufügen, ändern, löschen
- DQL (Data Query Language)
  - Daten filtern (auswählen)

SQL – Codebeispiel

Nachstehend ein Beispielcode in SQL:

```
SELECT *
FROM patienten
WHERE nachname = “Schmidt”;
```

Erklärung:

SELECT *
Wähle alle Felder aus (Sternchen steht für alle Felder).

FROM patienten
Die in der Tabelle mit der Bezeichnung “patienten” stehen.

WHERE nachname = “Schmidt”;
Wähle jedoch nur diejenigen Datensätze aus, in deren Feld “nachname” der Inhalt “Schmidt” steht.

Entwurf einer Datenbank

Entwurfsphasen

1. Analyse der Anforderungen
   Analyse des benötigten Datenbestands, der benötigten Genauigkeit, allen Ein- und Ausgabefunktionen etc.

2. Entwurf des Datenbankkonzepts
   Beschreibung durch semantisches Datenmodell (z.B. ER-Modell)

3. Logischer Entwurf
   Beschreibung durch logisches Datenmodell (z.B. relationales Datenmodell)

4. Physischer Entwurf
   Umsetzung des logischen Datenmodells im verwendeten DBS (z.B. Accss)

5. Test und ggf. Erweiterung
   Betatest des DBS mit eventl. Nachbesserungen

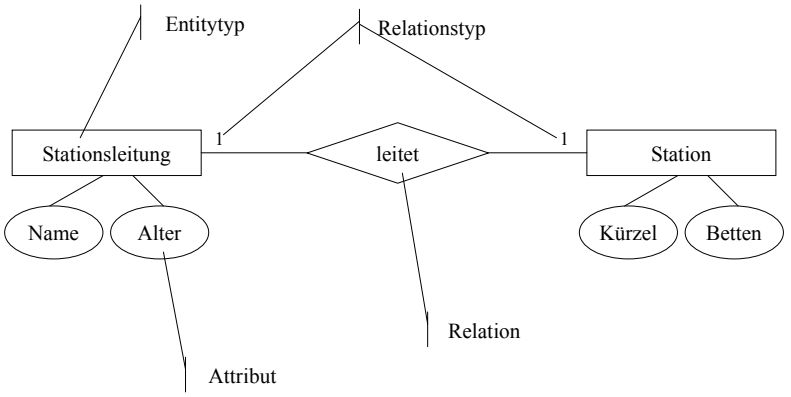
6. Wartung
   Wartungsarbeiten im laufenden Betrieb (Datensicherung, DB-Wartung)

zu 2) Semantische Datenmodelle

- 1976: **Entity-Relationship-Model**l (Chen) – auch **ER-Modell**
  - später Extended-Entity-Relationship-Modell (Fry, Teorey, Yang)
- 1978: SDM (McLeod)
- 1979: RM/T (Codd)
- 1979: DAPLEX (Shipman)
- 1980: TAXIS (Bernstein, Mylopoulos, Bernstein)
- 1984: IFO (Abiteboul, Hull)

- Ziel:
  Semantische Datenmodelle versuchen möglichst wirklichkeitsgetreu die Realität in einem Schema abzubilden. Sie nutzen dazu die natürlichen Begriffsstrukturen und bilden diese mit Hilfe von definierten Symbolen ab.
- Das meist genutzte semantische Datenmodell ist nach wie vor das ER-Modell.

Das ER-Modell



Das ER-Modell: Erläuterung

Das ER-Modell beschreibt Objekte, indem es unterteilt in:

- **Entitytyp**: Ein Entitytyp beschreibt eine Gruppe von gleichartigen Objekten (z.B. die Mitarbeiter, die Patienten)
- **Entity**: Meint ein einzelnes Objekt eines Entitytyps (z.B. den Mitarbeiter Herrn Müller)
- **Attribute**: Jedes Entity kann ein oder mehrere Eigenschaften besitzen (Bsp. Mitarbeiter: Nachname, Vorname, Geburtsdatum etc.)
- **Attributwert**: Meint ein einzelnes Attribut (z.B. den Nachnamen „Müller“, „Schmitt“ etc.)
- **Relation**: Jedes Entity kann mit anderen Entity(typen) in Beziehung stehen.

Relationstypen

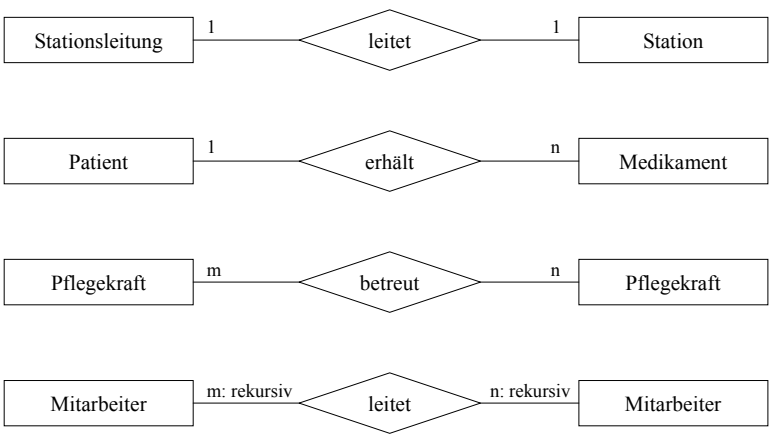
Allgemeines:

- Die **Relation** beschäftigt sich mit der Frage, wie zwei Entitytypen miteinander in Verbindung stehen (z.B. Die Stationsleitung **leitet** die Station)
- Der **Relationstyp** beschäftigt sich mit der Frage, nach welchen Regeln dies erfolgen darf (z.B. Die Station **darf nur von einer** Stationsleitung **geleitet werden**. Und jede Stationsleitung **darf nur eine** Station **leiten**).

Beziehungstypen:

- **1:1** (sprich: eins zu eins)
  Ein Objekt der Entität A steht einmal in Beziehung zu einem Objekt der Entität B.
- **1:n**
  Ein Objekt der Entität A kann zu ein oder mehreren Objekten der Entität B in Beziehung stehen.
- **m:n**
  Ein oder mehrere Objekte der Entiät A können zu ein oder mehreren Objekten der Entität B in Beziehung stehen.
- **rekursive Beziehung**
  Ein oder mehrere Objekte der **Entität A** können zu ein oder mehreren Objekten der **Entität A** in Beziehung stehen.

Das ER-Modell: Relationstypen



zu 3) Logische Datenmodelle

Zur Zeit ist das **relationale Datenmodell** das am häufigsten genutzte Datenmodell.

Weitere Modelle sind:

- Hierarchisches Datenmodell
- Netzwerk Datenmodell
- Objektorientiertes Datenmodell
- Deduktives Datenmodell

Ziel:

- Umsetzung der Ergebnisse des ER-Modells in ein Schema, das dem Aufbau der verwendeten Datenbank entspricht.
  - d.h. zuerst muss das später verwendete Datenbanksystem ausgewählt werden. Daraus ergibt sich dann das verwendete logische Datenmodell!!!

Das relationale Datenmodell

PersNr.	Name	Vorname
456	Müller	Hans
222	Schmitt	Frieda

1:n-Beziehung

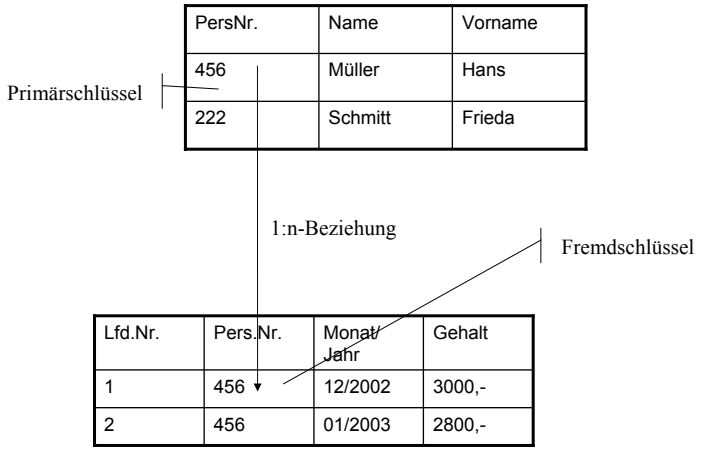
Lfd.Nr.	Pers.Nr.	Monat/ Jahr	Gehalt
1	456	12/2002	3000,-
2	456	01/2003	2800,-

Das relationale Datenmodell: Erläuterung

- Tupel:
  - Im relationalen Datenmodell werden die im ER-Modell als „Entity“ bezeichneten Elemente mit dem Begriff **Tupel** beschrieben.
  - Innerhalb der Datenbank stellt ein Tupel einen vollständigen **Datensatz** dar, d.h. ein Entity mit all seinen Attributwerten.

- Schlüssel:
  - Die Relationen des ER-Modells werden im relationalen Datenmodell präzisiert. Um ein Objekt mit einem anderen Objekt in Beziehung zu setzen, wird ein eindeutiger Wert benötigt. Diesen Wert bezeichnet man auch als **Schlüssel**.
  - Primärschlüssel:
    Ein Wert, der in der Ausgangstabelle nur einmal vorkommen darf.
  - Fremdschlüssel:
    Ein Wert, der in der verbundenen Tabelle je nach Relationstyp ein oder mehrfach vorkommen darf.

Das relationale Datenmodell: Schlüssel



Normalisierung

- Als Normalisierung bezeichnet man den Prozess Informationen so weit zu atomarisieren, so dass jede Einzelinformation ein eigenes Speicherfeld erhält.
- Belegen mehrere Werte ein Speicherfeld, so wird dieses Speicherfeld in eine neue Tabelle aufgeteilt, die mit der Ausgangstabelle verknüpft wird (sprich: es wird eine neue Relation erstellt).
- Ziel:
  - Vermeidung von Redundanzen!
- Man unterscheidet in der Praxis 3 Stufen der Normalisierung.

Praxisbeispiel

Praxisbeispiel: Grundlagen eines Datenbanksystems

Erstellen Sie eine Datenbank mit deren Hilfe Sie Ihren Patientenstamm und deren Medikation verwalten können.
Bitte brücksichtigen Sie, das jeder Patient mehrere Medikamente erhalten kann.

Minimimal müssen folgende Datenfelder vorhanden sein:
Nachname, Vorname, Straße, Postleitzahl, Ort, Diagnose, Medikament, Dosierung

Lösung

	ID	nachname	vorname	strasse	plz	ort	diagnose
▶ +	1	Müller	Hans	Hauptstraße 5	12345	Berlin	M. Parkinson
+	2	Schmidt	Rita	Klingweg 38	14455	Mühlhausen	KHK
+	3	Bittermann	Claudia	Am Bach 7	65478	Gersfeld	Adipositas, Diabetes Mellitus
*	(AutoWert)				0		

1:n Beziehung

	ID	IDPatient	Medikament	Dosierung
▶	1		Med. A	
	2	1	Med A 200mg	1 - 1 - 1
	3	1	Med B 500mg	2 - 1 - 1
	4	2	Med A 200mg	1 - 1 - 1
	5	2	Med C 100mg	0,5 - 0,5 - 1
	7	3	Med A 200mg	1 - 1 - 1
	8	3	Med G 1g	1 - 1 - 1
	9	3	Med F 300mg	1 - 1 - 1
*	(AutoWert)	0		

Anwendungsbereiche in der Pflegepraxis

- Planung:
  - Grundsätzliches Problem: Der Auftraggeber (die Pflegekraft) muss dem Auftragnehmer (der Programmierer) verständlich machen, welche Aufgaben mit welcher Genauigkeit von der Datenbank erfüllt werden sollen.
  - Hauptaugenmerk sollte dabei auf der **gewünschten Exaktheit** während der Planung liegen.
  - Folge bei Nichtbeachtung: Nachträgliche Korrekturen gehen in den meisten Fällen mit einem großen Zeit- und Kostenaufwand einher.
  - Dazu müssen folgende Punkte abgeklärt werden:
    - Welche Daten sollen im Einzelnen verwaltet werden?
    - Welche Filter- und Suchmöglichkeiten sollen möglich sein?
    - Welche Eingabemöglichkeiten sollen möglich sein?
    - Welche Ausgabe- und Exportmöglichkeiten sollen möglich sein?
    - Welche und wie viele Software-Lizenzen werden zur Umsetzung benötigt?

Anwendungsbereiche in der Pflegepraxis

- Nutzung:
  - Bedienelemente sollten bekannt sein, da diese in allen Datenbanken sehr ähnlich sind. Dazu gehören:
  - Elemente der Navigation (nächster/letzter Datensatz, vorheriger/erster Datensatz, neuen Datensatz anlegen)
  - Suchfunktionen (einfache Suche mit Suchfenster, Datenfilterung)
  - Datenausgabe
  - Datenexport in andere Anwendungen (z.B. Word, um dort einen Serienbrief zu erstellen)

- Entwicklung:
  - Einfache Datenbanken werden durch die Führungskraft selbst entworfen.
  - Gründe hierfür können sein:
    - Notwendiges KnowHow ist bereits vorhanden.
    - Fehlende finanzielle Ressourcen zur Beauftragung eines Software-Unternehmens mit der Programmierung.