DI (FH) Markus Falkensteiner

Datenbanken

CTME – 3. FSC – 2013/14

Themenübersicht

- Was ist eine Datenbank
- Anforderungen an eine Datenbank
- SQL

Datenbanken

- … dient dazu Daten (nicht Informationen!!) elektronisch zu Verwalten.
- Unterschied Daten / Informationen??

  Daten: z.B.: Zahl 42

  Information: Im Lager sind noch 42 Stück des Produktes xy

- Wesentlichen Aufgaben:

  Große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern

  Teilmengen des Datenbestandes soll bestimmten Benutzern/Anwendungen zugänglich gemacht werden.

Datenbank besteht im wesentlichen aus 2 Teilen:

- Datenbasis (Menge der zu verwaltenden Daten)
- Datenbankmanagementsystem (DBMS):
  - dient der Verwaltung der Daten
  - zur Strukturierung der Daten
  - Regelt Zugriffe (read/write) auf die Daten
  - Bietet eine Datenbanksprache zum Zugriff und Verwaltung der Daten an

• Datenbanken

– **Unter Datenbank versteht man ein System zur Beschreibung, Speicherung und Wiedergewinnung von umfangreichen Datenmengen**

– *Eine Datenbank ist eine integrierte Ansammlung von Daten, die allen Benutzern eines Anwendungsbereiches als gemeinsame Basis aktueller Information dient.*

– **Typische Anwendungen**:

Bank (Buchungen, Kontoverwaltung)

Bibliothek (Volltextsuche, Entleihe)

Redaktionssysteme im Internet (CMS) (Dokumente erstellen, Struktur einer Webseite)

eBusiness (Auftrag, Katalog, Lagerverwaltung)

ERP (Personal, Buchhaltung, Controlling)

Support (Ticketverwaltung)

Und und und

• Datenbanken/DBMS - Anforderungen

1. Persistenz

   Daten sollen dauerhaft gespeichert werden und zu einem späteren Zeitpunkt auch wieder abrufbar sein.

2. Anlegen von Datenschemata

   Daten haben je nach Kontext unterschiedliche Bedeutung. Ein Schema (z.B.: Tabelle) stellt den Zusammenhang zwischen Daten und Kontext her.

3. Einfügen, Ändern und Löschen von Daten

   Möglichkeit Daten in das Datenschema einzutragen, zu ändern oder auch wieder zu löschen.

   Eine Zeile im Datenschema wird Datensatz bezeichnet

4. Lesen von Daten

   Es muss möglich sein Daten aus der Datenbank wieder aufzufinden.

• Datenbanken/DBMS - Anforderungen

5. Integrität und redundanzfreie Datenhaltung

   Möglichkeit das ein Datum, welches an mehreren Stellen benutzt wird, nur an einer Stelle hinterlegt ist. Sicherstellung das Änderung des Datums an alle benutzen Stellen im System propagiert wird.

6. Koordination der parallelen Nutzung

   Sicherstellung das Integrität der Datenbank bei parallelen Zugriffen nicht verloren geht. Jeder Nutzer muss Eindruck haben, das ihm die Datenbank exklusive alleine gehört.

7. Rechteverwaltung

   Unterschiedliche Benutzer der Datenbank sollen unterschiedliche Berechtigungen besitzen

8. Datensicherung

   DBMS ermöglicht eine Datensicherung des aktuellen Datenbestandes herzustellen und diesen auch wieder in das System zurückzuspielen

9. Katalog

   Möglichkeit die Struktur des gesamten Systems bestimmten Benutzern zugänglich zu machen.

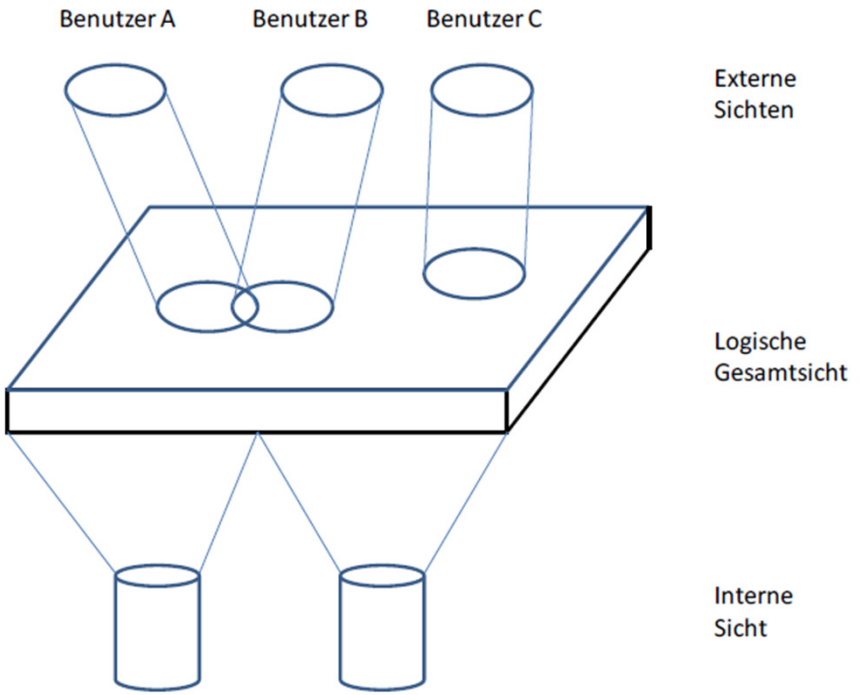
   (Datenschema, Nutzerrechte, …)

• Datenbanken – Ebenen eines DBMS

Je nach Anwender, ergeben sich unterschiedliche Sichten auf die DB/DBMS.

Ein Benutzer sieht z.B. die Daten völlig anders als der Systemprogrammierer, der die Organisation der Daten auf den Speichern festlegt.

1. Externe Ebene/Sicht
2. Logische Ebene/Sicht
3. Physische/Interne Ebene/Sicht



• Datenbanken – Ebenen eines DBMS

1. Externe Ebene/Sicht

   Jede Benutzergruppe sieht den Ausschnitt der Datenbank, der für sie von Bedeutung ist. Die Daten werden so dargestellt, wie es für die Benutzer wünschenswert oder natürlich ist.

2. Logische Ebene/Sicht

   In der Datenbank sind alle wichtigen Daten zusammengefasst. Um die Datenbank erstellen zu können, ist eine Gesamtschau der Daten notwendig. Alle Daten müssen zunächst auf logischer Ebene in Form von Informationseinheiten und deren Beziehungen untereinander beschrieben werden, unabhängig von EDV-Gesichtspunkten. Diese Beschreibung der Gesamtheit der Unternehmensdaten nennen wir logische Gesamtsicht.

3. Physische/Interne Ebene/Sicht

   Die Daten müssen auf den Speicher so organisiert werden, dass die Zugriffsanforderungen der verschiedenen Benutzer möglichst effizient erfüllt werden können.

• Datenbanken – Ebenen eines DBMS

Die Benutzer arbeiten mit der Datenbank ausschließlich über ihre externen Sichten. Sie sehen weder die logische Gesamtsicht noch die interne Organisation der Daten. Die notwendigen Umsetzungen von einer externen Sicht in die logische Gesamtsicht und von dort in die interne Sicht erledigt das DBMS. Es benötigt dazu Beschreibungen der jeweiligen Sichten und Regeln, die die Umsetzung von einer Sicht in die andere ermöglichen.

Jede Ebene der Daten modelliert die Daten auf einem anderen Abstraktionsniveau. Diese *Modelle* der Datenwelt werden mit Hilfe sogenannter *Datenbeschreibungssprachen* in einer für das DBMS verständlichen Form beschrieben; diese Beschreibung heißt dann ***Schema***. Es gibt also verschiedene externe Schemata, ein konzeptuelles Schema, das die logische Gesamtsicht beschreibt, und ein internes Schema.

Die einzelnen Schemata werden mittels Transformationen in das jeweils andere übergeführt. Die Transformation definiert die Abbildungen zwischen den einzelnen Schichten.

• ER-Modell – Entity Relationship Modell

Für die Modellierung einer Datenbank ist die logische Ebene ausschlaggebend. In ihr werden Modelle der „realen Welt“ abgebildet.

Das ER-Modell ist das wichtigste für die Abbildung der realen Welt. Es beschreibt Objekte (**Entity**) der Realwelt und ihre Beziehung (**Relationship**) zueinander.

Es wurde 1976 von Peter Chen vorgestellt. Von ihm stammt auch die Chen-Notation welche für die grafische Darstellung von Datenmodellen verwendet wird.

Begriffe des ER-Modells:

- **Entity** (Entität):

  individuell identifizierbares Objekt der Wirklichkeit
- **Entitätstyp**:

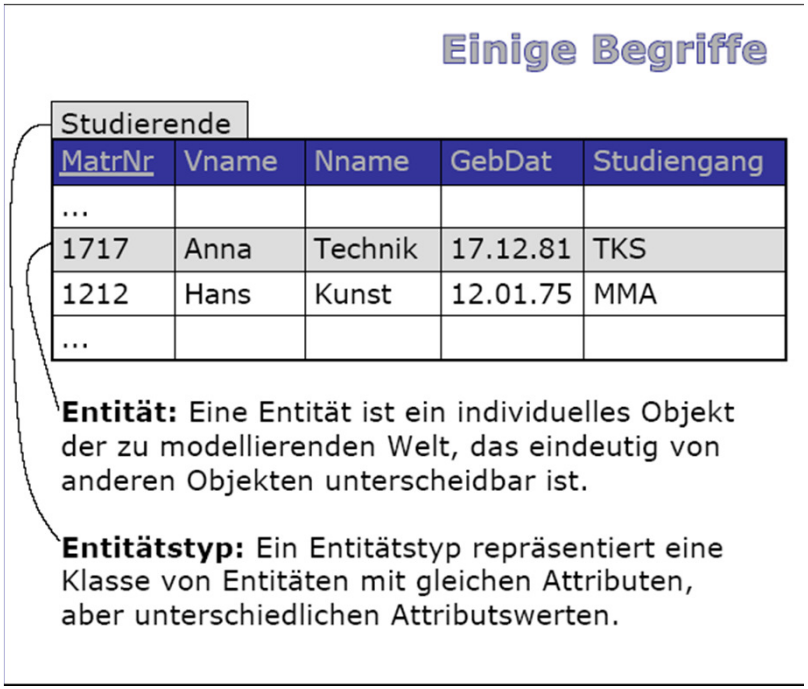
  Typisierung gleichartiger Entitäten z. B. Angestellter, Projekt, Buch, Autor, Verlag
- **Beziehungen**:

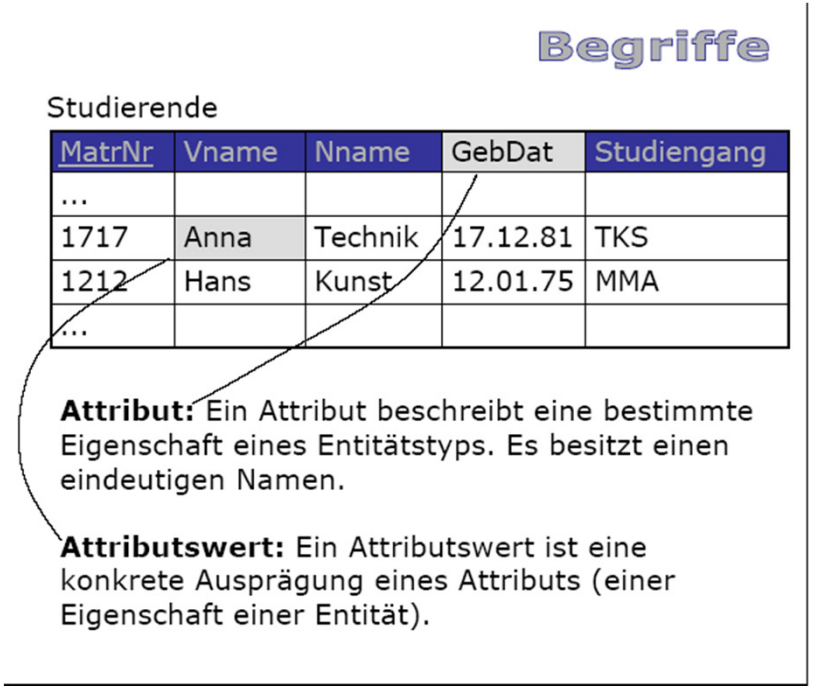
  Verknüpfung / Zusammenhang zwischen zwei oder mehreren Entitäten
- **Eigenschaften** (Attribute):

  Was über eine Entität (im Kontext) von Interesse ist; z. B. das Eintrittsdatum des Angestellten Müller. Beschreibt eine gewisse Eigenschaft
- **Beziehungstyp**:

  Gibt die Art der Beziehung zwischen Entitäten an

- ER-Modell – Entity Relationship Modell





• ER-Modell – Entity Relationship Modell

Begriffe

Studierende

MatrNr	Vname	Nname	GebDat	Studiengang
...				
1717	Anna	Technik	17.12.81	TKS
1212	Hans	Kunst	12.01.75	MMA
...				

Identifikationsschlüssel: Ist ein Attribut oder eine *Kombination* von Attributen. Er identifiziert jede Entität einer Entitätsmenge eindeutig und ändert sich während der Existenz einer Entität nicht. Gibt es mehrere voneinander unabhängige Attribute, die identifizierend wirken können, spricht man von Schlüssel-kandidaten.

Ein Schlüssel dient in einer relationalen Datenbank dazu, Datensätze (Entitäten) einer Relation (Tabelle) eindeutig zu identifizieren.

Primärschüssel (Primary key):

Attribut welche eine Entität eindeutig kennzeichnet. (Attributwert ist für alle Entitäten unterschiedlich).

Wird in referenzierten Tabellen als *Fremdschlüssel* verwendet.

Sekundärschlüssel (Secondary key):

Attribut welches einige Entitäten kennzeichnet. z.B.: PLZ. Dient meist als Indizes zum schnellen suchen in einer Tabelle

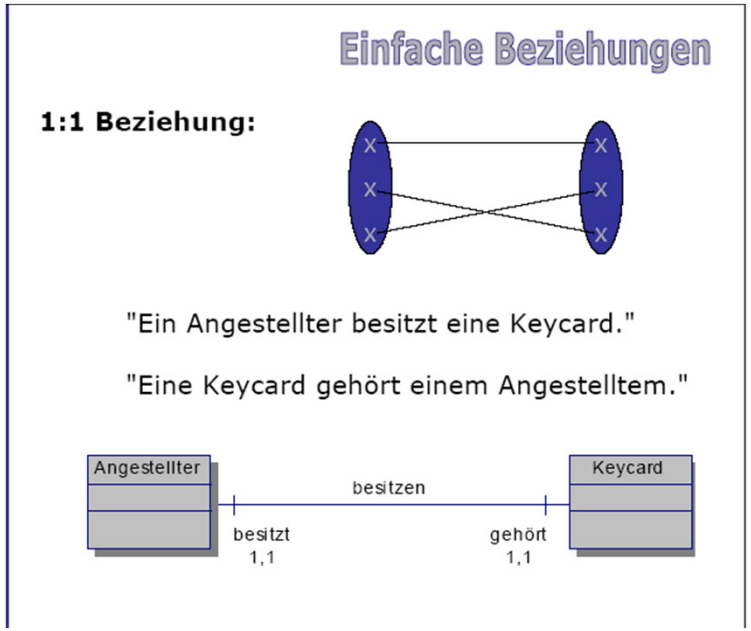
Fremdschlüssel (Foreign key):

Dient als Verweis zwischen 2 Relationen. Fremdschlüssel der einen Relation ist meist Primärschlüssel der anderen Relation

• ER-Modell – Beziehungen

Beziehungen geben an wie einzelne Entitäten zueinander „abhängig“ sind (zueinander in Beziehung stehen)

1:1 Beziehung

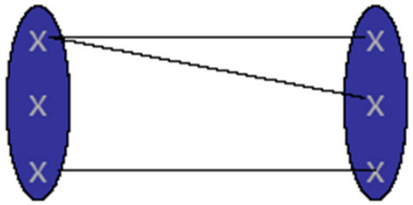


- ER-Modell – Beziehungen

1:N Beziehung

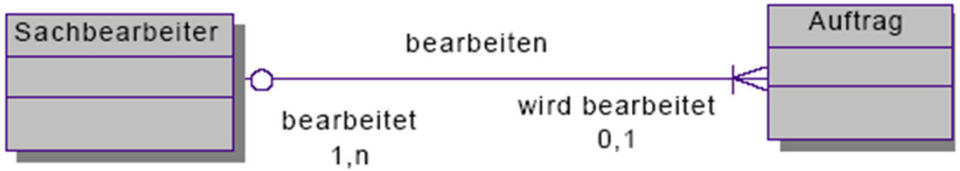
Einfache Beziehungen

1:n Beziehung:



"Ein Sachbearbeiter bearbeitet **einen oder mehrere** Aufträge."

"Ein oder mehrere Aufträge werden von **einem oder keinem** Sachbearbeiter bearbeitet."

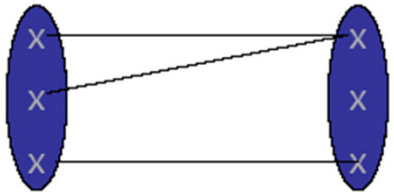


- ER-Modell – Beziehungen

N:1 Beziehung

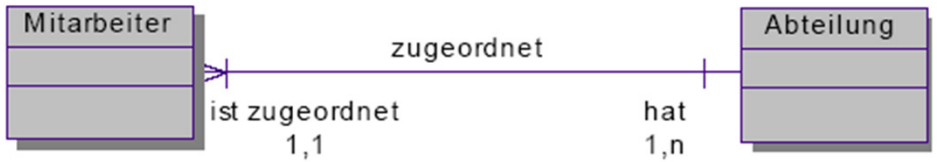
Einfache Beziehungen

n:1 Beziehung:



"Ein Mitarbeiter ist **einer** Abteilung zugeordnet."

"Eine Abteilung hat **einen oder mehrere** Mitarbeiter."

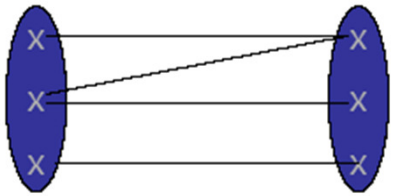


- ER-Modell – Beziehungen

M:N Beziehung

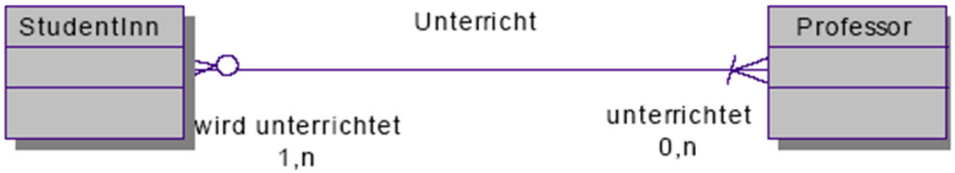
Einfache Beziehungen

m:n Beziehung:



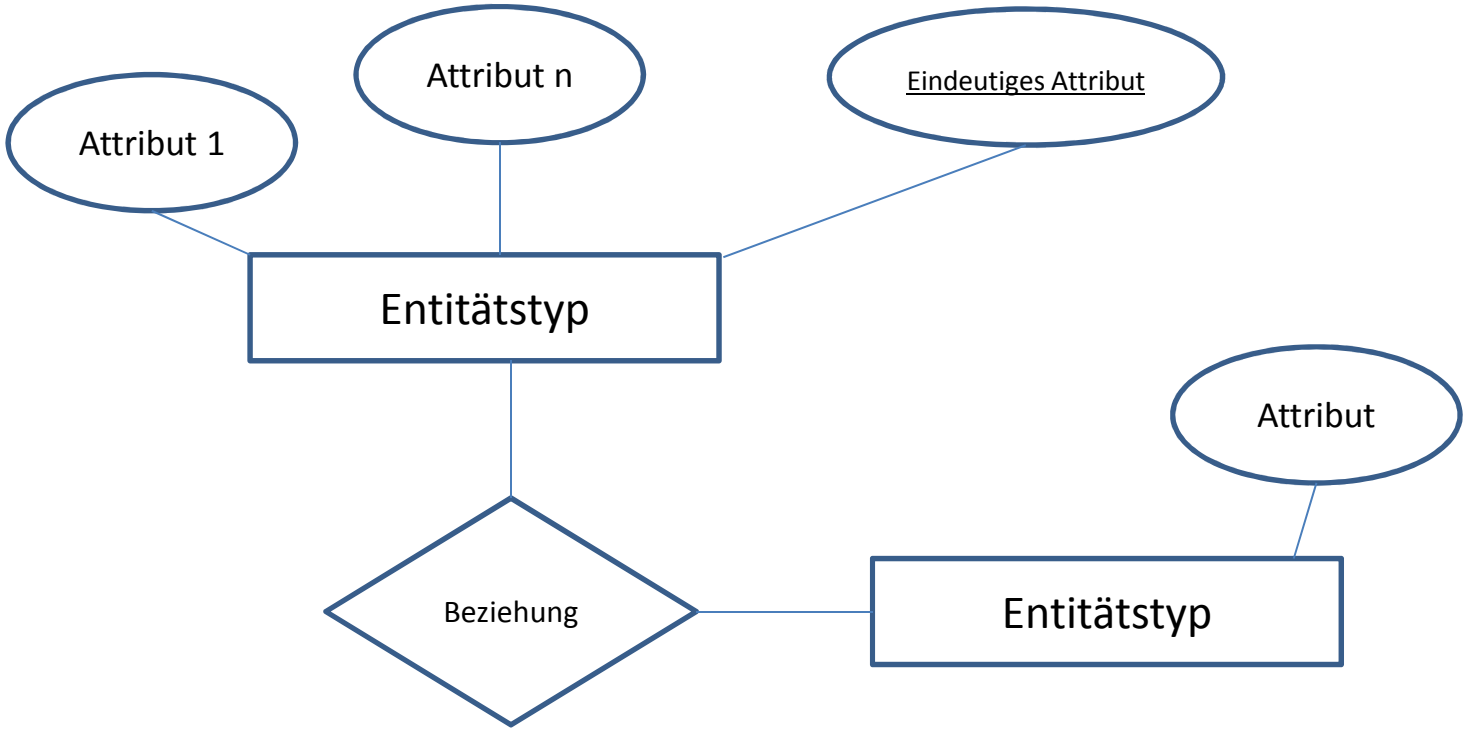
"Ein Student wird von **einem oder mehreren** Professoren unterrichtet."

"Ein Professor unterrichtet **mehrere** Studenten."

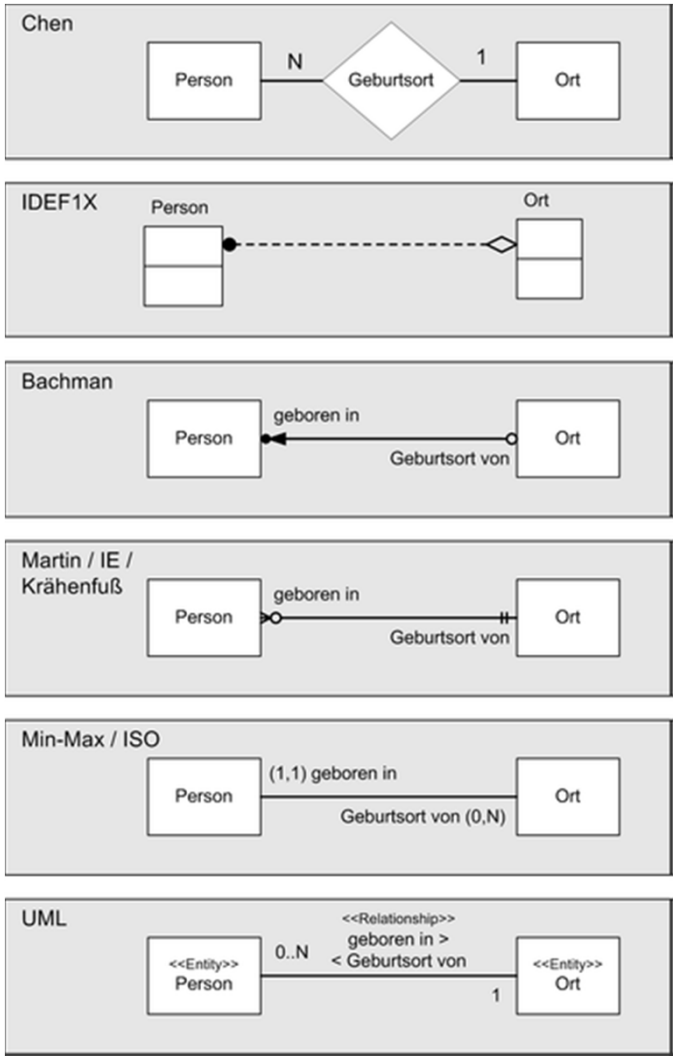


• ER-Modell – Darstellung

Entitätstypen und die Beziehungstypen werden grafisch in einem ER-Diagramm dargestellt. Dazu haben sich verschiedene Darstellungsformen entwickelt, welche allerdings alle die gleichen Informationen darstellen.



- ER-Modell – Darstellung



ER-Modell – Modellierung

Das bisher präsentierte über Datenbanken war eher theoretischer Natur. All das wird benötigt um ein konkretes Problem der realen Welt mit Hilfe einer Datenbank erfassen und abbilden zu können.

Der Schritt der Datenmodellierung dient nun dazu, die reale Welt in einer Datenbankstruktur abzubilden. Dabei müssen die verwendeten Objekte (Entitätstypen), Eigenschaften (Attribute) und deren Beziehungen zueinander gefunden und abgebildet werden.

Anforderungen an eine Software (Datenbank) werden oft schriftlich formuliert. Um aus diesen Formulierungen eine Datenbankstruktur ableiten zu können, kann für eine erste Analyse die Anforderung wie folgt durchgearbeitet werden. Daraus ergeben sich oft gute Ansätze für die Datenbankstruktur.

1. Analysiere alle Nomen im Text, sie könnten Entitätstypen oder aber Attribute sein.
2. Analysiere alle Verben im Text, sie können auf Beziehungen zwischen Entitätstypen hinweisen.
3. Analysiere alle Adjektive im Text, sie können auf Attribute hinweisen.

Beispiel:

Ein Kunde soll eine Bestellung im Webshop tätigen können. Er muss sich zuvor im System registrieren. Dabei sollen sein Name, Alter, Wohnort erfasst werden. Eine Bestellung soll eindeutig einem Kunden zugewiesen werden können indem sie eine eindeutige Nummer bekommt. Zusätzlich soll das Datum der Bestellung, der Status und der Gesamtpreis erfasst werden. Die einzelnen Status einer Bestellung sollen einheitlich erfasst werden.

ER-Modell – Modellierung

Wurde der Text analysiert bietet sich folgende Vorgehensweise an:

1. Abstecken des Problemrahmens (welche Teile der Anforderungen betreffen das Datenbankmodell)
2. Auswahl der Entitätstypen
3. Festlegen der Beziehungen
4. Definition der Attribute
5. Definition der Schlüssel
6. Überführung in ein Relationales Modell

Beispiel:

Ein Kunde soll eine Bestellung im Webshop tätigen können. Er muss sich zuvor im System registrieren. Dabei sollen sein Name, Alter, Wohnort erfasst werden. Eine Bestellung soll eindeutig einem Kunden zugewiesen werden können indem sie eine eindeutige Nummer bekommt. Zusätzlich soll das Datum der Bestellung, der Status und der Gesamtpreis erfasst werden. Die einzelnen Status einer Bestellung sollen einheitlich erfasst werden.

• SQL Abfragesprache

SQL = **S**tructured **Q**uery **L**anguage (Strukturierte Abfragesprache)

SQL ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen (Definition lt. Wikipedia)

SQL Befehle lassen sich in 3 Gruppen unterteilen:

- ✓ **DDL – Data Definition Language**
  Befehle zur Definition von Tabellen und anderer Datenstrukturen
- ✓ **DCL – Data Control Language**
  Befehle zur Kontrolle der Zugriffberechtigungen
- ✓ **DML – Data Manipulation Language**
  Befehle zur Datenmanipulation und Datenabfrage

• SQL Abfragesprache

SQL = **S**tructured **Q**uery **L**anguage (Strukturierte Abfragesprache)

Klasse	DDL	DCL	DML
Beispiele	Datenbank erzeugen **CREATE DATABASE** Tabelle erzeugen **CREATE TABLE** Tabellenaufbau ändern **ALTER TABLE** Tabelle löschen **DROP TABLE** Tabelle umbenennen **RENAME** Virtuelle Tabelle erzeugen **CREATE VIEW**	Zugriffsrechte gewähren **GRANT** Zugriffsrechte entziehen **REVOKE**	Tabelle abfragen **SELECT** Tabellenzeile löschen **DELETE** Tabellenzeile einfügen **INSERT** Tabellendaten ändern **UPDATE**

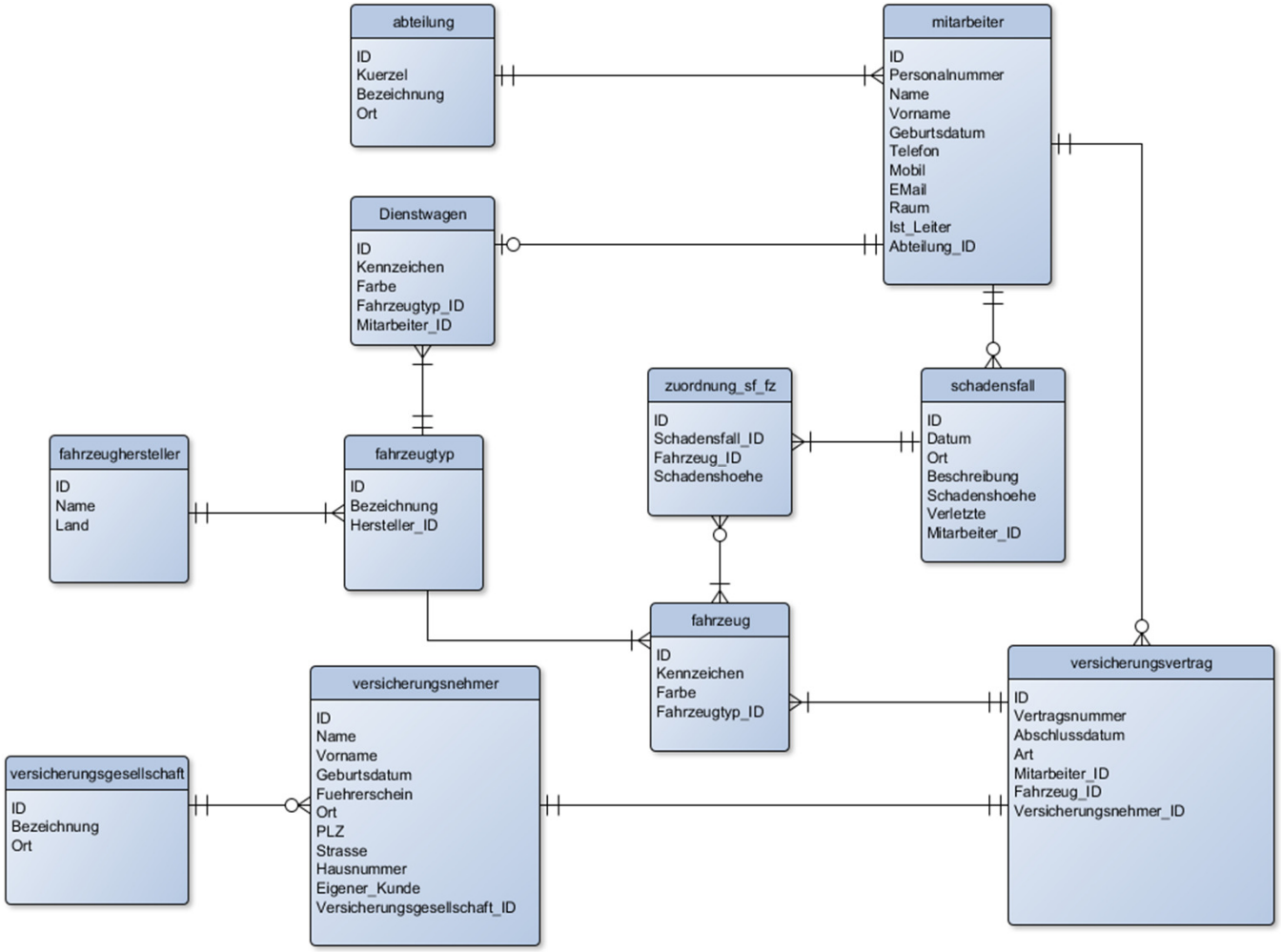
• DML - SELECT

Die SQL-Abfrage erfolgt mit dem Befehl SELECT unter Angabe von bis zu sechs Komponenten. Die allgemeine Syntax hat die Gestalt:

```
SELECT [ALL | DISTINCT] {spalten | *}
FROM tabelle [alias] [tabelle [alias]] ...
WHERE {bedingung | unterabfrage}]
GROUP BY spalten [HAVING {bedingung | unterabfrage}]]
ORDER BY spalten [ASC | DESC]...];
```

Klausel	Erläuterung
SELECT [DISTINCT]	**Wähle** die Werte aus der/den Spalte(n) **[mehrfache Datensätze nur einmal]**...
FROM	... **aus** der Tabelle bzw. den Tabellen ...
WHERE	... **wobei** die Bedingung(en) erfüllt sein soll(en) ...
GROUP BY	... und **gruppiere** die Ausgabe von allen Zeilen mit gleichem Attributwert zu einer einzigen ...
HAVING	... **wobei darin** folgende zusätzliche Bedingung(en) gelten müssen/muss ...
ORDER BY [ASC/DESC]	... und sortiere nach den Spalten **[auf- bzw. absteigend]**.

- Beispieldatenbank - Aufbau



• SELECT - Beispiele

Abfragen ALLER Spalten aus der Tabelle „Abteilung“:

```
SELECT * FROM Abteilung;
```

ID	Kuerzel	Bezeichnung	Ort
1	Fibu	Finanzbuchhaltung	Dortmund
2	Albu	Anlagenbuchhaltung	Dortmund
3	Kore	Kostenrechnung	Dortmund
4	Kopl	Kostenplanung	Dortmund
5	Vert	Vertrieb	Essen
6	Lagh	Lagerhaltung	Bochum
7	Prod	Produktion	Bochum
8	ScAb	Schadensabwicklung	Essen
9	Pers	Personalverwaltung	Essen
10	Soz	Sozialverwaltung	Essen
11	Ausb	Ausbildung	Herne
12	Fue	Forschung und Entwicklung	Bochum

ID	Kuerzel	Bezeichnung	Ort
1	Fibu	Finanzbuchhaltung	Dortmund
2	Albu	Anlagenbuchhaltung	Dortmund
3	Kore	Kostenrechnung	Dortmund
4	Kopl	Kostenplanung	Dortmund
5	Vert	Vertrieb	Essen
6	Lagh	Lagerhaltung	Bochum
7	Prod	Produktion	Bochum
8	ScAb	Schadensabwicklung	Essen
9	Pers	Personalverwaltung	Essen
10	Soz	Sozialverwaltung	Essen
11	Ausb	Ausbildung	Herne
12	Fue	Forschung und Entwicklung	Bochum

Abfragen der Spalten „ID“ und „Ort“ aus der Tabelle „Abteilung“:

```
SELECT ID, Ort FROM Abteilung;
```

id	ort
1	Dortmund
2	Dortmund
3	Dortmund
4	Dortmund
5	Essen
6	Bochum
7	Bochum
8	Essen
9	Essen
10	Essen
11	Herne
12	Bochum

• SELECT - Beispiele

Abfragen ALLER Spalten aus der Tabelle „Abteilung“:

```
SELECT Ort FROM Abteilung;
```

Aus diesem Beispiel ist ersichtlich das z.B. der Ort „Dortmund“ mehrfach vorkommt.
Will man nun dieses Mehrfachvorkommen vermeiden ist das Schlüsselwort „DISTINCT“ zu verwenden.

```
SELECT DISTINCT Ort FROM Abteilung;
```

Manchmal kann es Hilfreich sein die Ergebnisspalte(n) umzubenennen. SQL bietet hierfür die Möglichkeit eines Aliasnamen.

```
SELECT Ort Wohnort, ID
Identifikationsnummer FROM Abteilung;
```

ID	Kuerzel	Bezeichnung	Ort
1	Fibu	Finanzbuchhaltung	Dortmund
2	Albu	Anlagenbuchhaltung	Dortmund
3	Kore	Kostenrechnung	Dortmund
4	Kopl	Kostenplanung	Dortmund
5	Vert	Vertrieb	Essen
6	Lagh	Lagerhaltung	Bochum
7	Prod	Produktion	Bochum
8	ScAb	Schadensabwicklung	Essen
9	Pers	Personalverwaltung	Essen
10	Soz	Sozialverwaltung	Essen
11	Ausb	Ausbildung	Herne
12	Fue	Forschung und Entwicklung	Bochum

• SELECT – Bespiele WHERE

Bisher wurden immer alle Datensätze einer Tabelle ausgegeben. Meist sind aber nur wenige mit bestimmten Eigenschaften von Interesse. Mit Hilfe der WHERE-Bedingung können diese eingeschränkt werden.

Mehrere Einschränkungen werden mittels AND OR NOT verknüpft.

Beispiel: Alle Datensätze mit Ort = Dortmund selektieren.

```
SELECT * FROM Abteilung WHERE Ort = ‚Dortmund‘;

SELECT * FROM Abteilung WHERE Ort <> ‚Dortmund‘;

SELECT * FROM Abteilung WHERE Ort = ‚Dortmund‘ OR Ort
= ‚Essen‘;

SELECT * FROM Abteilung WHERE Ort like ‚Dort%‘;

SELECT * FROM Abteilung WHERE Ort like ‚%o%‘;
```

ID	Kuerzel	Bezeichnung	Ort
1	Fibu	Finanzbuchhaltung	Dortmund
2	Albu	Anlagenbuchhaltung	Dortmund
3	Kore	Kostenrechnung	Dortmund
4	Kopl	Kostenplanung	Dortmund
5	Vert	Vertrieb	Essen
6	Lagh	Lagerhaltung	Bochum
7	Prod	Produktion	Bochum
8	ScAb	Schadensabwicklung	Essen
9	Pers	Personalverwaltung	Essen
10	Soz	Sozialverwaltung	Essen
11	Ausb	Ausbildung	Herne
12	Fue	Forschung und Entwicklung	Bochum

• SELECT - Aggregatsfunktionen

SQL bietet auch eine Möglichkeit gewissen statistische Auswertungen durchzuführen.

- ✓ `AVG() - Returns the average value`
- ✓ `COUNT() - Returns the number of rows`
- ✓ `FIRST() - Returns the first value`
- ✓ `LAST() - Returns the last value`
- ✓ `MAX() - Returns the largest value`
- ✓ `MIN() - Returns the smallest value`
- ✓ `SUM() - Returns the sum`

COUNT berechnet die Zahl der Zeilen im Ergebnis.

Anzahl Orte ermitteln: `SELECT COUNT(Ort) FROM Abteilung`
Ausgabe Anzahl an Datensätzen: `SELECT COUNT(*) FROM Abteilung;`

Übung: Bestimmen sie den Minimalen, Maximalen und Durchschnittlichen Schadenswert. (Tabelle: zuordnung_sf_fz)

ID	Kuerzel	Bezeichnung	Ort
1	Fibu	Finanzbuchhaltung	Dortmund
2	Albu	Anlagenbuchhaltung	Dortmund
3	Kore	Kostenrechnung	Dortmund
4	Kopl	Kostenplanung	Dortmund
5	Vert	Vertrieb	Essen
6	Lagh	Lagerhaltung	Bochum
7	Prod	Produktion	Bochum
8	ScAb	Schadensabwicklung	Essen
9	Pers	Personalverwaltung	Essen
10	Soz	Sozialverwaltung	Essen
11	Ausb	Ausbildung	Herne
12	Fue	Forschung und Entwicklung	Bochum

• SELECT – Anfragen über mehrere Tabellen

Bisherige Abfragen bezogen sich immer nur auf 1 Tabelle.
Gerade bei relationalen Datenbanken, ist aber eine Verknüpfung mehrerer Tabellen wünschenswert.

Beispiel: Abfrage aller Daten über Abteilungen UND Mitarbeiter:
SELECT * FROM Abteilung, Mitarbeiter;

Als Ergebnis wird das **Kreuzprodukt** der beiden Tabellen zurückgeliefert.
Beim Kreuzprodukt wird jeder Eintrag der einen Tabelle mit jedem Eintrag der anderen Tabelle verknüpft.

Übung:
Ermitteln Sie die Anzahl der Zeilen für die Tabelle Mitarbeiter und Abteilung;
Ermitteln Sie die Anzahl der Zeilen des Kreuzproduktes für die Tabelle Mitarbeiter und Abteilung
Frage: In welcher Beziehung stehen die Ergebnisse des Kreuzproduktes zu den beiden Einzelwerten der Tabelle??

Das Kreuzprodukt enthält meist für das Ergebnis unsinnige Werte. Diese können durch Angabe einer geeigneten WHERE-Klauses eliminiert werden.
Typischerweise wird hier auf eine Primary/Foreign Ky Beziehung eingeschränkt.

Beispiel: Alle Mitarbeiter der Abteilung ‚Vertrieb'.
SQL-Query = ???

• Datenbank – Beispiel

Aufgabenstellung:
Entwerfen sie eine Datenbank die folgenden Businesscase abdecken soll.
Ein Musiklabel will alle Interpreten und deren bisher produzierten Titel abspeichern. Folgende Daten sollen für den Interpreten erfasst werden: Name, Vorname, Künstlername sowie sein Geburtsdatum.
Ein Musiktitel muss eindeutig einem Interpreten (und nur einem, keine Musikgruppen) zugeordnet werden können. Ein Musiktitel zeichnet sich durch seinen Namen, sein Erscheinungsdatum und seinen Lyriktext aus.

Entwerfen Sie ein entsprechendes Datenbankdesign. Für die Modellierung des ER-Modelles verwenden sie die „Krähenfußnotation“.

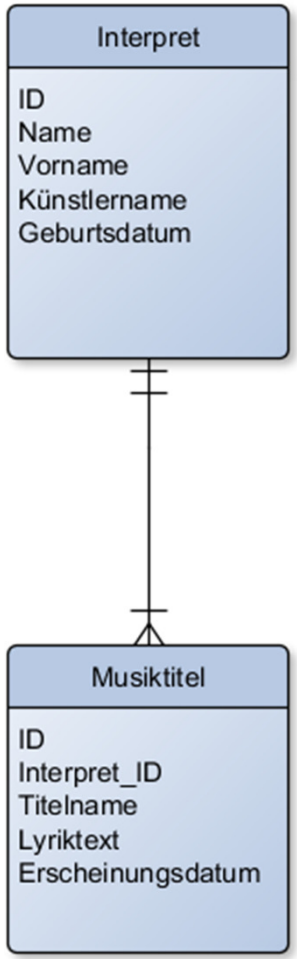
Ausgehend von Ihren Datenbankdesign, stellen sie die Select-Statements für folgende Abfragen auf.

1) Es sollen ALLE Titel aller Interpreten ausgegeben werden deren Erscheinungsdatum vor dem 1.1.1980 liegt.

2) Geben sie alle Musiktitel des Interpreten (Künstername) ‚Falco‘ aus.

```
1) select * from Musiktitel m, Interpret int where
   m.Interpret_ID = int.ID AND Erscheinungsdatum <
   `1.1.1980`;

2) select * from Musiktitel where  Interpret_ID =
   select id from Interpret where Kuenstlername =
   `Falco`;
```



- SELECT – Gruppierungen in SQL

TODO

- Fragen!!!!