2. Datenbanken (Herr Butscher)

2.1. Was sind Datenbanken und worin liegen deren Vorzüge?

Antwort: Eine Datenbank ist ein elektronisches Archiv, in dem inhaltlich zusammenhängende Daten für einen zentralen Zweck strukturiert gesammelt und gespeichert werden. Eigenschaften/Vorteile sind:

- Daten sind zentral organisiert
- Daten zentral, effizient, konsistent* und permanent gespeichert
- Mehrere Nutzer können gleichzeitig auf die Daten zugreifen
- Anwendungsprogramme sind von der physischen Speicherung der Daten unabhängig
- Anwendungen haben nur über das DBMS* Zugriff auf die Daten

***DB:** Datenbank ***DBS:** Datenbank-System ***DBMS:** Datenbank-Management-System

**konsistent = übereinstimmend, beständig*

DB | Hierarchische Datenbanken = Hierarchische Datenbanken haben ein Datenmodell in der Art eines **Baumes**. Die Adressverknüpfungen werden mit den Daten gespeichert. Ein Datensatz kann höchstens mit einem übergeordneten sowie mehreren untergeordneten Datensätzen in Beziehung stehen.

DB | Objektorientierte Datenbanken = Daten werden als Objekte mit ihren Methoden und Attributen in der Datenbank gespeichert. Die Definition eines Objekts ist eng verwandt mit der bei der objektorientierten Programmierung.

DB | Relationale Datenbanken = Daten werden als Menge von Tabellen (Relationen) mit festen Anzahl von Spalten und beliebig vielen Zeilen dargestellt. Jeder Datensatz wird als Zeile dargestellt. Attribute stehen in Spalten. Über Primärschlüssel bleiben alle Zeilen in der Tabelle identifizierbar (!)

Als Datenbank-System wird die Gesamtheit von DBMS und allen Datenbanken, die von dem DBMS verwaltet werden, bezeichnet.

2.2. Was versteht man unter einem Entity-Relationship-Model (ERM)? Bilden Sie ein Anwendungsbeispiel in einer ERM-Notation ab. (Erklärung: https://youtu.be/F5rTvnbjPq8)

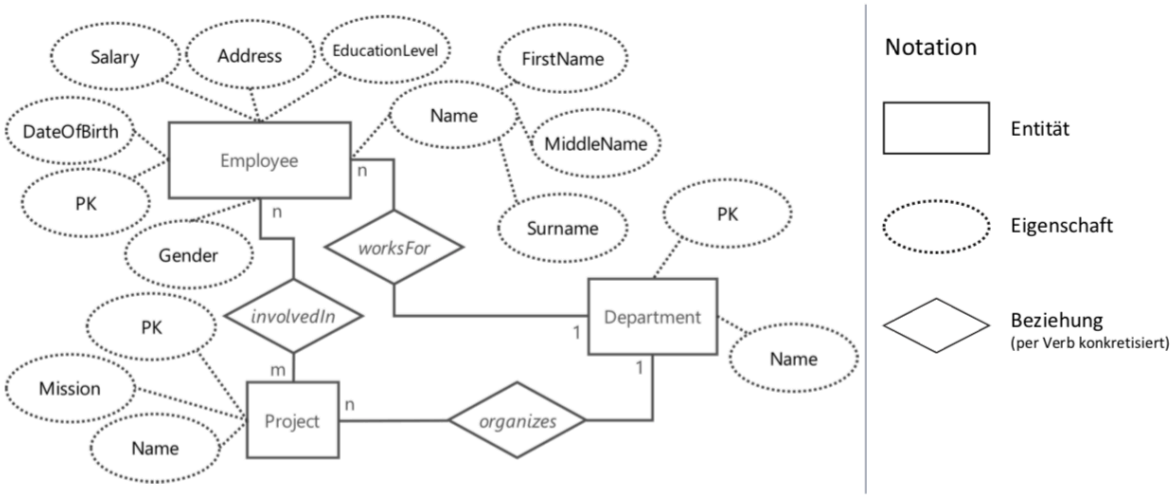
Antwort: Ein Schema von Entity- und Beziehungsmengen wird als Entity-Relationship-Model (ERM) bezeichnet. **Hiermit kann die Datenbankstruktur vor der Programmierung geplant werden.** Man unterscheidet zwischen Entitäten, Attributen und Beziehungen:

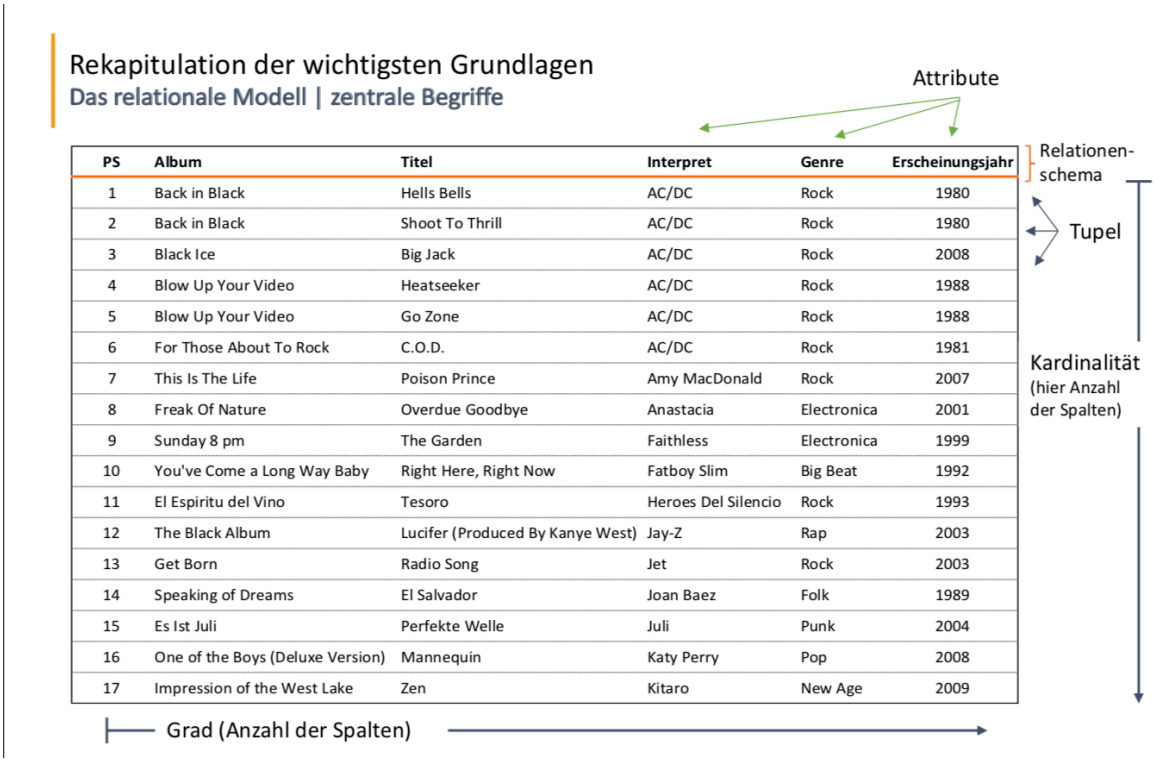
- Entität: Objekt aus der realen Welt. Es wird durch Attribute beschrieben.
  (Mitarbeiter, Abteilungen, Projekte, Kunden, Aufträge, usw.)

- Beziehungen /Relations: Beziehungen/Relations zwischen zwei Entitäten.
  (Mitarbeiter *arbeiten* in Abteilungen, Kunden *erteilen* Aufträge, Aufträge *enthalten* Projekte

- Attribut: Eigenschaft, die allen Entitäten der Entity- Menge gemeinsam ist
  (Arbeit wird in Stunden gemessen, Aufträge erhalten Auftragsnummern, von Kunden wird Vorname, Nachname, Adresse sowie Telefonnummer erfasst)

Beispiel eines visualisierten ERM:



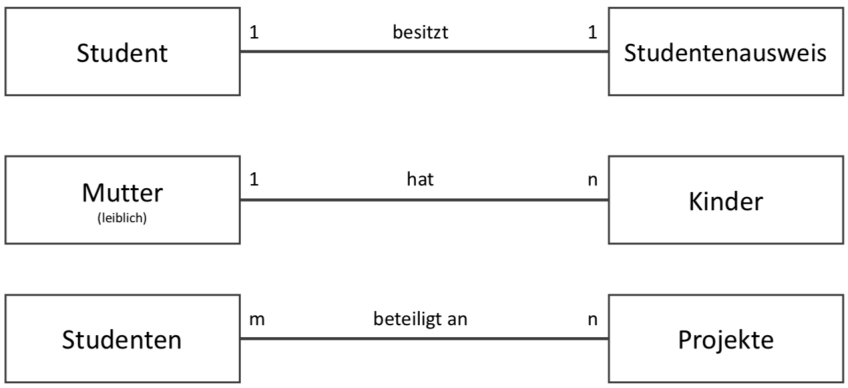


PS	Album	Titel	Interpret	Genre	Erscheinungsjahr
1	Back in Black	Hells Bells	AC/DC	Rock	1980
2	Back in Black	Shoot To Thrill	AC/DC	Rock	1980
3	Black Ice	Big Jack	AC/DC	Rock	2008
4	Blow Up Your Video	Heatseeker	AC/DC	Rock	1988
5	Blow Up Your Video	Go Zone	AC/DC	Rock	1988
6	For Those About To Rock	C.O.D.	AC/DC	Rock	1981
7	This Is The Life	Poison Prince	Amy MacDonald	Rock	2007
8	Freak Of Nature	Overdue Goodbye	Anastacia	Electronica	2001
9	Sunday 8 pm	The Garden	Faithless	Electronica	1999
10	You've Come a Long Way Baby	Right Here, Right Now	Fatboy Slim	Big Beat	1992
11	El Espiritu del Vino	Tesoro	Heroes Del Silencio	Rock	1993
12	The Black Album	Lucifer (Produced By Kanye West)	Jay-Z	Rap	2003
13	Get Born	Radio Song	Jet	Rock	2003
14	Speaking of Dreams	El Salvador	Joan Baez	Folk	1989
15	Es Ist Juli	Perfekte Welle	Juli	Punk	2004
16	One of the Boys (Deluxe Version)	Mannequin	Katy Perry	Pop	2008
17	Impression of the West Lake	Zen	Kitaro	New Age	2009

2.3. Interpretieren Sie die Bedeutung von *„1:1“, „1:n“ und „n:m“* -Beziehungen.

Antwort: Beziehungen lassen sich insgesamt in drei Arten unterteilen:

- **1:1-Beziehung** | Genau eine Beziehung in beide Richtungen
- **1:n-Beziehung** | Keine, eine oder mehrere Beziehungen in eine Richtung
- **n:m-Beziehung** | Keine, eine oder mehrere Beziehungen in beide Richtungen



2.4. Erklären Sie ein relationales Datenmodell hinsichtlich der Grundlagen, den Verstößen und deren Abhilfe sowie FK-/PK-Beziehungen.

Antwort:

1. Normalform (muss in seiner kleinsten Form sein, d.h. atomar): Attribute müssen atomar, das bedeutet in ihrer kleinstmöglichen Einheit sein. D.h. wenn z.B. in einer Zeile 2 zusammengefasste Attribute stehen, müssen diese atomisiert (in mehreren Spalten wie Tupel aufgelöst) werden. Beispiel: Die Adresse darf nicht als einzelnes Attribut verwendet werden, sondern muss in PLZ, Ort, Straße, Hausnummer etc. aufgeteilt werden.

2. Normalform (eindeutig zuordenbar, NSA muss von PS vollständig funktional Abhängig sein, d.h. nur von einem Primärschlüssel abhängen): Die zweite Normalform ist erfüllt, wenn er sich in der ersten Normalform (1NF) befindet und jedes Nichtschlüsselattribut (NSA) von jedem Schlüsselattribut voll funktional abhängig ist (d.h. eindeutig zuordenbar ist). Alle NSAs müssen eindeutig von einem Primärschlüssel abhängen.

PS	Interpret	Album	Titel	Genre	Erscheinungsjahr
1	AC/DC	Back in Black	Hells Bells	Rock	1980
2	AC/DC	Back in Black	Shoot To Thrill	Rock	1980
3	AC/DC	Back in Black	What Do You Do For Money Honey	Rock	1980
4	AC/DC	Back in Black	Givin The Dog A Bone	Rock	1980
5	AC/DC	Back in Black	Let Me Put My Love Into You	Rock	1980
6	AC/DC	Back in Black	Back In Black	Rock	1980

Beispiel: In der Tabelle werden Songs durch den Primärschlüssel [PS] voneinander unterschieden. Die Attribute [Interpret] und [Album] hängen aber nicht direkt von diesem Primärschlüssel ab; gemäß der 2. NF sind diese Attribute in separaten Tabellen mit eigenen Primärschlüssel zu halten. Problem: Daten können fehlerhaft erfasst werden (z.B. „ac\dc“ oder „Ac/ DC“ für „AC/DC“), sodass jede Zeile zu prüfen wäre. Wird [Interpret] nicht in einer gesonderten Tabelle verwaltet, sind Datenanomalien infolge falscher Schreibweise die Folge.

3. Normalform (keine Redundanz, NSA darf nicht transitiv abhängig sein, d.h. von einem anderen Nichtschlüsselattribut abhängig sein): Ein Relationstyp befindet sich genau dann in der dritten Normalform (3NF), wenn er sich in der zweiten Normalform (2NF) befindet und kein Nichtschlüsselattribut (NSA) transitiv von einem Schlüsselattribut abhängt. Alle Spalten dürfen also nur vom Schlüsselattribut und nicht von anderen Werten abhängen. Einfach gesagt: Wenn kein Nichtschlüsselattribut eine Abhängigkeit zu einem anderen Nichtschlüsselattribut hat, ist es ok.

PS_Song	PS_Album	FS_Komponist	FS_Genre	Name_Song	Erscheinungsjahr
1	1067	102031	10201	Hells Bells	1980
1	1067	102032	10201	Shoot To Thrill	1980
2	1068	102033	10201	Big Jack	2008
3	1069	102034	10201	Heatseeker	1988
3	1069	102035	10201	Go Zone	1988
4	1070	102036	10201	C.O.D.	1981
5	1071	102037	10201	Poison Prince	2007
6	1072	102038	10201	Tesoro	1993
7	1073	102039	10206	Overdue Goodbye	2001
8	1074	102040	10206	The Garden	1999
9	1075	102041	10203	Right Here, Right Now	1992

Verstoß gegen 3 NF: „Erscheinungsjahr“ hängt hier nicht vom Song (PS_Song), sondern vom „Album“ ab. Die Spalte [Erscheinungsjahr] muss somit aus der Tabelle „Song“ entfernt und in „Album“ eingefügt werden.

2.4.1. Einhaltung der Integritätsbedingungen

Der Begriff Integritätsbedingung bezeichnet in der Informatik Bedingungen, die an den Zustand eines Prozesses oder einer Datenstruktur gestellt werden. In Bezug auf Datenbanken werden Zustände als konsistent bezeichnet, wenn sie die Integritätsbedingungen erfüllen. Diese sind:

- **Entity-Integrität:** Wenn jede Relation einen Primärschlüssel besitzt, in der Folge jeder Tupel über einen eindeutigen Primärschlüssel zweifelsfrei identifizierbar ist. (Kurz: Jeder Tupel muss über einen Primärschlüssel identifizierbar sein)

- **Semantische Integrität:** Wenn (z.B. durch den Datentyp) die formal korrekte Eingabe gewährleistet wird. Der Anwender etwa in ein Datumsfeld auch tatsächlich nur ein (gültiges) Datum eingeben kann. (Kurz: Datentypen angeben.)

- **Referenzielle Integrität:** Wenn jeder Wert eines Fremdschlüssels einer Relation Wert eines Primärschlüssels in einer anderen Relation ist. Mit anderen Worten: Es darf in der FK-Spalte kein Wert enthalten sein, für den es keine Repräsentation in jener Relation mit den zugehörigen Primärschlüsseln gibt. (Kurz: Hinter jedem FK muss ein PK stehen)

- **Operative Integrität /Ablaufintegrität:** Wenn mehrere Benutzer konkurrierend auf die Datenbank zugreifen können und sichergestellt bleibt, dass die Datenbank anschließend nach wie vor in einem konsistenten Zustand ist.

2.4.2. FK- /PK-Beziehungen:

- Primärschlüssel (Primary Key, PK oder PS): Ein Primärschlüssel kennzeichnet eindeutig einen Datensatz (Tupel) in einer Datenbank. Eine relationale Datenbank ist darauf ausgelegt, die Eindeutigkeit von Primärschlüsseln zu erzwingen, indem nur eine einzelne Zeile mit einem bestimmten Primärschlüsselwert in einer Tabelle zugelassen wird. Dieser sollte so minimal/klein wie möglich sein. Es gibt jedoch auch sog. „Zusammengesetzte Primärschlüssel“, bspw. Name + Geburtstag + Wohnort. Diese wiedersprechen jedoch zumeist dem Minimalprinzip. Optisch ist ein PK in einer Tabelle meist mit einem **Unterstrich** gekennzeichnet.

- Fremdschlüssel (Foreign Key, FK oder FS): Ein Fremdschlüssel ist eine Spalte oder eine Spaltengruppe in einer Tabelle, deren Werte den Werten des Primärschlüssels in einer anderen Tabelle entsprechen. Um eine Zeile mit einem bestimmten Fremdschlüsselwert hinzufügen zu können, muss in der zugehörigen Tabelle eine Zeile mit demselben Primärschlüsselwert vorhanden sein (siehe auch referenzielle Integrität). Optisch ist ein FK in einer Tabelle meist mit einem **Überstrich** gekennzeichnet.

2.5. Nennen Sie die wichtigsten SQL –Funktionen und Aufgaben.

Antwort: SQL (Structured Query Language) ist eine standardisierte Datenbanksprache und dient als:

- Datendefinitionssprache (Data Definition Language, DDL),
- Datenmanipulationssprache (Data Manipulation Language, DML),
- Datenabfragesprache (Data Retrieval Language, DRL) und
- Rechteverwaltung und Transaktionskontrolle (Data Control Language, DCL)

Aufgaben von SQL:

1. **Anlegen/Erzeugen** von Tabellen inkl. Schlüsseln, Attributen usw.
2. **Ändern/Bearbeiten** von Tabellen, z. B. Beschriftungen ändern
3. **Abfragen** von Tabellen samt Daten

Vorteile von SQL:

- Einfache wie logische Struktur /Aufbau
- An englische Sprache angelehnt (einfach zu verstehen)
- **Case-Insentive** (Keine Unterscheidung von Groß- und Kleinschreibung)

2.6. Zeigen Sie einfache SQL-Befehle auf und interpretieren Sie die Statements.

Antwort: Wesentliche SQL-Befehle sind:

2.6.1 Anlegen einer neuen Tabelle mit SQL (DDL)

Attribute sind in einer Relation die Spalten.
Nach dem Attribut gilt es, den Datentyp festzulegen (semantische Integrität). Diese können sein:

- `char(n)` → Zeichenkette mit feste Länge.
  Bspw. die deutschen Postleitzahlen char(5)
- `varchar(n)` → Zeichenkette mit variabler Länge
  Bspw. Namen und Vornamen varchar (30)
- `int` → Legt den Datentyp auf Zahlen fest.

```
CREATE TABLE Professoren (
PersNr int primary key,
Name varchar(255),
Addresse varchar(255)
);
```

Mit wird `primary key` der Primärschlüssel gesetzt.

Ergebnis:

PersNr	*Name*	*Addresse*

2.6.2. Ändern einer bestehenden Tabellenstruktur (DDL)

- **ADD** fügt der bestehenden Tabelle Professoren die Spalte Postleitzahl mit dem Datentyp varchar(5) zu.
- **MODIFY** ändert den Datentyp bei der Spalte Postleitzahl von varchar in int.
- **DROP** löscht aus der bestehenden Tabelle Professoren die Spalte Postleitzahl.
- **CHANGE** ändert den Spaltennamen „Addresse“ in „Adresse“ um.

```
ALTER TABLE Professoren
ADD Postleitzahl varchar(5)

ALTER TABLE Professoren
MODIFY Postleitzahl int

ALTER TABLE Professoren
DROP Postleitzahl

ALTER TABLE Professoren
CHANGE Addresse Adresse
varchar(255)
```

2.6.3. Löschen einer Datenbank/Tabelle mit SQL (DDL)

```
DROP DATABASE Verwaltung
DROP TABLE Professoren
```

2.6.4. Einfügen von Tupeln (DML)

Jeder Tupel wird durch ( ) angegeben bzw. umgrenzt.
Es müssen so viele Einträge wie Spalten der Tabelle vorhanden sein. In diesem Beispiel besteht die Tabelle „Professoren“ aus drei Spalten, d.h. es müssen drei
Einträge pro Klammerausdruck vorkommen. NULL steht für kein Wert, zählt aber als gültiger Eintrag (sofern „NULL“ als Eingabe zulässig ist).

```
INSERT INTO Professoren
VALUES
(0815, “Heinze“, “Domstraße“),
(0816, “Meunier“, NULL)
```

– ALTERNATIV –

Beim Insert von Zeichenketten (Datentyp varchar) ist es wichtig, die einzugebenden Zeichenketten zusammen mit ”” einzugeben. Das Datenbanksystem gibt bei Abfragen die Zeichenketten ohne ”” zurück.

```
INSERT INTO Professoren
(PersNr, Name, Adresse)
VALUES
(0815, “Heinze“, “Domstraße“),
(0816, “Meunier“, NULL)
```

2.6.5. Löschen von Tupeln (DML)

- **PersNr = 0815** löscht jenen Datensatz, bei dem in der Spalte PersNr der Wert 0815 vorkommt.

```
DELETE FROM Professoren
WHERE
PersNr = 0815
```

- **PersNr <= 0815** löscht alle Datensätze, bei denen die PersNr kleiner gleich dem Wert 0815 ist.

```
DELETE FROM Professoren
WHERE
PersNr <= 0815
```

- **Name = “Heinze”** löscht jene Zeilen, bei dem in der Spalte Name, der Wert Heinze vorkommt. Wichtig ist hier wieder das Hochkomma ””, da Heinze eine Zeichenkette ist.

```
DELETE FROM Professoren
WHERE
Name = “Heinze”
```

Wichtig: Die obigen SQL-Statement löschen stets die gesamte Zeile, nicht nur den Werte in der jeweiligen Spalte, die der Bedingung genügen. Gäbe es mehrere Datensätze mit dem Eintrag Heinze, so wären diese allesamt gelöscht, auch wenn es unterschiedliche Personen mit dem Name Heinze wären.

```
DELETE FROM Professoren
WHERE
PersNr = 0815
```

2.6.6. Verändern von Tupeln (DML)

Der Befehl ändert für alle Zeile(n) mit Name gleich Huber in den neuen Wert Heinze-Huber um.

```
UPDATE Professoren
SET
Name = "Heinze-Huber"
WHERE Name = "Heinze"
```

Vorher	0815	Heinze	Domstraße
Nachher	0815	Heinze-Huber	Domstraße

2.6.7. Löschen aller Tupeln (DML)

Der Befehl löscht alle Datensätze aus der Tabelle Professoren, lässt aber die Tabellenstruktur bestehen.

```
TRUNCATE Professoren
```

Vorher

PersNr	*Name*	*Adresse*
0815	Heinze-Huber	Domstraße
0816	Meunier	NULL

Nachher

PersNr	*Name*	*Adresse*

2.6.8. Einfachste Selektion (ohne Vergleichsausdruck)

Irgendwas = Spalte(n)/ Attribute einer Relation („was“)
Irgendwo = Quellrelation(en) („woher“)

```
SELECT Irgendwas
FROM Irgendwo
```

Fragt alle vorhandenen Spalte der Relation Irgendwo ab. Zwar erspart der Befehl Schreibarbeit, sollte in der Praxis aber mit Bedacht verwendet werden. Select * kann ein Performance-Killer werden (lange Wartezeiten, bis das System die Ergebnismenge zur Verfügung stellt) oder riesige, kaum handelbare Ergebnismengen zurückliefern.

```
SELECT *
FROM Irgendwo
```

`LIKE` Liefert in der Ergebnismenge nur Datensätze, die beim Attribut Name mit dem Buchstaben H beginnen.

```
SELECT PersNr, Name
FROM Professoren
WHERE Name LIKE "H%"
```

`NOT LIKE` Liefert in der Ergebnismenge nur Datensätze, die beim Attribut Name mit einem anderen Buchstaben als H beginnen („Alles außer H“).

```
SELECT PersNr, Name
FROM Professoren
WHERE Name NOT LIKE "H%"
```

- % steht für kein oder beliebig viele Zeichen
- _ steht für genau ein Zeichen

2.6.9. Selektion & Projektion kombiniert

- **Projektion:** Legt fest, welche Attribute überhaupt in der Ergebnismenge vorkommen sollen.

```
SELECT PersNr, Name
FROM Professoren
```

- **Selektion:** Legt fest, welche Zeilen („Treffer“) in der Ergebnismenge vorkommen

```
WHERE Besoldungsstufe = “W2”
```

In der Ergebnismenge tauchen nur PersNr und Name auf, Besoldungsstufe hingegen nicht, da dieses Attribut nicht beim SELECT-Statement steht. Besoldungsstufe filtert die Daten quasi im Hintergrund.

```
SELECT PersNr, Name
FROM Professoren
WHERE Besoldungsstufe = “W2”
```

2.7. Erklären Sie den Begriff des INNER JOIN.

Antwort: Ein (Inner-) Join („Verbund“) bezeichnet die beiden hintereinander ausgeführten Operationen „Kreuzprodukt“ und „Selektion“. Einfacher ausgedrückt, er selektiert aus zwei oder mehreren Tabellen die gesuchten Ergebnisse und kombiniert gleichzeitig redundante Ergebnisse in einer Spalte.

```
SELECT *
FROM Professoren INNER JOIN Kurse
ON Professoren.Kurs_ID = Kurse.Kurs_ID
```

Über `SELECT *` werden alle Werte ausgewählt. `FROM` und `INNER JOIN` kombinieren die gewählten Tabellen. Würde das jetzt so stehenbleiben, wären Spalten, welche bei beiden Tabellen vorkommen gedoppelt. Bspw. ist sowohl unter Professoren eine `Kurs_ID` (Fremdschlüssel) wie auch bei den Kursen (Primärschlüssel) angegeben. Um diese Doppelung zu vermeiden, legt man sog. Kreuzpunkte:

```
ON Professoren.Kurs_ID = Kurse.Kurs_ID
```

Die Spalte `Kurs_ID`, welche in beiden Tabellen vorkommt wird in eine einzelne Spalte zusammengeführt. `Professoren.Kurs_ID = Kurse.Kurs_ID` sagt dabei aus, aus welcher Tabelle welche Spalten zusammengeführt werden sollen.