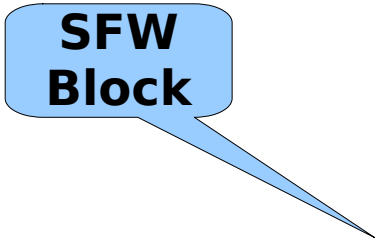
Tag 3
Inhaltsverzeichnis

SQL Survival Kit... (lesend, mit SQLite)

SQL Select-Anweisung (Grundbild)




```
SELECT [DISTINCT] Auswahlliste [AS Neuer Name]+
FROM Quelle
[WHERE Where-Klausel]
[GROUP BY (Group-by-Attribut)+
 [HAVING Having-Klausel]]
[ORDER BY (Sortierungsattribut [ASC|DESC])+];
```

- **S:**
  - Projektionsliste (--> ***Spalten*** selektieren) ("*" für alle)
  - Arithmetische Operationen, Aggregatfunktionen
  - Resultat: eine neue Relation
- **F:**
  - Zu verwendende Relationen
- **W:**
  - Bedingung(en) (--> ***Zeilen*** selektieren)
  - Schachtelung möglich

Daten filtern
Gleichheitsbedingung (1)

Tabellen-name und alias

```
sql> select * from autor a where a.PersNr='12';
+--------+---------+--------+
| persnr | vorname | name   |
+--------+---------+--------+
|     12 | Alfons  | Kemper |
+--------+---------+--------+
```

Die SQL-Schlüsselworte schreibe ich klein

```
sql> select persnr as keypers from autor a where a.persnr='12';
+---------+
| keypers |
+---------+
|      12 |
+---------+
```

*Spalten-*Alias

Daten filtern
Gleichheitsbedingung (2)

```
sql> select distinct a.vorname from autor a;
+---------+
| vorname |
+---------+
| Alfons  |
| Michael |
| Gilles  |
+---------+
```

Duplikate entferne

```
sql> select a.vorname, a.name from autor a
     order by a.vorname, a.name asc;
+---------+--------+
| vorname | name   |
+---------+--------+
| Alfons  | Kemper |
| Gilles  | Maitre |
| Michael | Kofler |
+---------+--------+
```

Sortierung

Daten filtern
Wertebereichsbedingung

```
sql> select * from autor a
       where a.persnr >= 11 and a.persnr <= 35;
+--------+---------+--------+
| persnr | vorname | name   |
+--------+---------+--------+
|     12 | Alfons  | Kemper |
|     34 | Michael | Kofler |
+--------+---------+--------+
```

```
sql> select * from autor a
       where a.persnr between 11 and 35;
+--------+---------+--------+
| persnr | vorname | name   |
+--------+---------+--------+
|     12 | Alfons  | Kemper |
|     34 | Michael | Kofler |
+--------+---------+--------+
```

Daten filtern
Wildcards

"_": genau ein Zeichen
"%": beliebig viele Zeichen

```
sql> select * from autor a where a.name = "K%";
Empty set (0.00 sec)
```

```
sql> select * from autor a where a.name like "K%";
+--------+---------+--------+
| persnr | vorname | name   |
+--------+---------+--------+
|     12 | Alfons  | Kemper |
|     34 | Michael | Kofler |
+--------+---------+--------+
```

Daten filtern
Null ist nicht null

```
sql> alter table autor
       add column bemerkung varchar(20);

sql> select * from autor a where a.bemerkung = null;
Empty set (0.00 sec)

sql> select * from autor a where a.bemerkung is null;
+--------+---------+--------+-----------+
| persnr | vorname | name   | bemerkung |
+--------+---------+--------+-----------+
|     12 | Alfons  | Kemper | NULL      |
|     34 | Michael | Kofler | NULL      |
|     56 | Gilles  | Maitre | NULL      |
+--------+---------+--------+-----------+


sql> alter table autor drop column bemerkung;
```

Komplexe Abfragen strukturieren
Common Table Expression (CTE)

```
WITH query_name1 AS (
      SELECT ...
      )
    , query_name2 AS (
      SELECT ...
        FROM query_name1
          ...
      )
SELECT ...
    FROM query_name1, query_name2, ...

-- Beispiel aus http://www.mysqltutorial.org/mysql-cte/

WITH customers_in_usa AS (
    SELECT customerName, state FROM customers WHERE country = 'USA'
)
SELECT customerName FROM customers_in_usa WHERE state = 'CA';
```

Übung 1
Siehe letztes Slide

Bitte Übung 1 machen,
siehe letztes Slide.

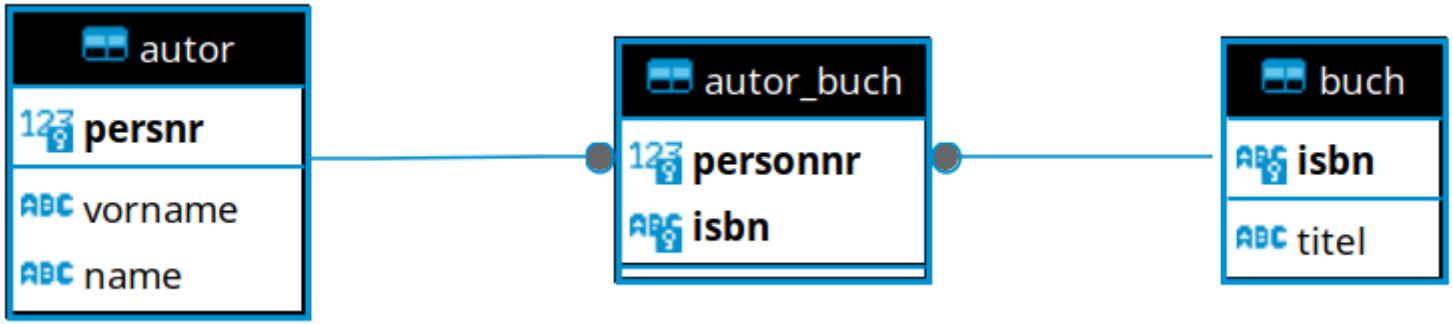
Kommentare
Standard und MySQL spezifische

```
sql> -- Dies ist ein Standard SQL Kommentar


sql> # Dies ist ein MySQL "Python-like" Kommentar


sql> /* Dies ist ein MySQL "Java-like" Kommentar
          Der Vorteil: Anwendung auf mehreren Zeilen
     */
```

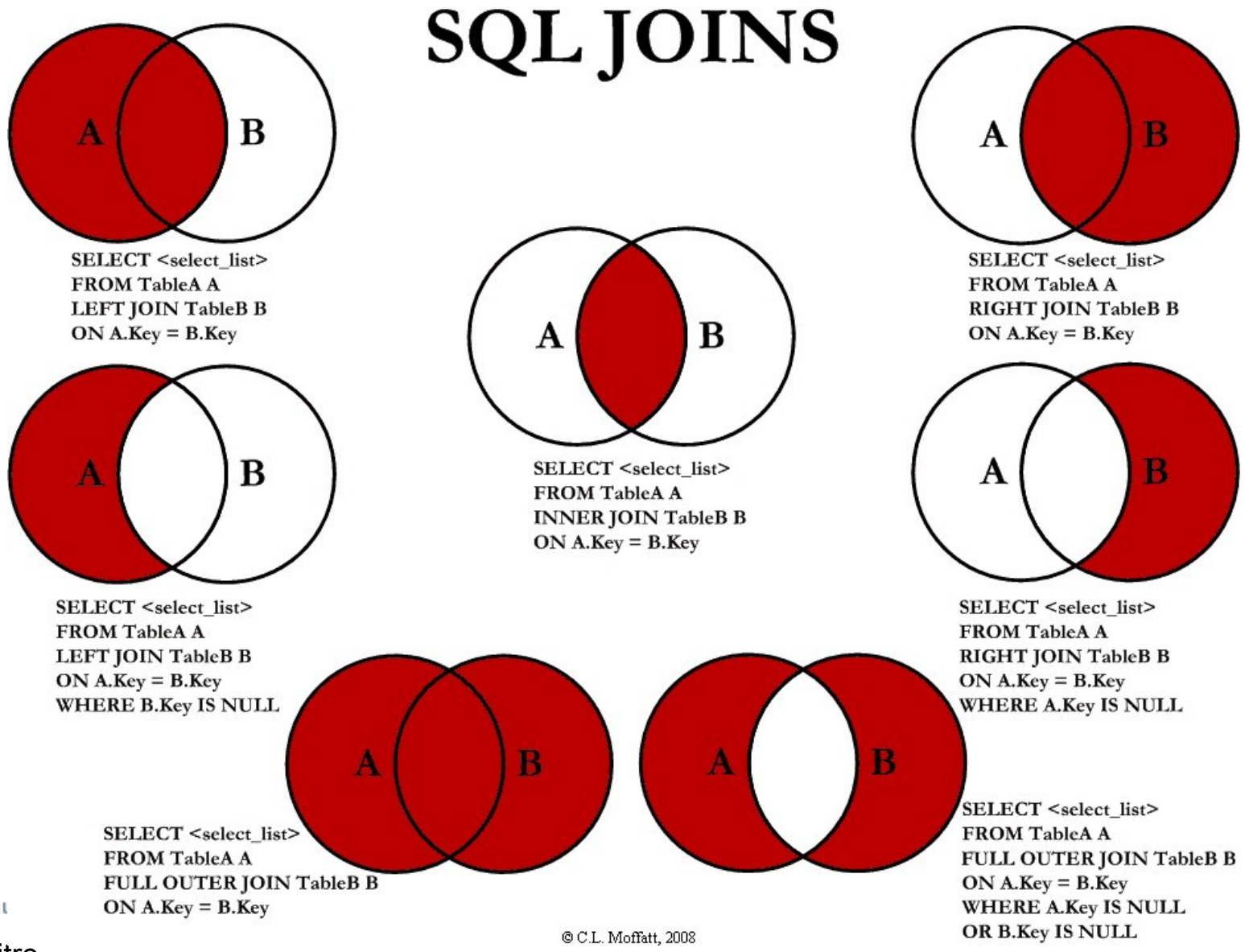
Mehrere Tabellen abfragen (Joins)
Lösung "von Hand"



Welche Autoren welche Bücher geschrieben haben?

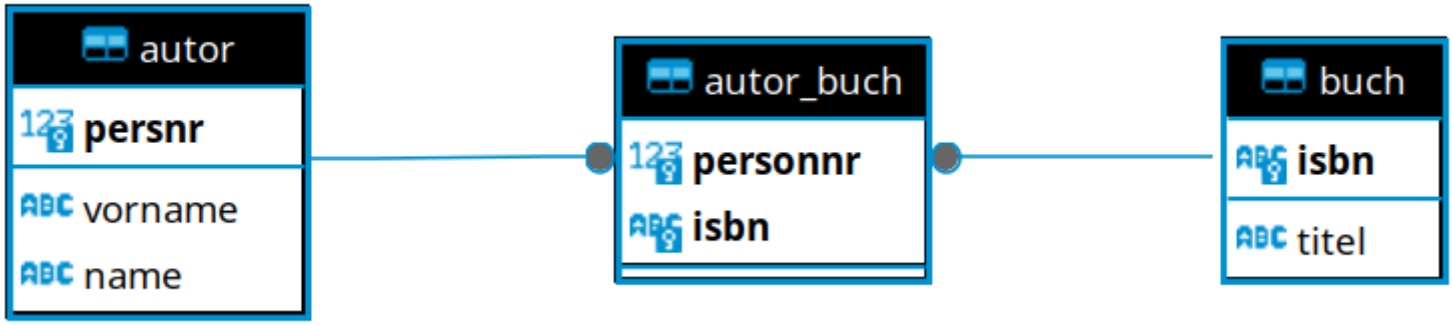
```
sql> select a.name, b.titel
     from autor a,
          autor_buch ab,
          buch b
     where a.persnr = ab.personnr
     and b.isbn = ab.isbn;
+--------+------------------+
| name   | titel            |
+--------+------------------+
| Kemper | Datenbanksysteme |
| Kofler | VisualBasic 2008 |
| Kofler | Mathematica      |
| Kofler | MySQL 5          |
| Kofler | Linux            |
+--------+------------------+
```

Mehrere Tabellen abfragen (Joins)
Join-Typen



Quelle: http://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

Mehrere Tabellen abfragen (Joins)
Lösung "mit INNER JOIN"

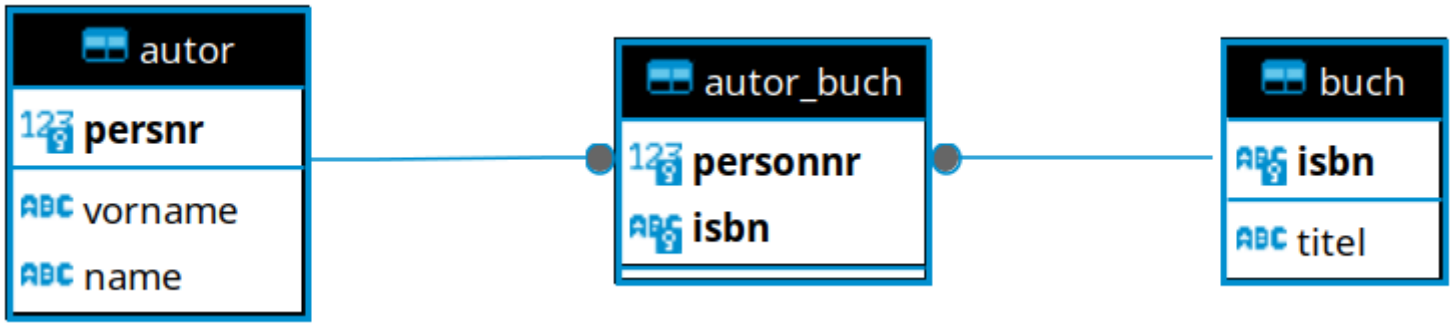



```
sql> select a.name, b.titel from autor a
     inner join autor_buch ab on a.persnr = ab.personnr
     inner join buch b on b.isbn = ab.isbn;
+--------+------------------+
| name   | titel            |
+--------+------------------+
| Kemper | Datenbanksysteme |
| Kofler | VisualBasic 2008 |
| Kofler | Mathematica      |
| Kofler | MySQL 5          |
| Kofler | Linux            |
+--------+------------------+
```

"INNER JOIN" ist der Default-JOIN

Welche Autoren welche Bücher geschrieben haben?

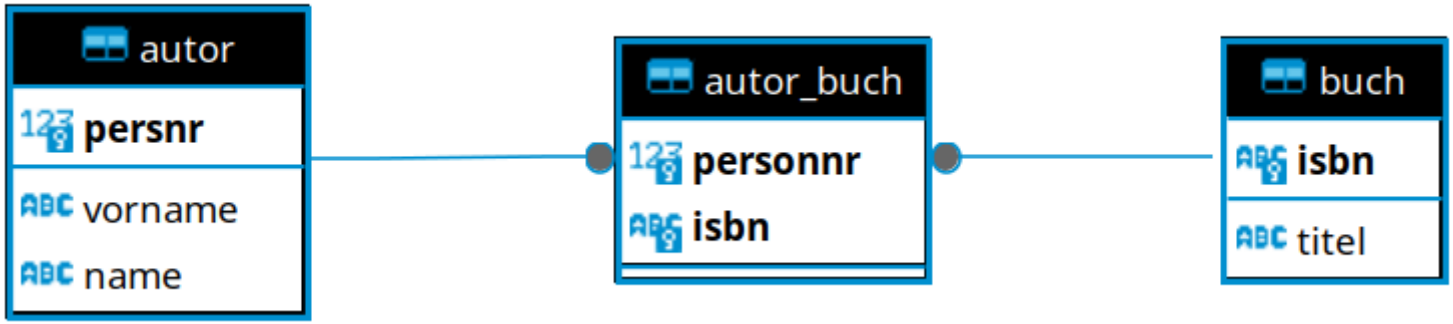
Mehrere Tabellen abfragen (Joins)
Left Join




```
sql> select a.name, b.titel from autor a
     left join autor_buch ab on a.persnr = ab.personnr
     left join buch b on ab.isbn = b.isbn;
+--------+------------------+
| name   | titel            |
+--------+------------------+
| Kemper | Datenbanksysteme |
| Kofler | VisualBasic 2008 |
| Kofler | Mathematica      |
| Kofler | MySQL 5          |
| Kofler | Linux            |
| Maitre | NULL             |
| Wall   | NULL             |
+--------+------------------+
```

Frage: **Alle** Autoren und ihre Bücher.

Mehrere Tabellen abfragen (Joins)
Left Join praktische Anwendung




```
sql> select a.name, b.titel from autor a
     left join autor_buch ab on a.persnr = ab.personnr
     left join buch b using (isbn)
     where b.titel is null;
```

```
+--------+
| name   |
+--------+
| maitre |
| wall   |
+--------+
```

Antwort auf die Frage: Welche Autoren kein Buch geschrieben haben?

Übung 2
Siehe letztes Slide

Bitte Übung 2 machen.

Mit Strings arbeiten
MySQL Datentypen

- **CHAR(n)**
  - Festlänge, MySQL bis 255 Zeichen, Oracle 2000
- **varchar(n)**
  - Variable Länge, MySQL bis 65K Zeichen, Oracle 4K
- **text(n)**
  - Für Dokumente, MySQL bis 4 GB, Oracle 128 TB
  - (CLOB in Oracle)

Für alle: (n) == maximale Länge
insert(Data > n) => if "SQL_Strict" then error else warning + truncate

Mit Datum/Zeit arbeiten
MySQL Datentypen

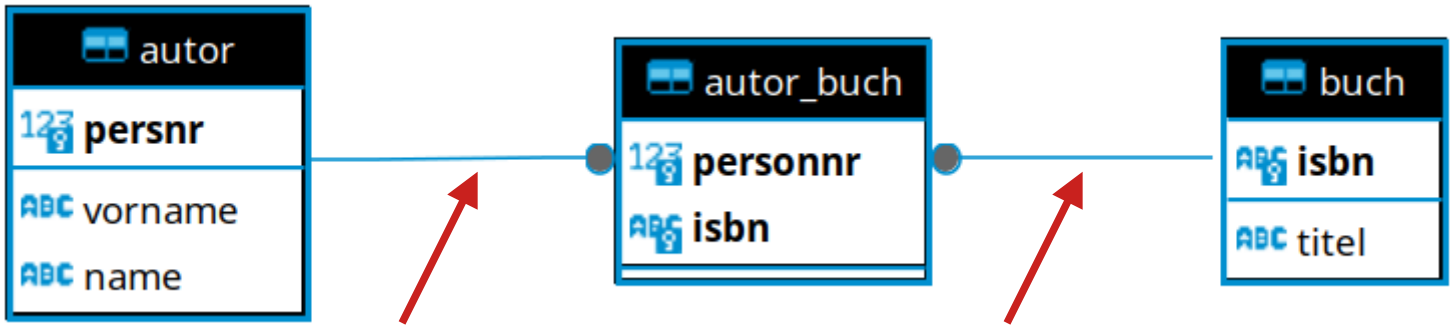
- **Date (YYYY-MM-DD)**
  (Bereich: '1000-01-01' bis '9999-12-31')
  3 Bytes
- **Datetime (YYYY-MM-DD HH:MI:SS)**
  (Bereich: '1000-01-01 00:00:00' bis '9999-12-31 23:59:59')
  8 Bytes
- **Timestamp (YYYY-MM-DD HH:MI:SS)**
  (Bereich: '1970-01-01 00:00:01' UTC bis '2038-01-09 03:14:07' UTC)
  4 Bytes, #Sek. seit 1970-01-01 00:00:00
- **Time (HH:MI:SS)**
  3 Bytes
- **Mögliche Eingabe-Formate:**
  - 2009/07/17 14:29:00
  - 2009,07,17,14,29,00
  - 20090717142900

Mit Datum/Zeit arbeiten
SQLite Beispiele

```
mysql> select strftime('%d %m %Y', '2025-04-07 10:49:17');
+-----------------------------------------+
| strftime('%d %m %Y', '2025-04-07 10:49:17')|
+-----------------------------------------+
| 07 04 2025                              |
+-----------------------------------------+

sql> select date('now'), time('now'), datetime('now');
+-------------+-------------+-------------------+
| date('now') | time('now') | datetime('now')   |
+-------------+-------------+-------------------+
| 2025-04-07  | 08:43:57    | 2025-04-07 08:43:57|
+-------------+-------------+-------------------+
```

Referenzielle Integrität
Nach Leichen mit Unterabfrage suchen




```
-- Gibt es Autoren in autor_buch, die aber in autor
-- gelöscht wurden?

sql> select * from autor_buch ab
     where ab.personnr not in
        (select a.persnr from autor a);

Empty set
```

Sog. "Integritätsregel" stellen sicher, dass es nie passiert, dass Fremdschlüssel **nicht mehr existierende** Primärschlüssel referenzieren.

Daten gruppieren

```
sql> select * from autor_buch ab;
+----------+------+
| personnr | isbn |
+----------+------+
|       34 | 111  |
|       12 | 123  |
|       34 | 222  |
|       34 | 456  |
|       34 | 789  |
+----------+------+

sql> select personnr from autor_buch ab group by ab.personnr;
+----------+
| personnr |
+----------+
|       12 |
|       34 |
+----------+

-- Welche Personen haben mehr als 2 Bücher geschrieben?
sql> select ab.personnr, count(ab.personnr) from autor_buch ab
       group by ab.personnr having count(ab.personnr) > 2;
+----------+-----------------+
| personnr | count(personnr) |
+----------+-----------------+
|       34 |               4 |
+----------+-----------------+
```

Daten gruppieren
Nach Duplikaten suchen

```
sql> select ab.personnr, count(ab.personnr)
     from autor_buch ab
     group by personnr
     having count(ab.personnr) > 1;

+----------+--------------------+
| personnr | count(ab.personnr) |
+----------+--------------------+
|       34|                   4|
+----------+--------------------+
```

Antwort auf die Frage:
Haben wir in
autor_buch doppelte
Autoren?

Daten aggregieren
Allgemein

- Funktionen: max(), min(), avg(), sum(), count()

```
sql> select count(personnr) from autor_buch ab;
+------------------+
| count(personnr)  |
+------------------+
|                5 |
+------------------+

sql> select count(distinct personnr) from autor_buch ab;
+--------------------------+
| count(distinct personnr) |
+--------------------------+
|                        2 |
+--------------------------+
```

Daten gruppieren UND aggregieren

```
sql> select personNr, count(isbn) from autor_buch ab
     group by personnr;
+----------+-------------+
| personnr | count(isbn) |
+----------+-------------+
|       12 |           1 |
|       34 |           4 |
+----------+-------------+

sql> select ab.personnr, a.name, count(isbn)
     from autor_buch ab, autor a where ab.personnr=a.persnr
     group by personnr, name;
+----------+--------+-------------+
| personnr | name   | count(isbn) |
+----------+--------+-------------+
|       12 | Kemper |           1 |
|       34 | Kofler |           4 |
+----------+--------+-------------+
```

Übungen

Meine "gmcd.sqlite" CD-Datenbank steht Ihnen auf Moodle zur Verfügung.
Laden Sie diese herunter und speichern Sie diese ins Ihr Kursverzeichnis.
Erstellen Sie dann im DBeaver eine neue Verbindung darauf, ähnlich wie für *autorbuch* DB.
Starten Sie mit SQL-Editor ein neues Skript und testen Sie die Verbindung mit dem Befehl:

```
select * from cd;
```

1) Wann wurde die CD "Yellow Submarine" publiziert?

2) Alle Titel und deren Dauer auflisten, die sich auf der CD "The Koeln Concert" befinden (ohne und mit JOIN).

3) Wie heissen die Musiker, die das Instrument "Gitarre" spielen?

4) Wie heissen alle Musiker, die auf der CD "Yellow Submarine" spielen? (mit JOIN, und die Spalte soll "Musiker" heissen).

5) Fakultative Aufgabe: Welches sind die Stücke (Titel auflisten), die länger dauern als die Durchschnittsdauer (Funktion "AVG" und WITH-Statement verwenden)?