

Tag 3

Inhaltsverzeichnis

SQL Survival Kit... (lesend)

- SQL Select-Anweisung (Grundbild)
- Daten filtern
- Kommentare in SQL
- Komplexe Abfragen strukturieren
- Mehrere Tabellen abfragen (Joins)
- Arbeiten mit Strings und Datum/Zeit
- Referenzielle Integrität
- Daten gruppieren und aggregieren
- Übungen
- BYOQ

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)

```
mysql> select * from Autor where PersNr='12';
```

PersNr	Vorname	Name
12	Alfons	Kemper

```
mysql> select * from Autor a where a.PersNr='12';
```

PersNr	Vorname	Name
12	Alfons	Kemper

```
mysql> select PersNr as keyPers from Autor a where a.PersNr='12';
```

keyPers
12



Spalten-Alias

Daten filtern

Gleichheitsbedingung (2)

```
mysql> select distinct Vorname from Autor;
```

```
+-----+  
| Vorname |  
+-----+  
| Alfons  |  
| Michael |  
| Gilles  |  
+-----+
```

Entferne
Duplicate

```
mysql> select Vorname, Name from Autor  
order by Vorname, Name asc;
```

Sortierung

```
+-----+-----+  
| Vorname | Name  |  
+-----+-----+  
| Alfons  | Kemper |  
| Gilles  | Maitre |  
| Michael | Kofler |  
+-----+-----+
```

Daten filtern

Wertebereichsbedingung

```
mysql> select * from Autor
      where PersNr >= 11 AND PersNr <= 35;
```

PersNr	Vorname	Name
12	Alfons	Kemper
34	Michael	Kofler

```
mysql> select * from Autor where PersNr
      between 11 and 35;
```

PersNr	Vorname	Name
12	Alfons	Kemper
34	Michael	Kofler

Daten filtern

Wildcards

"_": genau ein Zeichen

"%": beliebig viele Zeichen

```
mysql> select * from Autor where Name = "K%";  
Empty set (0.00 sec)
```

```
mysql> select * from Autor where Name like "K%";  
+-----+-----+-----+  
| PersNr | Vorname | Name   |  
+-----+-----+-----+  
|      12 | Alfons  | Kemper |  
|      34 | Michael | Kofler |  
+-----+-----+-----+
```

Daten filtern

Null ist nicht null

```
mysql> alter table Autor add column Bemerkung varchar(20) after name;
```

```
mysql> select * from Autor where Bemerkung = null;  
Empty set (0.00 sec)
```

```
mysql> select * from Autor where Bemerkung is null;
```

```
+-----+-----+-----+-----+  
| PersNr | Vorname | Name   | Bemerkung |  
+-----+-----+-----+-----+  
|      12 | Alfons  | Kemper | NULL      |  
|      34 | Michael | Kofler | NULL      |  
|      56 | Gilles  | Maitre | NULL      |  
+-----+-----+-----+-----+
```

```
mysql> alter table Autor drop column Bemerkung;
```

Kommentare

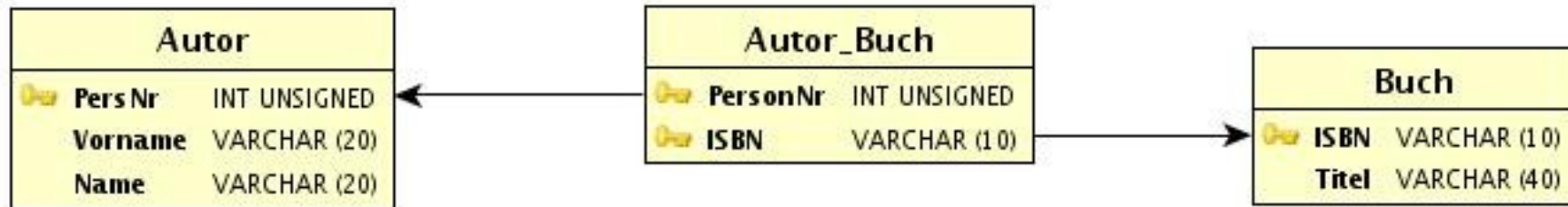
Standard und MySQL spezifische

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

```
mysql> # Dies ist ein MySQL "Python-like" Kommentar
```

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


Mehrere Tabellen abfragen (Joins) Lösung "von Hand"



```
mysql> select a.Name, b.Titel
        from Autor a, Autor_Buch ab, Buch b
        where a.PersNr=ab.PersonNr and b.ISBN=ab.ISBN;
```

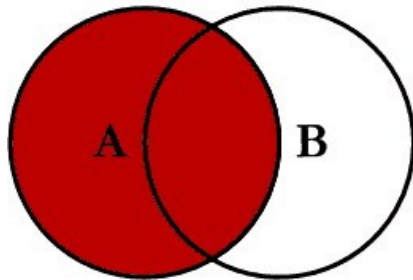
```
+-----+-----+
| Name   | Titel                               |
+-----+-----+
| Kemper | Datenbanksysteme                   |
| Kofler | VisualBasic 2008                   |
| Kofler | Mathematica                         |
| Kofler | MySQL 5                             |
| Kofler | Linux                               |
+-----+-----+
```

**Frage: welche
Autoren haben
welche Bücher
geschrieben?**

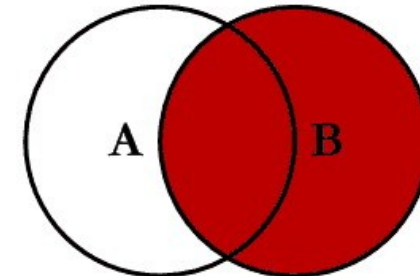
Mehrere Tabellen abfragen (Joins)

Join-Typen

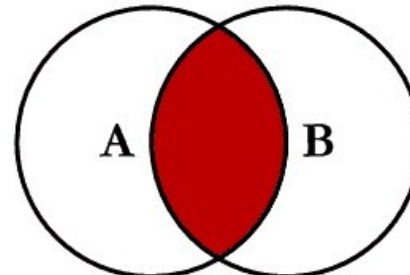
SQL JOINS



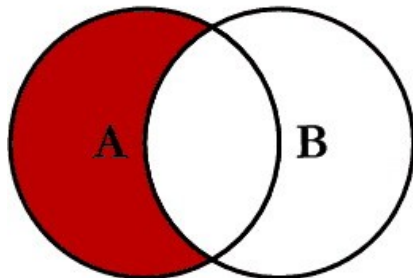
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



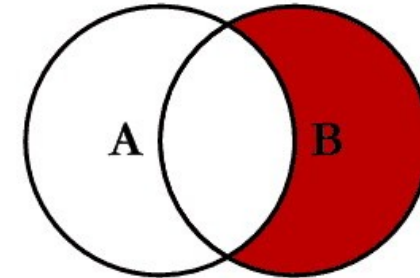
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



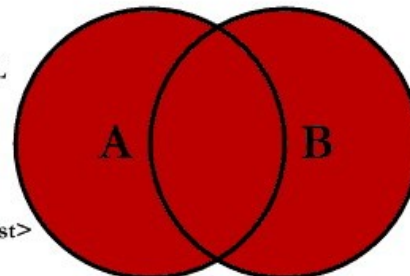
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



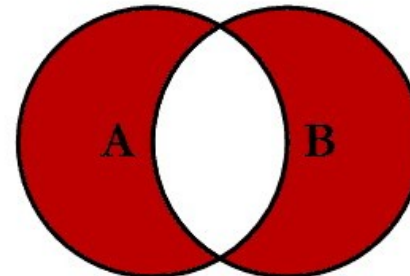
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

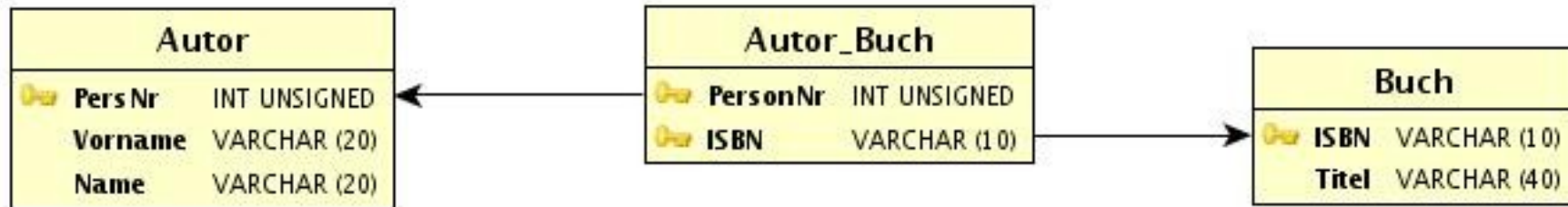


```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Mehrere Tabellen abfragen (Joins) Lösung "mit INNER JOIN" (1)



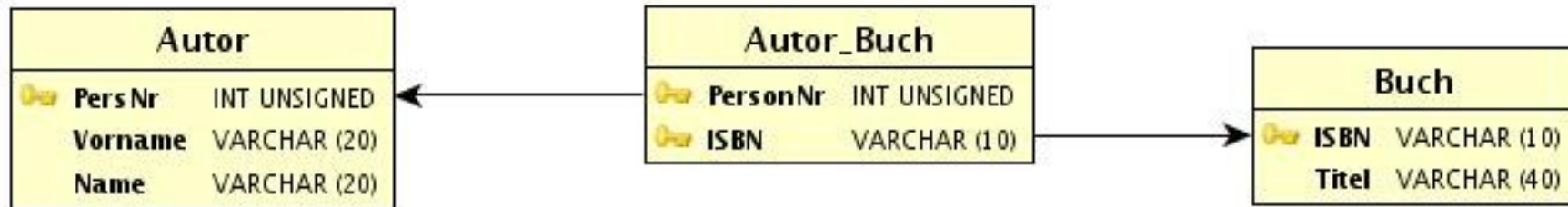
```
mysql> select a.Name, b.Titel from Autor a
       inner join Autor_Buch ab on a.PersNr=ab.PersonNr
       inner join Buch b on b.ISBN=ab.ISBN;
```

```
+-----+-----+
| Name   | Titel |
+-----+-----+
| Kemper | Datenbanksysteme |
| Kofler | VisualBasic 2008 |
| Kofler | Mathematica      |
| Kofler | MySQL 5          |
| Kofler | Linux            |
+-----+-----+
```

"INNER JOIN"
ist der Default-
JOIN

Mehrere Tabellen abfragen (Joins)

Left Join



```
mysql> select a.Name, b.Titel from Autor a
      left join Autor_Buch ab on a.PersNr = ab.PersonNr
      left join Buch b on b.ISBN = ab.ISBN;
```

```
+-----+-----+
| Name   | Titel                |
+-----+-----+
| Kemper | Datenbanksysteme    |
| Kofler | VisualBasic 2008    |
| Kofler | Mathematica          |
| Kofler | MySQL 5              |
| Kofler | Linux                |
| Maitre | NULL                 |
| Wall   | NULL                 |
+-----+-----+
```

Arbeiten mit Strings

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

Arbeiten mit Datum/Zeit 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

Arbeiten mit Datum/Zeit

Beispiele

```
mysql> select str_to_date('17 July 09', '%d %M %y');
+-----+
| str_to_date('17 July 09', '%d %M %y') |
+-----+
| 2009-07-17 |
+-----+
```

```
mysql> select current_date(), current_time(), current_timestamp();
+-----+-----+-----+
| current_date() | current_time() | current_timestamp() |
+-----+-----+-----+
| 2009-07-17 | 14:34:38 | 2009-07-17 14:34:38 |
+-----+-----+-----+
```

```
mysql> select now();
+-----+
| now() |
+-----+
| 2009-07-17 14:35:19 |
+-----+
```

```
mysql> select date_add(now(), interval 3 day);
+-----+
| date_add(now(), interval 3 day) |
+-----+
| 2009-07-20 14:38:20 |
+-----+
```

MySQL

build-in Funktionen

Chapter 11. Functions and Operators

[Table of Contents](#) [+/-]

[11.1. Operator and Function Reference](#)

[11.2. Operators](#) [+/-]

[11.3. Control Flow Functions](#)

[11.4. String Functions](#) [+/-]

[11.5. Numeric Functions](#) [+/-]

[11.6. Date and Time Functions](#)

[11.7. What Calendar Is Used By MySQL?](#)

[11.8. Full-Text Search Functions](#) [+/-]

[11.9. Cast Functions and Operators](#)

[11.10. Other Functions](#) [+/-]

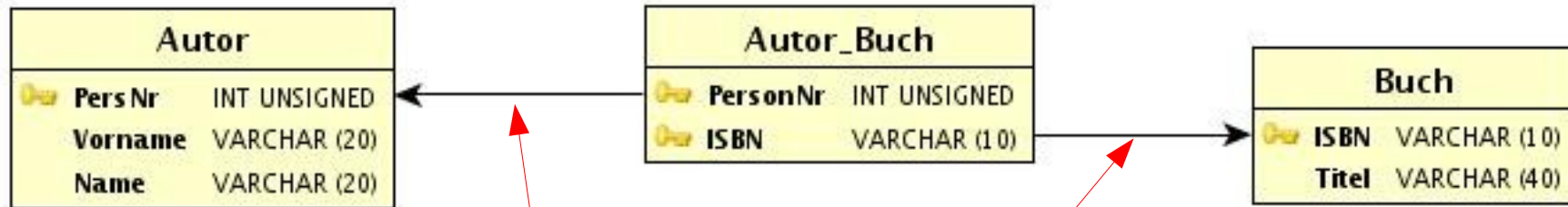
[11.11. Functions and Modifiers for Use with `GROUP BY` Clauses](#) [+/-]

Referenzielle Integrität

Problematik der Daten-Integrität

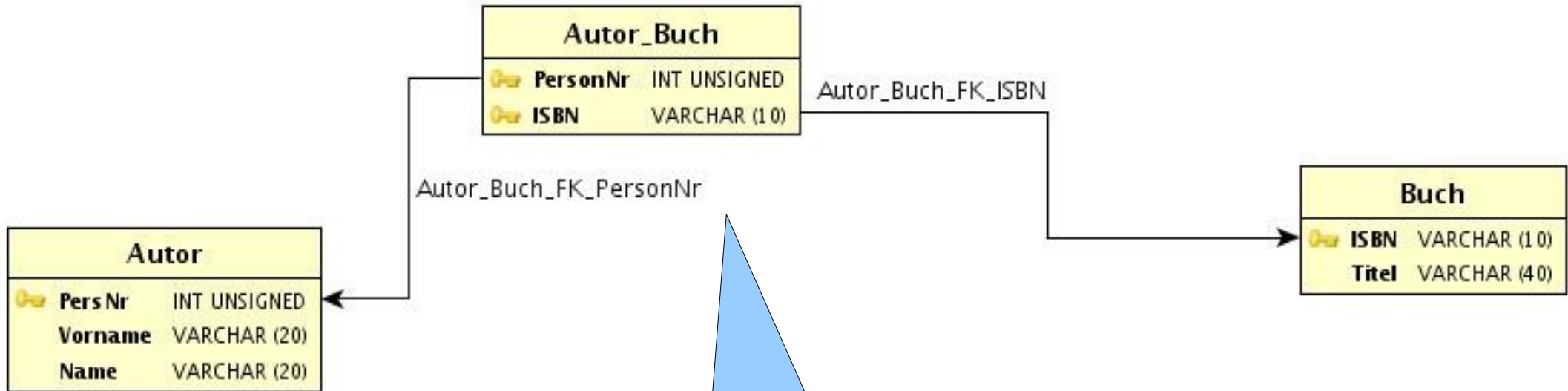
- Integrität der Primärschlüssel
 - Unique, not null
- Integrität der Fremdschlüssel
 - Referenzierte Primärschlüssel existieren
- Integrität der anderen Daten (nach Typ)
 - Wertbereich
 - Nicht NULL

Referenzielle Integrität Kohärenz der Fremdschlüssel



- **Kohärenz der Fremdschlüssel**
 - "Dangling Pointer" Problem vermeiden
- => **Integritätsregel** setzen

Referenzielle Integrität FK Namenskonvention



**FK Namenskonvention:
'fromTable'_FK_'fromField'**

Referenzielle Integrität Regel anzeigen

- **In MySQL Workbench**
 - "Table Information" anzeigen / Tab "Foreign Key"
- **In MySQL Command Line Interpreter**
 - `show create table <table name>`

Referenzielle Integrität

Leichen mit Unterabfrage suchen

```
-- Gibt es Autoren in Autor_Buch, die aber in Autor  
-- gelöscht wurden?
```

```
mysql> select * from Autor_Buch where PersonNr not in  
        (select distinct PersNr from Autor);
```

Empty set

Daten gruppieren

```
mysql> select * from Autor_Buch;
```

```
+-----+-----+
| PersonNr | ISBN |
+-----+-----+
|      34 | 111  |
|      12 | 123  |
|      34 | 222  |
|      34 | 456  |
|      34 | 789  |
+-----+-----+
```

```
mysql> select PersonNr from Autor_Buch group by PersonNr;
```

```
+-----+
| PersonNr |
+-----+
|      12 |
|      34 |
+-----+
```

-- Welche Personen haben mehr als 2 Bücher geschrieben?

```
mysql> select PersonNr, count(PersonNr) from Autor_Buch group by
PersonNr having count(PersonNr) > 2;
```

```
+-----+-----+
| PersonNr | count(PersonNr) |
+-----+-----+
|      34 |          4      |
+-----+-----+
```

Daten gruppieren

Referenzielle Integrität, PK-Prüfung

-- Frage: Ist mein Primary Key wirklich unique?

```
mysql> select PersNr, count(PersNr) from Autor  
      group by PersNr having count(PersNr) > 1;
```

Empty set

-- Autor_Buch überprüfen. **PersonNr + ISBN sind PK.**

-- Ein Autor kann nicht zweimal das gleiche Buch geschrieben haben!

```
mysql> select PersonNr, ISBN, count(PersonNr) from Autor_Buch  
      group by PersonNr, ISBN having count(PersonNr) > 1;
```

Empty set

Daten aggregieren

Allgemein

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

```
mysql> select count(PersonNr) from Autor_Buch;
```

```
+-----+  
| count(PersonNr) |  
+-----+  
|                5 |  
+-----+
```

```
mysql> select count(distinct PersonNr) from Autor_Buch;
```

```
+-----+  
| count(distinct PersonNr) |  
+-----+  
|                        2 |  
+-----+
```


Daten aggregieren

count(*) != count('Spalte')

- Achtung:
 - count(*) => Anzahl *aller Zeilen der Tabelle*
 - count('Spalte') => Anzahl aller *nicht-NULL Zeilen dieser Spalte*

-- Dies könnte unterschiedliche Resultate liefern!

```
SELECT SUM(column) / COUNT(*) FROM table
```

```
SELECT SUM(column) / COUNT(column) FROM table
```

Daten gruppieren UND aggregieren

```
mysql> select PersonNr, count(ISBN) from Autor_Buch
        group by PersonNr;
```

PersonNr	count(ISBN)
12	1
34	4

```
mysql> 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 CD-Datenbank steht Ihnen zur Verfügung.

In MySQL Workbench oder auf MySQL Kommandointerpreter, die Befehle ausführen:

```
use GMCD;  
show tables;  
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) Welches sind die Stücke (Titel auflisten), die länger dauern als die Durchschnittsdauer (Funktion "AVG" verwenden)?