

Tag 3

Inhaltsverzeichnis

- SQL Select-Anweisung (Grundbild)
- Daten filtern
 - Gleichheitsbedingung
 - Wertebereichsbedingung
 - Mitgliedschaftbedingung
 - Wildcards
 - NULL ist nicht null
- Mehrere Tabellen abfragen (Joins)
- Arbeiten mit Strings und Datum/Zeit
- Referenzielle Integrität
- Daten gruppieren und aggregieren
- Übungen

SQL Select-Anweisung (Grundbild)

```
SSELECT [DISTINCT] Auswahlliste [AS Neuer Name]+  
FFROM Quelle  
[WWHERE 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 |
+-----+-----+-----+
```

Die SQL-Schlüsselwörter schreibe ich klein

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

```
+-----+-----+-----+
| PersNr | Vorname | Name  |
+-----+-----+-----+
|      12 | Alfons  | Kemper |
+-----+-----+-----+
```

Verwendung einer Tabellen-Variable

```
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

Mitgliedschaftbedingung

```
mysql> select * from Autor where PersNr in ('11', '35');  
Empty set (0.00 sec)
```

```
mysql> select * from Autor where PersNr in ('12', '34');  
+-----+-----+-----+  
| 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 |  
+-----+-----+-----+
```

```
mysql> select * from Autor where Vorname regexp "^[AM]";
```

```
+-----+-----+-----+  
| PersNr | Vorname | Name  |  
+-----+-----+-----+  
|      12 | Alfons  | Kemper |  
|      34 | Michael | Kofler |  
+-----+-----+-----+
```

fängt mit A
oder M an

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;
```


Komplexe Abfragen strukturieren

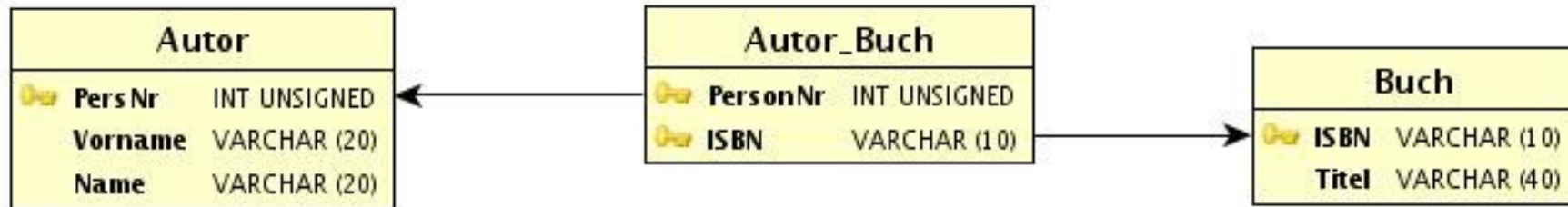
Der WITH-Statement

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

-- 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';
```

Mehrere Tabellen abfragen (Joins) Erster Versuch



```
mysql> select Autor.Name, Buch.Titel from Autor join Buch ;
```

```

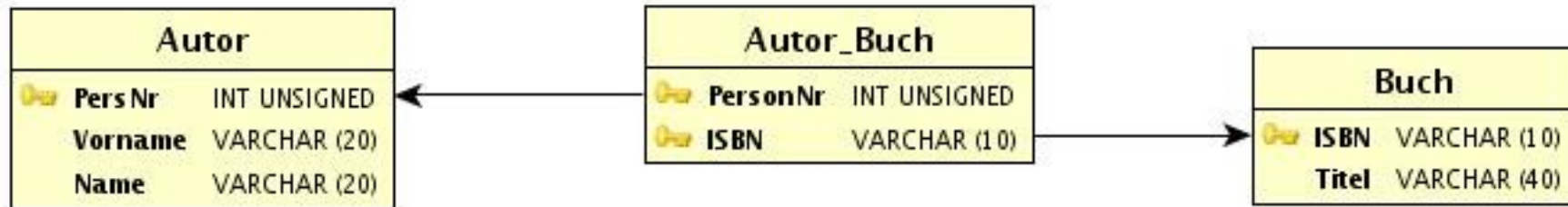
+-----+-----+
| Name   | Titel                               |
+-----+-----+
| Kemper | VisualBasic 2008                   |
| Kofler | VisualBasic 2008                   |
| Maitre | VisualBasic 2008                   |
| Wall   | VisualBasic 2008                   |
| Kemper | Datenbanksysteme                   |
| Kofler | Datenbanksysteme                   |
| Maitre | Datenbanksysteme                   |
| Wall   | Datenbanksysteme                   |

```

...

Auf die Frage: welche Autoren haben welche Bücher geschrieben?
Das kartesische Produkt gibt die Antwort nicht!

Mehrere Tabellen abfragen (Joins) Lösung "von Hand"



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

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

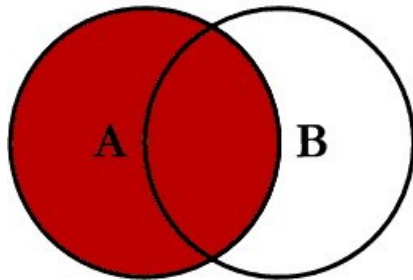
Frage: welche Autoren
haben welche Bücher
geschrieben?

*Kartesisches Produkt
PLUS Selektion*

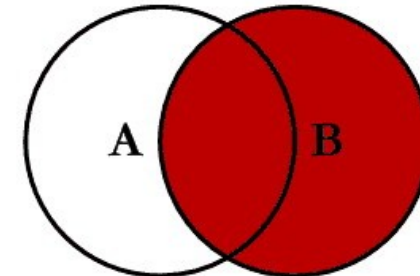
Mehrere Tabellen abfragen (Joins)

Join-Typen

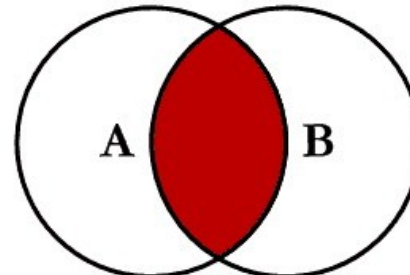
SQL JOINS



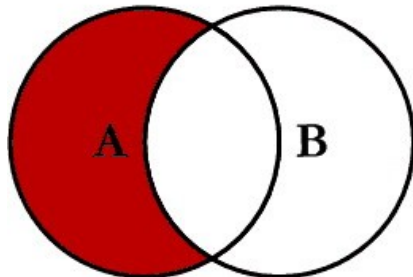
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



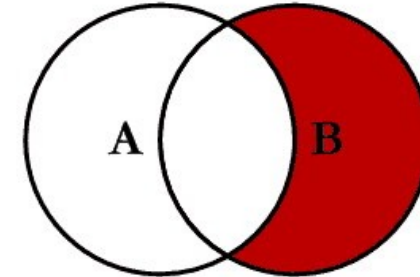
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



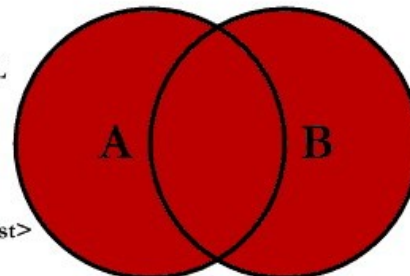
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



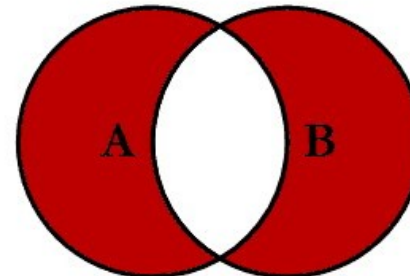
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



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



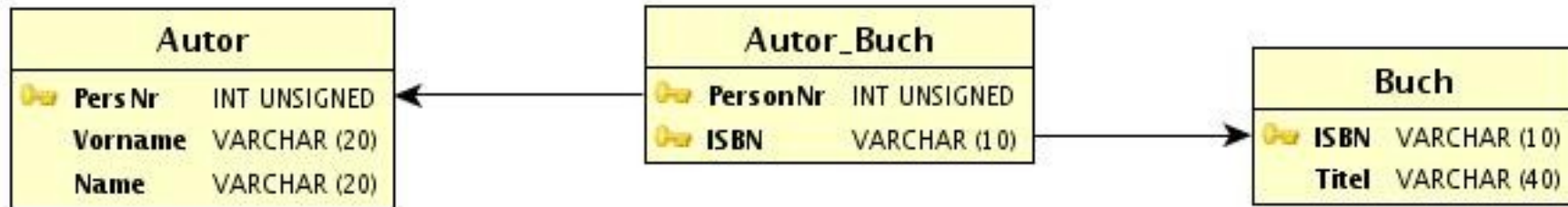
```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
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)



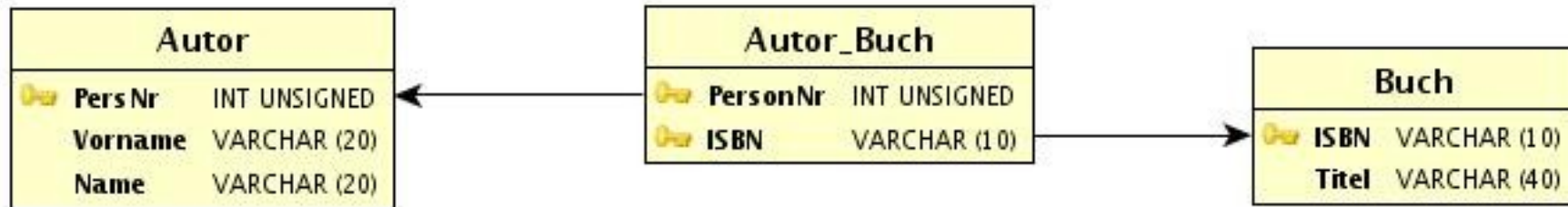
```
mysql> select a.Name, b.Titel from Autor a
        inner join Autor_Buch ab on a.PersNr=ab.PersonNr
        inner join Buch b on b.ISBN=ab.ISBN;
```

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

"INNER JOIN"
ist der Default-
JOIN

Mehrere Tabellen abfragen (Joins)

Lösung "mit INNER JOIN" (2)



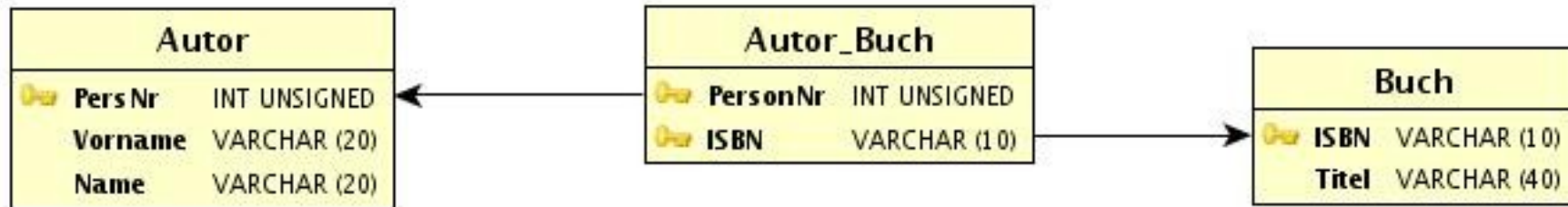
```
mysql> select a.Name, b.Titel from Autor a
        inner join Autor_Buch ab on a.PersNr=ab.PersonNr
        inner join Buch b using (ISBN);
```

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

Nur möglich
wenn beide
Tabellen
"ISBN" als
Attribut
haben

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) => ERROR 1406 (22001): Data too long for column '...'

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](#) [+/-]

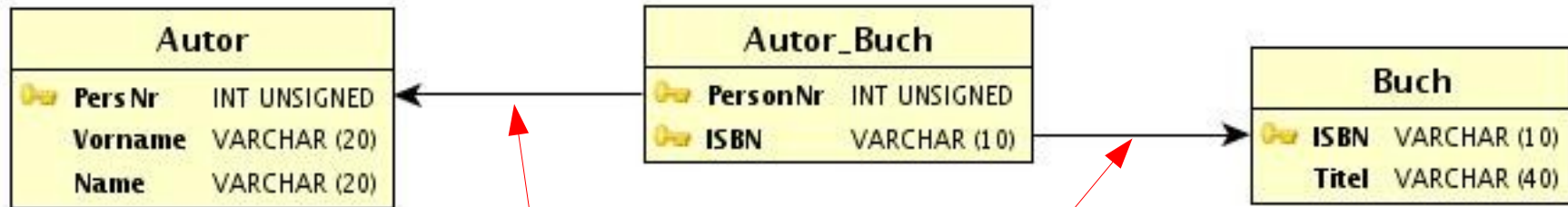
Referenzielle Integrität

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

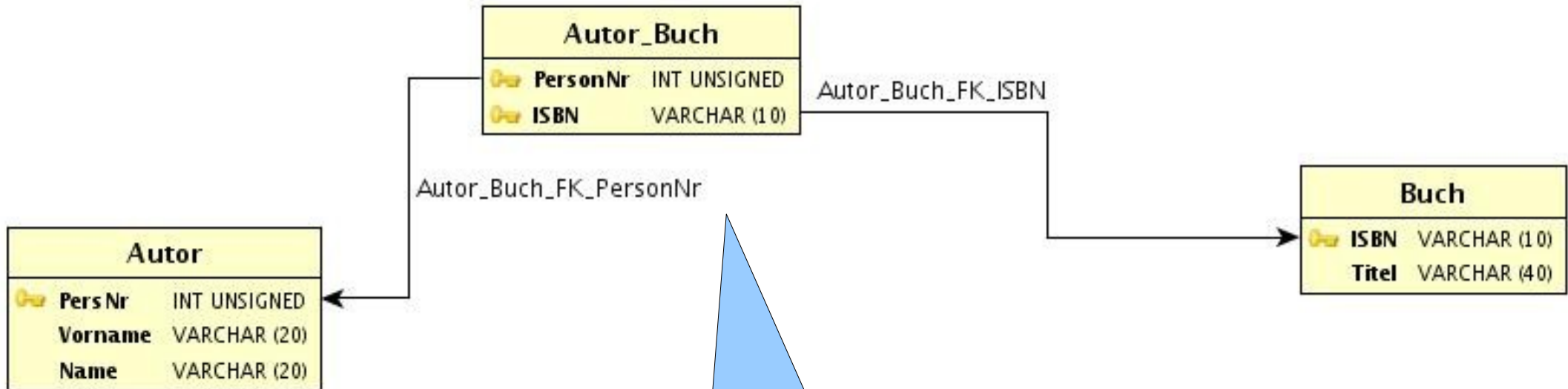
Konsistenz der Fremdschlüssel



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

Referenzielle Integrität

FK Namenskonvention



FK Namenskonvention:
'fromTable'_FK_'fromField'

Referenzielle Integrität Regel erstellen

- In MySQL, nur für **InnoDB**-Tabellen

Namenskonvention:
'fromTable'_FK_'fromField'

- **Beispiel:**

```
ALTER TABLE Autor_Buch
  ADD CONSTRAINT Autor_Buch_FK_PersonNr
  FOREIGN KEY (PersonNr) REFERENCES Autor (`PersNr`);
```

=> In "Autor", kein Löschen von einem **referenzierten** Autor

- **Syntax:**

```
FOREIGN KEY [Name] (Kolonne 1) REFERENCES Tabelle2 (Kolonne
2)
  [ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT}]
  [ON UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT}]
```

- **"RESTRICT" ist der Default**

Referenzielle Integrität Regel anzeigen

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

```
| Autor_Buch | CREATE TABLE `Autor_Buch` (  
  `PersonNr` int(10) unsigned NOT NULL,  
  `ISBN` varchar(10) COLLATE utf8_unicode_ci NOT NULL,  
  PRIMARY KEY (`PersonNr`,`ISBN`),  
  KEY `Autor_Buch_FK_ISBN` (`ISBN`),  
  CONSTRAINT `Autor_Buch_FK_ISBN` FOREIGN KEY (`ISBN`) REFERENCES `Buch`  
  (`ISBN`),  
  CONSTRAINT `Autor_Buch_FK_PersonNr` FOREIGN KEY (`PersonNr`) REFERENCES  
  `Autor` (`PersNr`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci |
```


Referenzielle Integrität

Regel prüfen

```
mysql> delete from Autor where PersNr='12';
```

```
ERROR 1451 (23000):
```

```
Cannot delete or update a parent row: a foreign key constraint fails  
(`AutorBuch/Autor_Buch`,  
CONSTRAINT `Autor_Buch_FK_PersonNr`  
FOREIGN KEY (`PersonNr`) REFERENCES `Autor` (`PersNr`))
```

Referenzielle Integrität Prinzipien

- Erzeugen, ändern, löschen: ALTER TABLE...
- Löschen: ALTER TABLE 'table' DROP FOREIGN KEY 'name';
- Temporär Regel ausschalten:
 - SET foreign_key_checks = {0, 1};
- Regel muss von Anfang an stimmen!
 - Problem mit existierenden Daten...
- Wieviel Integrität ist erwünscht?
 - Kompromiss *Integrität / Flexibilität* suchen

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

```
-- Prüfung mit Autor_Buch... Keine dumme Idee, da ein Autor  
-- nicht zweimal das gleiche Buch geschrieben haben konnte!
```

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

Empty set

Daten aggregieren

- 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-DB steht Ihnen zur Verfügung:

Die Datei aus <https://web.mtg1.bfh.science/SD-RDB/gmcd.sql> herunterladen, MySQL Workbench starten, MyLocalhostConnection eröffnen, unter dem Menüpunkt File / OpenSQLScript... die Datei laden und ausführen.

- 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)?
- 6) Gleiche Frage wie bei 5) aber mit dem WITH-Statement.
- 7) Setzen Sie referentielle Integritätsregel in die CD DB (mit MySQL CommandLine und/oder Workbench). Und prüfen Sie sie.