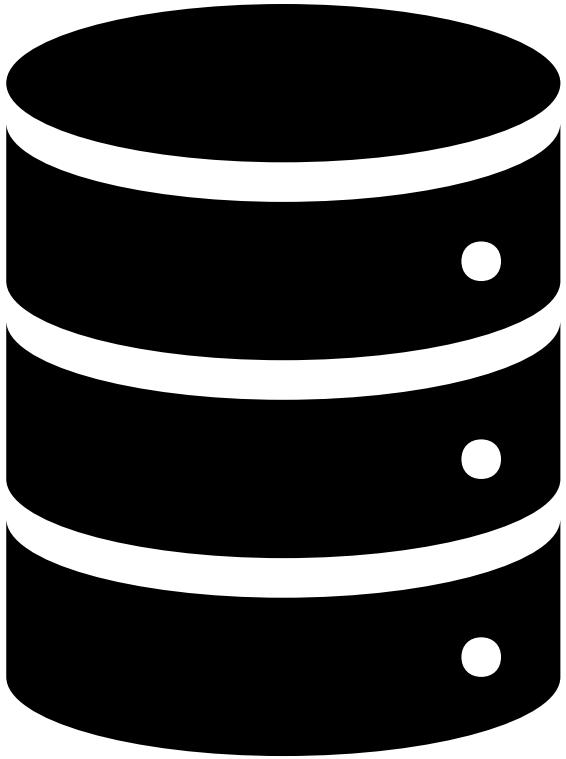


Einführung in Datenbanken

März 2025

Plan für die Woche

Montag	Dienstag	Mittwoch	Donnerstag	Freitag
• Was sind Datenbanken? • Normalisierung	• Was ist SQL? • DDL und DML	• DDL und DML-Fortsetzung	• DDL und DML-Fortsetzung	• JOIN

Plan für heute

- Was sind Aliases?
- Was sind JOINS?
- Welche Arten von JOINS gibt es?
- Wie erstellt man einen JOIN?

SMART INDUSTRY CAMPUS

Aliases

Aliases

- wird benutzt, um Tabellen oder Spalten einen temporären Namen zu geben
  - dadurch sind sie oft lesbarer
- Syntax:

```
SELECT column_name AS alias_name
FROM table_name;
```

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

SMART INDUSTRY CAMPUS

Aliases

- wird benutzt, um Tabellen oder Spalten einen temporären Namen zu geben
  - dadurch sind sie oft lesbarer
- Beispiel:

```
SELECT firstname AS Vorname, lastname AS Nachname FROM Patient;
```

Vorname	Nachname
Max	Müller
Anna	Schmidt
Lukas	Meier
Sophie	Fischer
Felix	Becker
Jonas	Weber
Marie	Hoffmann
Leon	Schulz
Emily	Richter
David	Wolf

Aliases

SMART INDUSTRY CAMPUS

- Aliases können auch genutzt werden, um zu spezifizieren, auf welche Spalte sich bezogen werden soll, falls mehrere Spalten den gleichen Namen besitzen
- Beispiel:

```sql
SELECT p.firstname FROM Patient p WHERE p.lastname = 'Becker';
```

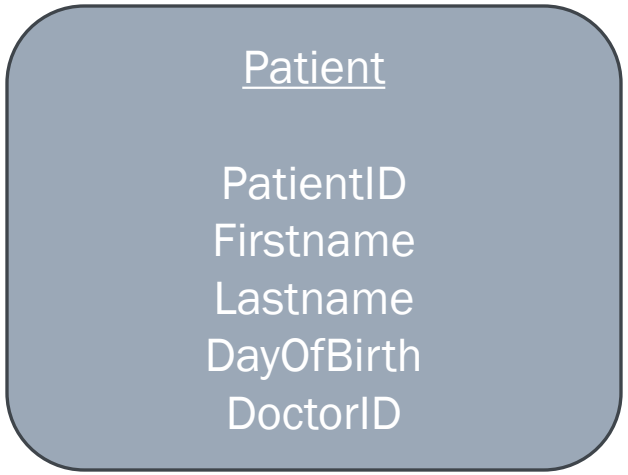
```sql
SELECT Patient.firstname FROM Patient  WHERE Patient.lastname = 'Becker';
```

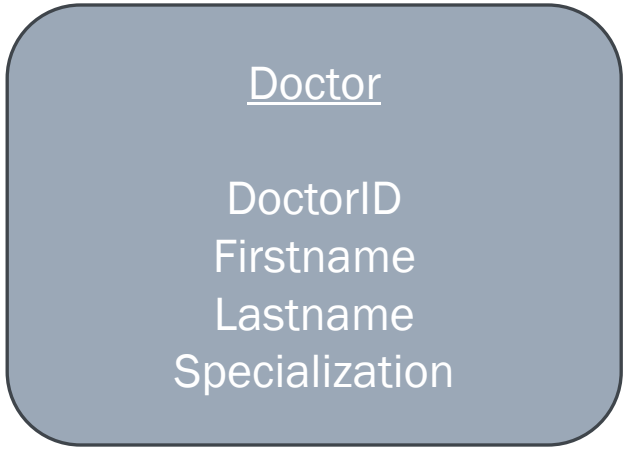
SMART INDUSTRY CAMPUS

JOINS

Was sind Joins?

- werden benötigt, um relevante Informationen zu erhalten, die auf mehrere Tabellen verteilt sind
- Beispiel: Datenbank Hospital

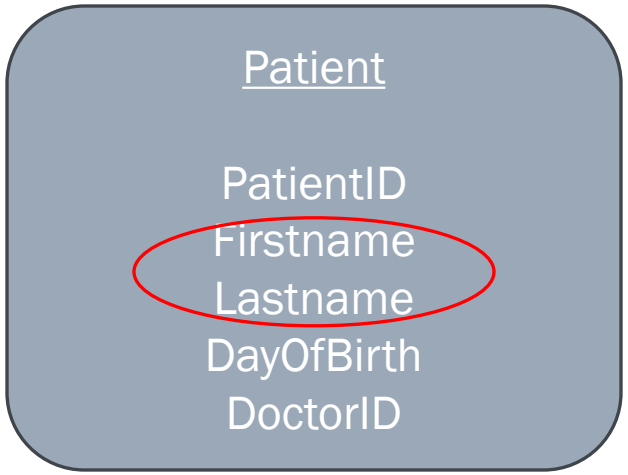


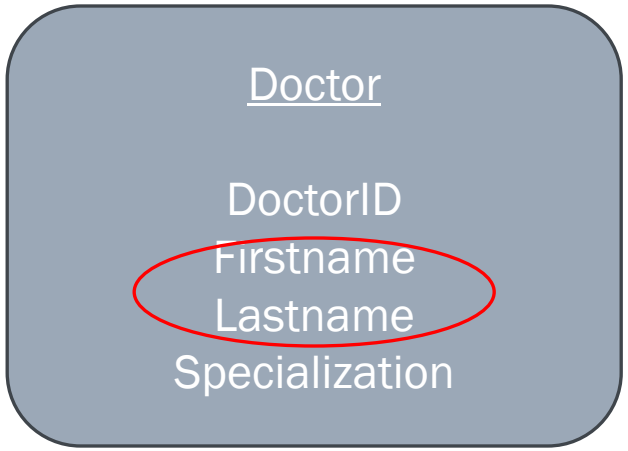


Wie heißt der Doktor, der den Patienten Fabian Becker betreut?

Was sind Joins?

- werden benötigt, um relevante Informationen zu erhalten, die auf mehrere Tabellen verteilt sind
- Beispiel: Datenbank Hospital





Wie heißt der Doktor, der den Patienten Fabian Becker betreut?

Was sind Joins?

- **nur Spalten können verbunden werden, die den gleichen Datentypen haben und das gleiche widerspiegeln!**
- Beispiel: Datenbank Hospital

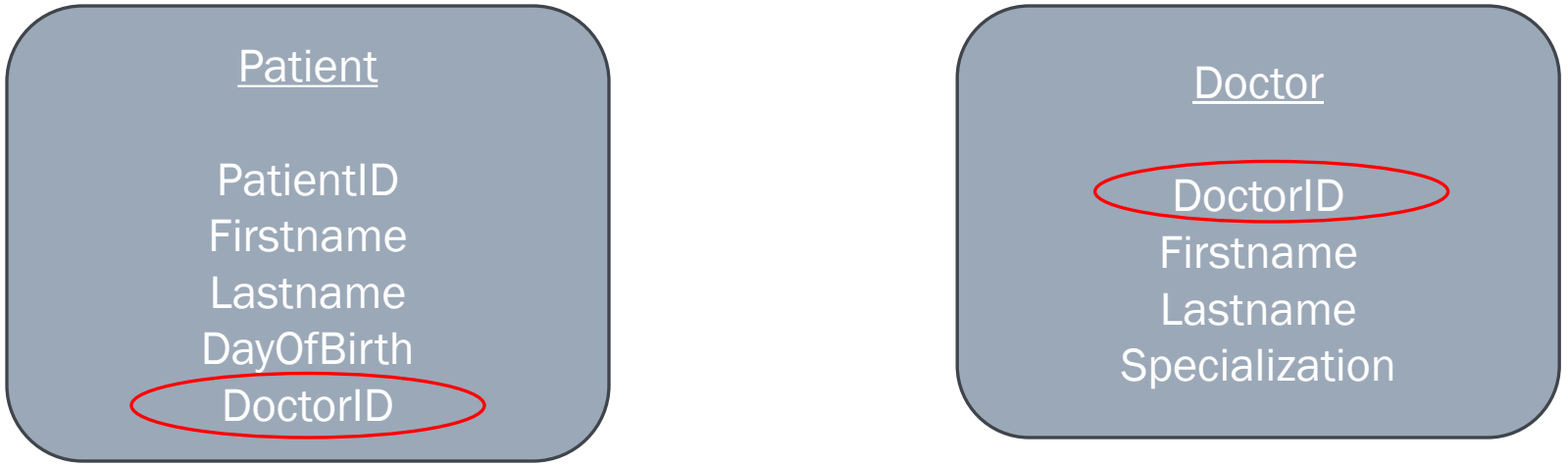
Patient
PatientID
Firstname
Lastname
DayOfBirth
DoctorID

Doctor
DoctorID
Firstname
Lastname
Specialization

Welche Spalten können miteinander logisch verbunden werden?

Was sind Joins?

- **nur Spalten können verbunden werden, die den gleichen Datentypen haben und das gleiche widerspiegeln!**
- Beispiel: Datenbank Hospital



Fremdschlüssel in der Patiententabelle und Primärschlüssel in der Doktortabelle!

Was sind Joins?

– Wünschenswert wäre, wenn die Tabelle folgendermaßen aussieht:

Wie heißt der Doktor, der den Patienten Fabian Becker betreut?

PatientID	Firstname	Lastname	Email	DayOfBirth	Age	Weight	DoctorID	DoctorID	Firstname	Lastname	Specialization	Email
1	Max	Müller	max.mueller@example.com	1985-07-12	38	80.50	1	1	Michael	Schneider	Allgemeinmedizin	michael.schneider@example.com
4	Sophie	Fischer	sophie.fischer@example.com	2000-01-15	24	55.80	1	1	Michael	Schneider	Allgemeinmedizin	michael.schneider@example.com
2	Anna	Schmidt	anna.schmidt@example.com	1992-04-23	31	65.00	2	2	Lisa	Krause	Kardiologie	lisa.krause@example.com
5	Felix	Becker	felix.becker@example.com	1989-09-30	34	78.40	2	2	Lisa	Krause	Kardiologie	lisa.krause@example.com
3	Lukas	Meier	lukas.meier@example.com	1978-11-05	45	92.30	3	3	Thomas	Berger	Orthopädie	thomas.berger@example.com

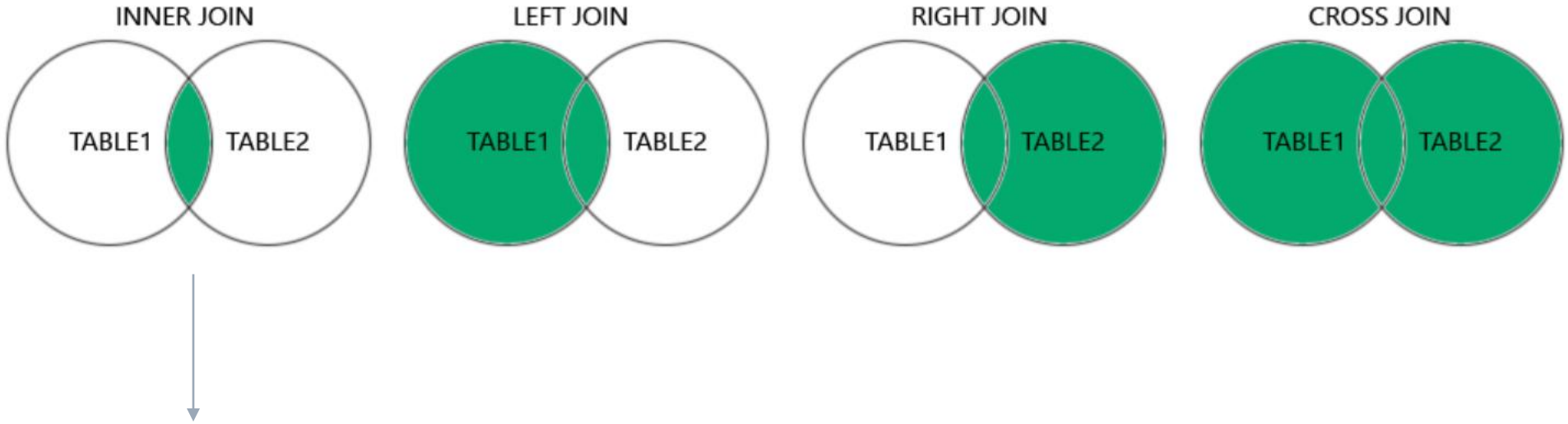
Mit JOINS machbar!

SMART INDUSTRY CAMPUS

Arten von JOINS

- Arten von JOINS:

https://www.w3schools.com/MySQL/mysql_join.asp



Gibt nur Datensätze zurück, die in beiden Tabellen eine Übereinstimmung haben.

INNER JOIN

- gibt nur Datensätze zurück, die in beiden Tabellen eine Übereinstimmung haben
- Syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```

- Beispiel: Zeige mir alle Patienten, die einen Doktor haben.
  - Alle Patienten ohne Doktor werden nicht angezeigt

SMART INDUSTRY CAMPUS

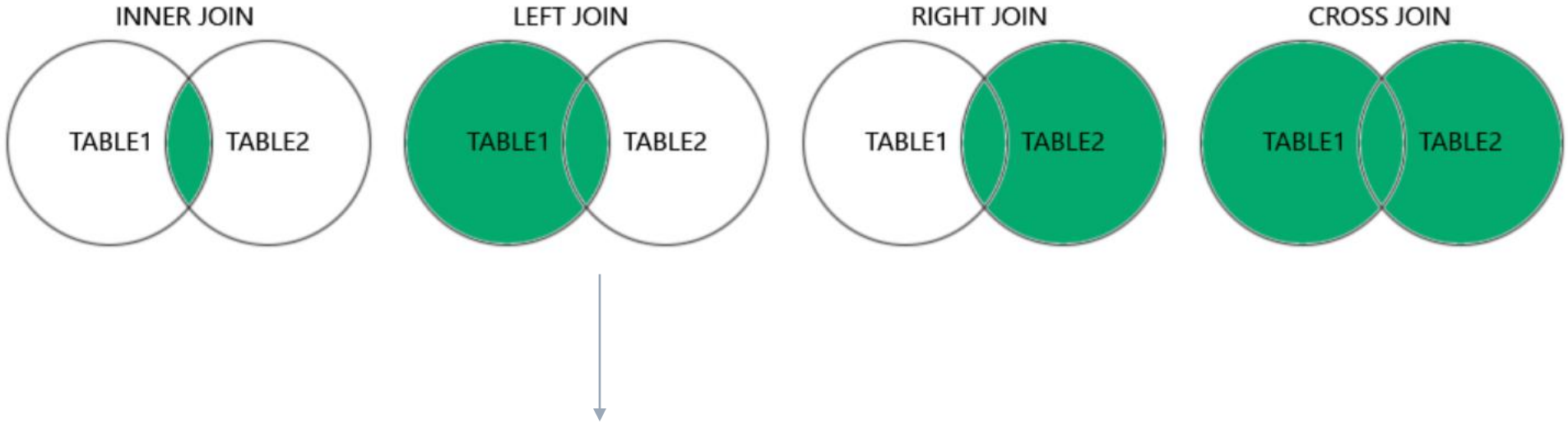
DEMO INNER JOIN

ZEIGE ALLE PATIENTEN MIT DEN NAMEN UND FACHRICHTUNGEN IHRER ÄRZTE.

Arten von JOINS

– Arten von JOINS:

https://www.w3schools.com/MySQL/mysql_join.asp



Gibt alle Datensätze der linken Tabelle zurück, auch wenn keine Übereinstimmung mit der rechten Tabelle besteht.

LEFT JOIN

Annahme:
Table 1 – Patienten
Table 2 - Doktor

SMART INDUSTRY CAMPUS

- Gibt alle Datensätze der linken Tabelle zurück, auch wenn keine Übereinstimmung mit der rechten Tabelle besteht

- Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name =
table2.column_name;
```

- Beispiele
  - Zeige mir alle Patienten, die einen und keinen Doktor haben
  - Zeige eine Liste an Kunden, die sowohl etwas bestellt haben als auch noch nichts bestellt haben

SMART INDUSTRY CAMPUS

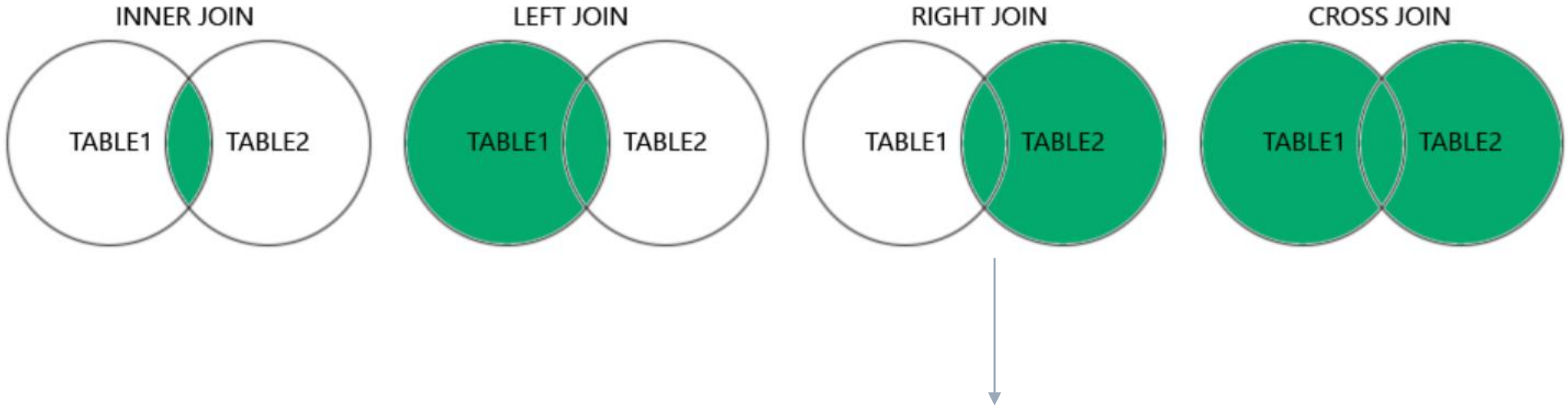
DEMO LEFT JOIN

ZEIGE ALLE PATIENTEN UND FALLS VORHANDEN IHRE ÄRZTE. FALLS KEIN ARZT ZUGEWIESEN IST, SOLL ES ANGEZEIGT WERDEN.

Arten von JOINS

– Arten von JOINS:

https://www.w3schools.com/MySQL/mysql_join.asp



Gibt alle Datensätze der rechten Tabelle zurück, auch wenn keine Übereinstimmung mit der linken Tabelle besteht.

RIGHT JOIN

Annahme:
Table 1 – Patienten
Table 2 - Doktor

SMART INDUSTRY CAMPUS

- Gibt alle Datensätze der rechten Tabelle zurück, auch wenn keine Übereinstimmung mit der linken Tabelle besteht
- wird selten genutzt, da es ein ‚umgedrehter' LEFT JOIN ist
- Syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name =
table2.column_name;
```

- Beispiele
  - Zeige mir alle Doktoren, die Patienten haben und keine Patienten haben
  - Zeige eine Liste an Produkten, auch derjenigen, die nie verkauft wurden

SMART INDUSTRY CAMPUS

DEMO RIGHT JOIN

ZEIGE ALLE VORNAMEN DER ÄRZTE UND FALLS VORHANDEN DEN VORNAMEN IHRER PATIENTEN

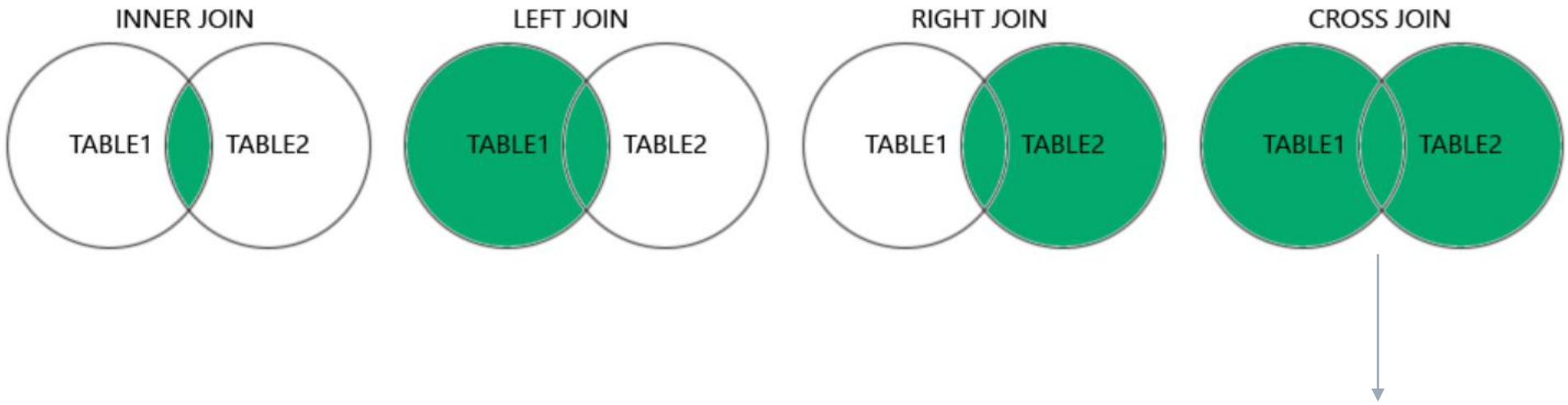
Arten von JOINS

Annahme:
Table 1 – Patienten
Table 2 - Doktor

SMART INDUSTRY CAMPUS

– Arten von JOINS:

https://www.w3schools.com/MySQL/mysql_join.asp



Kombiniert jede Zeile der ersten Tabelle mit jeder Zeile der zweiten Tabelle, auch wenn keine Verbindung besteht.

CROSS JOIN

- Erstellt jede mögliche Kombination von Zeilen aus zwei Tabellen
- Syntax:

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2
```

SMART INDUSTRY CAMPUS

DEMO CROSS JOIN

SMART INDUSTRY CAMPUS

JOINS Verknüpfen

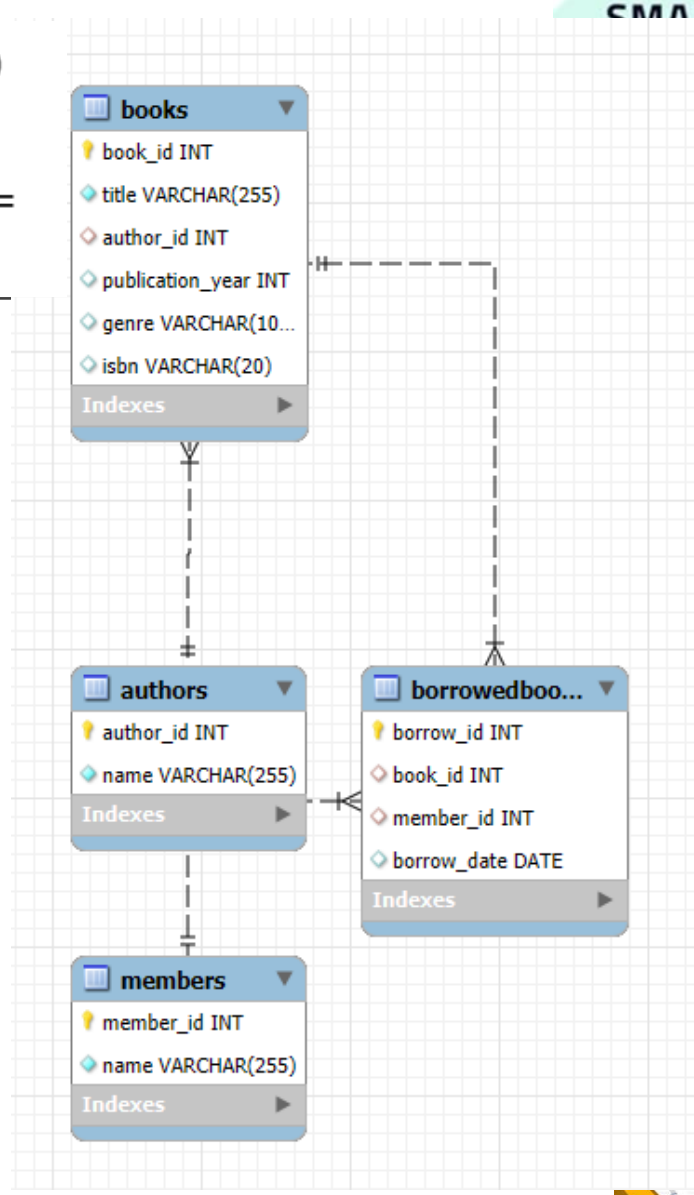
- man kann mehrere Tabellen miteinander verknüpfen
- Beispiel:
  - Eine Liste mit Patienten, ihren Ärzten und den Abteilungen, in denen die Ärzte arbeiten
  - Eine Liste von Kunden, ihren Bestellungen und den Lagerstandorten der Produkte

```
SELECT Patient.firstname , Doctors.firstname, Department.name
FROM Patient
INNER JOIN Doctors ON Patient.doctorid = Doctors.doctorid
INNER JOIN Department ON Doctors.specialization = Department.name;
```

```sql
SELECT p.firstname FROM Patient p WHERE p.lastname = 'Becker';
SELECT Patient.firstname FROM Patient  WHERE Patient.lastname = 'Becker';
```

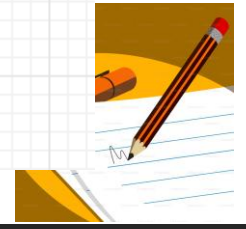
Aufgaben

```sql
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```



- Zeige die Titel der Bücher zusammen mit den Namen der Autoren
- Erwartete Ausgabe:

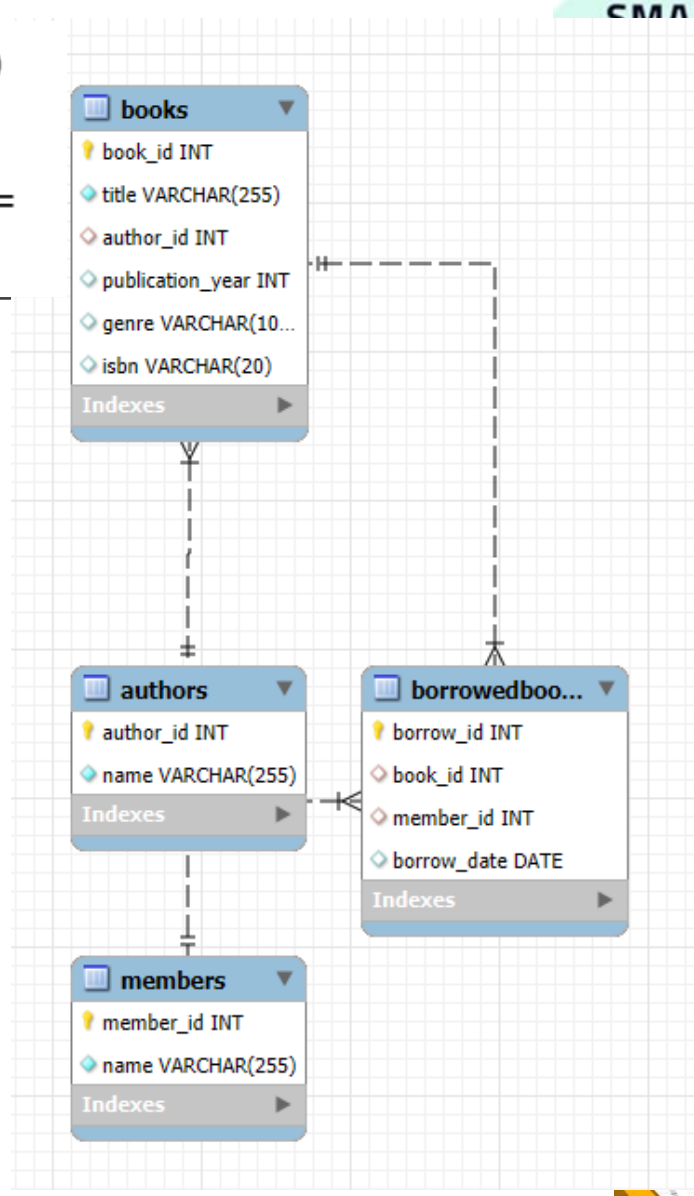
Harry Potter and the Philosopher's Stone	J.K. Rowling
1984	George Orwell
The Hobbit	J.R.R. Tolkien
Animal Farm	George Orwell
The Great Gatsby	Agatha Christie
The Catcher in the Rye	Stephen King
Pride and Prejudice	Jane Austen
To Kill a Mockingbird	Harper Lee



```sql
SELECT p.firstname FROM Patient p WHERE p.lastname = 'Becker';
SELECT Patient.firstname FROM Patient  WHERE Patient.lastname = 'Becker';
```

Aufgaben

```sql
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```



– Liste alle ausgeliehenen Bücher mit den Namen der Mitglieder auf, die sie ausgeliehen haben

– Erwartete Ausgabe:

title	name
Harry Potter and the Philosopher's Stone	Alice
1984	Bob
The Hobbit	Charlie

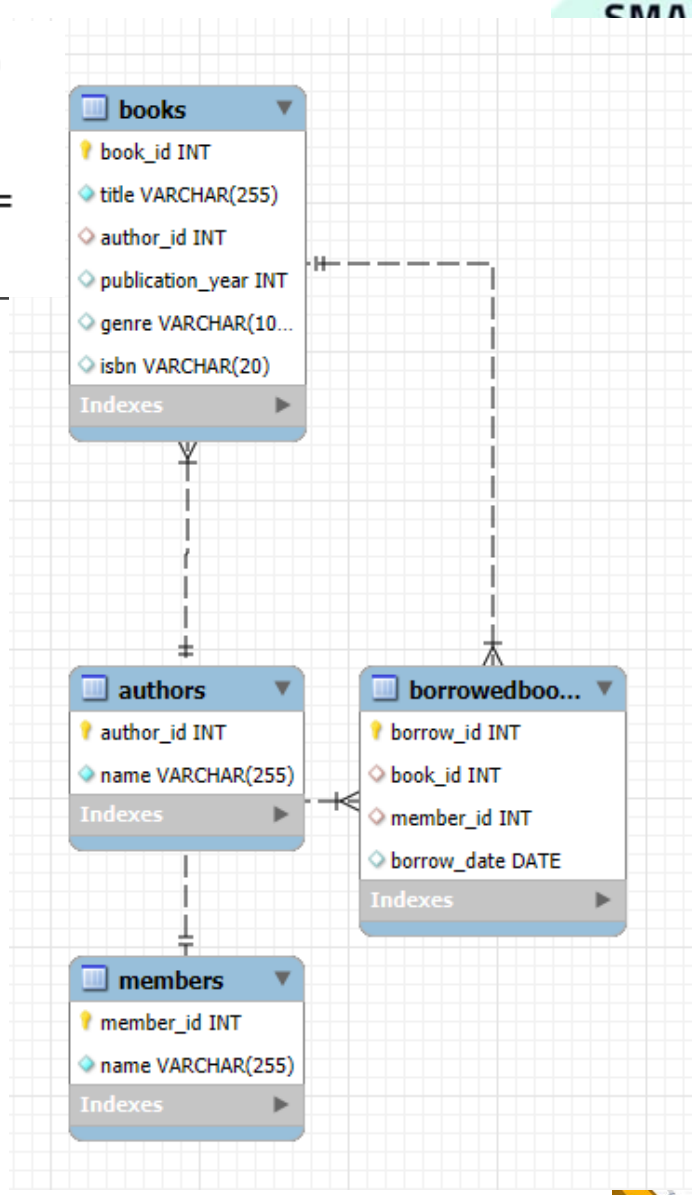
```sql
SELECT p.firstname FROM Patient p WHERE p.lastname = 'Becker';
SELECT Patient.firstname FROM Patient  WHERE Patient.lastname = 'Becker';
```

Aufgaben

```sql
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```

- Liste alle Autoren und die Bücher auf, die sie geschrieben haben (auch Autoren ohne Bücher sollen angezeigt werden)
- Erwartete Ausgabe:

name	title
J.K. Rowling	Harry Potter and the Philosopher's Stone
George Orwell	1984
George Orwell	Animal Farm
J.R.R. Tolkien	The Hobbit
Agatha Christie	The Great Gatsby
Stephen King	The Catcher in the Rye
Jane Austen	Pride and Prejudice
Harper Lee	To Kill a Mockingbird
F. Scott Fitzgerald	NULL
J.D. Salinger	NULL
J%K%Sphinx	NULL



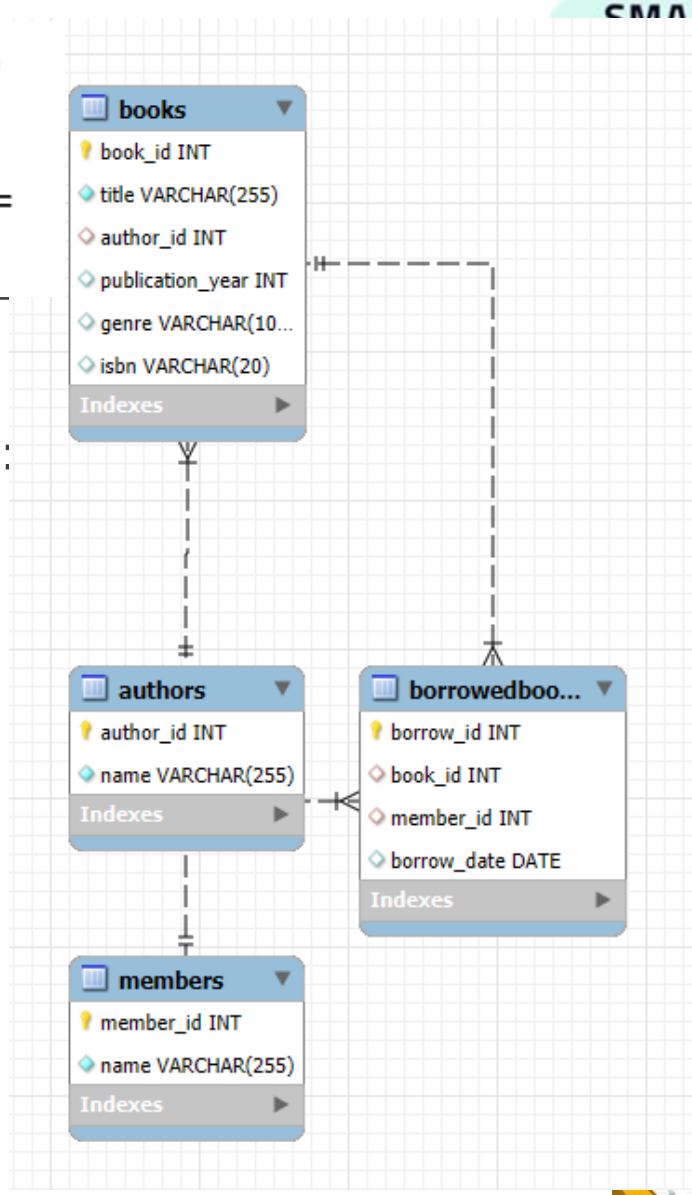
```sql
SELECT p.firstname FROM Patient p WHERE p.lastname = 'Becker';
SELECT Patient.firstname FROM Patient  WHERE Patient.lastname = 'Becker';
```

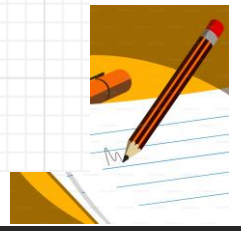
Aufgaben

```sql
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```

– Zeige alle Mitglieder und die Bücher, die sie ausgeliehen haben (auch Mitglieder ohne ausgeliehene Bücher sollen angezeigt werden):

name	title
Alice	Harry Potter and the Philosopher's Stone
Bob	1984
Charlie	The Hobbit
Lucien	NULL
Lucien	NULL





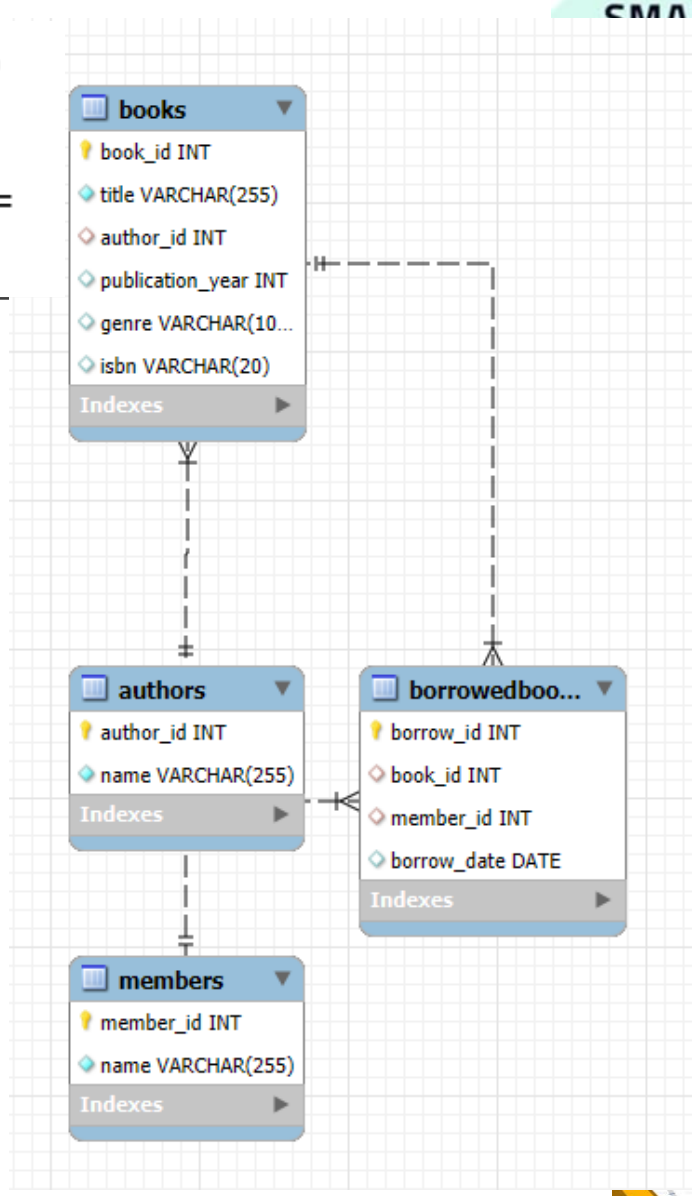
```sql
SELECT p.firstname FROM Patient p WHERE p.lastname = 'Becker';
SELECT Patient.firstname FROM Patient  WHERE Patient.lastname = 'Becker';
```

Aufgaben

```sql
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```

– Zeige alle Bücher und die Mitglieder, die sie ausgeliehen haben (auch nicht ausgeliehene Bücher sollen angezeigt werden)

name	title
Alice	Harry Potter and the Philosopher's Stone
Bob	1984
Charlie	The Hobbit
NULL	Animal Farm
NULL	The Great Gatsby
NULL	The Catcher in the Rye
NULL	Pride and Prejudice
NULL	To Kill a Mockingbird



SMART INDUSTRY CAMPUS

Zusammenfassung

INNER JOIN zeigt nur Datensätze mit Übereinstimmung.

LEFT JOIN zeigt alle Daten aus der linken Tabelle und die Passenden aus der rechten.

RIGHT JOIN zeigt alle Daten aus der rechten Tabelle und die Passenden aus der linken.

CROSS JOIN erzeugt jede mögliche Kombination von Zeilen beider Tabellen.

Mehrere JOINS können kombiniert werden, um komplexe Abfragen über mehrere Tabellen hinweg zu erstellen.

SELF JOIN

- in einigen Fällen muss man dieselbe Tabelle mit sich selbst verknüpfen
- Anwendungsfälle:
  - Hierarchien (Ärzte mit Vorgesetzen, Mitarbeiter mit Managern,...)
  - Vergleiche innerhalb der gleichen Tabelle
- Syntax:

```
SELECT t1.column_name(s), t2.column_names(s)
FROM table1 as t1
JOIN table2 as t2 ON t1.column = t2.column;
```

SELF JOIN

– Beispiel: Zeige mir alle Patienten, die das gleiche Alter haben

```sql
SELECT
    p1.Firstname AS Patient1_Firstname,
    p1.Lastname AS Patient1_Lastname,
    p2.Firstname AS Patient2_Firstname,
    p2.Lastname AS Patient2_Lastname,
    p1.Age
FROM Patient p1
JOIN Patient p2
    ON p1.Age = p2.Age
    AND p1.PatientID <> p2.PatientID;
```

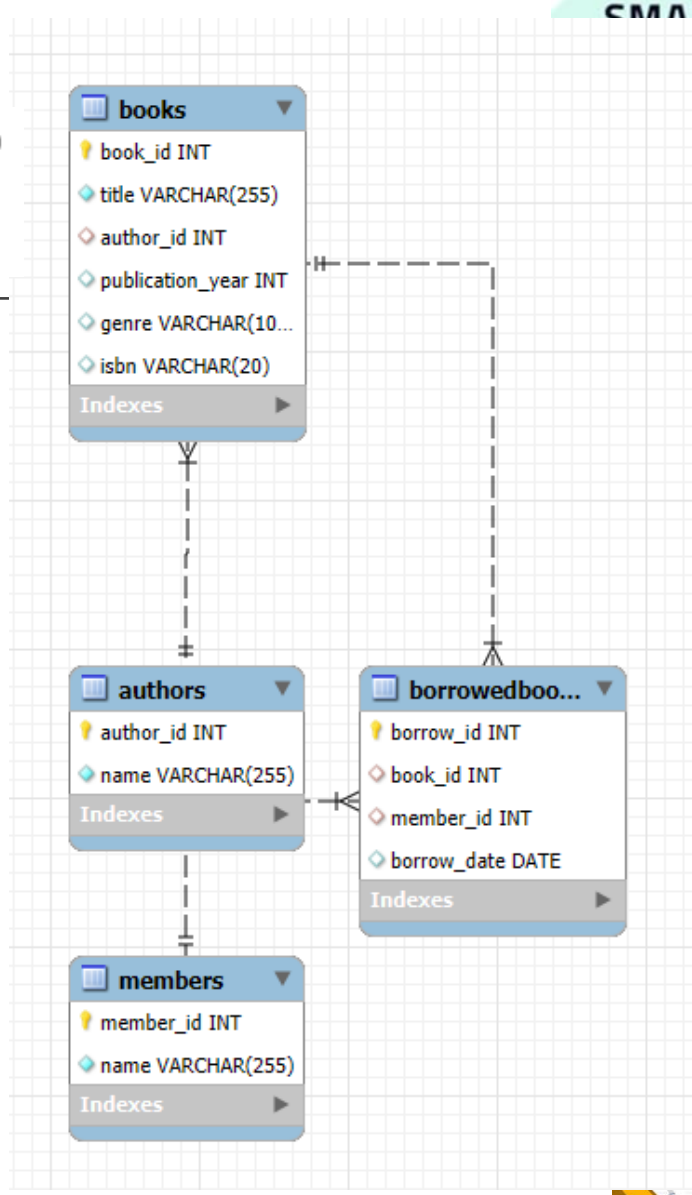
```sql
SELECT p.firstname FROM Patient p WHERE p.lastname = 'Becker';
SELECT Patient.firstname FROM Patient  WHERE Patient.lastname = 'Becker';
```

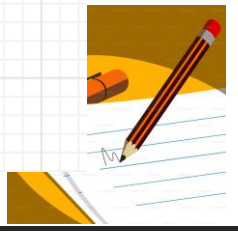
Aufgaben

```sql
SELECT t1.column_name(s), t2.column_names(s)
FROM table1 as t1
JOIN table2 as t2 ON t1.column = t2.column;
```

– Zeige mir Mitglieder, die denselben Namen tragen

name	name
Lucien	Lucien
Lucien	Lucien





NATURAL JOIN

- hier wird automatisch die gemeinsamen Spalten zweier Tabellen basierend auf ihren Namen verbunden
- es werden nur Spalten mit identischen Namen und Datentypen berücksichtigt
- entfernt automatisch eine der doppelten Spalten aus der Ausgabe
- Syntax:

```
SELECT column_name(s)
FROM table1
NATURAL JOIN table2
```

NATURAL JOIN

- Vorsicht geboten!:
  - Funktioniert nur, wenn die gemeinsamen Spalten exakt gleich heißen
  - Kann problematisch sein, wenn es mehrere gemeinsame Spalten gibt
  - Falls es keine gemeinsamen Spalten gibt, erfolgt ein CrossJoin

- Wird eher selten genutzt, da er nicht explizit angibt, über welche Spalten gejoint wird