Modul 105: Datenbanken mit SQL bearbeiten

SQL Crashkurs II: JOIN

Ausgangslage

Wir haben im letzten Arbeitsauftrag den SELECT-Befehl mit vielen Klauseln und Optionen näher angeschaut. Unser Testobjekt war eine Datenbank mit zwei Tabellen, wobei wir nur die Tabelle *Schueler* benutzten. In dieser Tabelle waren viele Daten enthalten: Name, Adresse und Ort der Schüler, Firma inklusive Adresse und Ort und Schulklasse. In der Tabelle gab es viele Redundanzen, sie war aber für unseren Zweck geeignet.

Nun wollen wir einen Schritt weitergehen. Die Daten in der neuen Datenbank sind exakt dieselben. Aus zwei Tabellen sind jedoch ganze sieben geworden: Die Daten wurden normalisiert, Redundanzen wurden bis auf eine kleine Ausnahme eliminiert. Um nun die Daten der Schüler anzuzeigen, müssen wir zwei oder mehr Tabellen verbinden. Zu diesem Zweck werden wir den SELECT-Befehl mit der JOIN-Klausel erweitern.

Ziele

- Theorie der Mengenlehre
- SELECT erweitern mit Abfragen über mehrere Tabellen
- INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN
- Aufgaben

Vorgehen

Eine **neue** Übungsdatenbank liegt vor, die Tabellen sind erstellt und die Daten vorhanden. Kopieren Sie die **Datenbank *schuelerV2.sqlite*** von Moodle in Ihre Arbeitsumgebung auf der *bmLP1*. Starten Sie darauf *Sqliteman* und öffnen Sie die Datenbank, um mit dem Arbeitsblatt loslegen zu können.

Gehen Sie dieses Arbeitsblatt Schritt für Schritt durch. Konsultieren Sie die Unterlagen, suchen Sie zusätzliche Informationen im Internet oder fragen Sie Ihre Lehrperson.

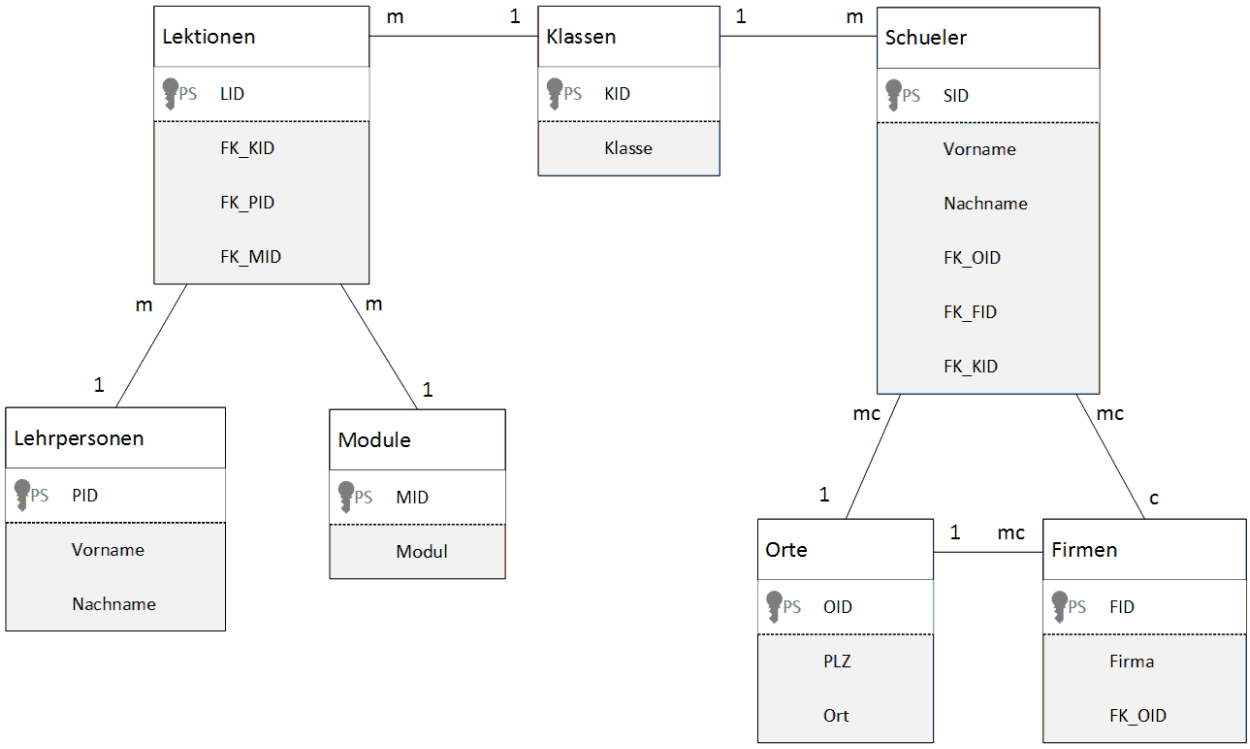
Unterlagen / Quellen

- SQL Referenz auf Moodle: SQL-Referenz.pdf
- Allgemeine SQL-Tutorials:
  - https://de.wikibooks.org/wiki/Einf%C3%BChrung_in_SQL
  - https://www.w3schools.com/sql
- SQLite-spezifisches Tutorial:
  - https://www.tutorialspoint.com/sqlite/

Zeit: 4 Lektionen

Schüler-Datenbank

Wir verwenden dieselben Daten wie im letzten Arbeitsblatt. Jedoch sind die Daten nun normalisiert, d.h. bereinigt, so dass es keine Redundanzen mehr gibt (einzige, vernachlässigbare Ausnahme: die Bezeichnung der Räume). Inkonsistenzen können trotzdem noch auftreten, weil keine CONSTRAINTS (Einschränkungen) definiert sind. Das bedeutet, dass es ungültige Werte geben kann bei den Fremdschlüsseln. CONSTRAINTS werden wir erst in Modul 104 anschauen.



Bei diesem Entity-Relationship-Diagramm handelt es sich um das logische Datenmodell, d.h. m:n-Beziehungen sind bereits mit Zwischentabelle abgebildet.

Wir haben folgende Beziehungen:

Typ	Tabelle 1	Tabelle 2	Beschreibung
1:n	Schueler	Orte	Jeder Schüler lebt in einem Ort In jedem Ort leben 0, 1 oder mehrere Schüler
1:n	Schueler	Firmen	Jeder Schüler arbeitet in 0 oder 1 Firma Jede Firma beschäftigt 0, 1 oder mehrere Lernende
1:n	Firmen	Orte	Jede Firma (bzw. Firmensitz) befindet sich an 1 Ort In jedem Ort befinden sich 0, 1 oder mehrere Firmen
1:n	Schueler	Klassen	Jeder Schüler befindet sich in einer Klasse Jede Klasse hat mehrere Schüler
m:n	Klassen	Lehrpersonen	Jede Klasse wird von mehreren Lehrpersonen unterrichtet Jede Lehrperson unterrichtet 1 oder mehrere Klassen
m:n	Klassen	Module	In jeder Klasse werden mehrere Module unterrichtet Jedes Modul wird in mehreren Klassen unterrichtet

Warum brauchen wir JOINs?

In der Schülertabelle im letzten Arbeitsblatt hatten wir alle Informationen über die Schüler in einer Tabelle: Name, Adresse, Ort, Firma und Klasse. Mit einem einfachen SELECT konnten wir alle diese Informationen abrufen. Nun haben wir dieselben Informationen über vier Tabellen verteilt. Der Endbenutzer möchte weiterhin alle Daten auf einmal sehen. Das kriegen wir hin, indem wir den SELECT-Befehl mit JOINs erweitern. Als Resultat bekommen wir eine temporäre Tabelle zurück, die genauso aussieht wie die Tabelle im letzten Arbeitsblatt.

Die wichtigsten JOIN-Typen

JOIN steht für den Verbund oder die Vereinigung zweier Mengen oder Tabellen.

Wir erweitern die FROM-Klausel des SELECT-Befehls wie folgt:

```
SELECT
     <Spaltenliste>
     FROM <linke Tabelle> AS <Alias>
     [ <Join-Typ> ] JOIN <rechte Tabelle> AS <Alias>
     ON <Bedingung>
     WHERE...;
```

Die Beschreibung der einzelnen Elemente erfolgt anhand des folgenden Beispiels:

```
SELECT *
     FROM Schueler AS s
     INNER |LEFT | RIGHT JOIN Orte AS o
     ON s.FK_OID = o.OID;
```

<linke tabelle> und <rechte tabelle>

Wenn zwei Tabellen vereinigt werden, gibt es eine linke (*Schueler*) und eine rechte (*Orte*) Tabelle.

<join typ>

Nachfolgend sind die drei Join-Typen aufgeführt (INNER JOIN kennt zwei Schreibweisen). Die Typen werden in den folgenden Kapiteln genauer beschrieben.

Join Typ	Beschreibung	Beispiel
INNER	Schnittmenge von 2 Tabellen	SELECT * FROM Schueler INNER JOIN Orte…
	Ist identisch mit INNER JOIN	SELECT * FROM Schueler JOIN Orte…
LEFT [OUTER]	Alle Datensätze von *Schueler*	SELECT * FROM Schueler LEFT JOIN Orte…
RIGHT [OUTER]	Alle Datensätze von *Orte*	SELECT * FROM Schueler RIGHT JOIN Orte…

Hinweis: LEFT und RIGHT JOIN werden zusammenfassend als OUTER JOIN bezeichnet, als Gegensatz zum INNER JOIN. Im SELECT wird *OUTER* normalerweise weggelassen.

ON

Nach dem Schlüsselwort *ON* wird die Verknüpfung vom Fremdschlüssel der einen Tabelle (*Schueler*) mit dem Primärschlüssel der anderen Tabelle (*Orte*) aufgeführt.

Beispiel: ON s.FK_OID = o.OID

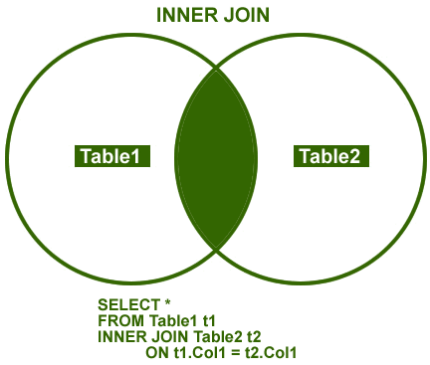
AS <alias>

Wie wir im letzten Arbeitsauftrag gesehen haben, können wir mit *AS* Spaltenüberschriften für die Ausgabe definieren. *AS* kann auch dazu verwendet werden, ein Alias für eine Tabelle zu setzen. In

diesem Fall wollen wir ein möglichst kurzes Alias (meist nur 1 Buchstabe), damit wir die Attributliste und ON-Bedingung kürzer und übersichtlicher darstellen können.

Beispiele dazu finden Sie auf den folgenden Seiten.

INNER JOIN



Bedeutung

INNER JOIN ist die Schnittmenge der beiden Tabellen. Es werden nur diejenigen Datensätze zurückgeliefert, die in beiden Tabellen vorkommen.

Definition

INNER JOIN verbindet nur die Zeilen zweier Tabellen, welche die ON-Klausel erfüllen.

Testen Sie folgendes Statement:

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort
     FROM Schueler AS s
     INNER JOIN Orte AS o
     ON s.FK_OID = o.OID;
```

Ausgesprochen

Retourniere nur die Zeilen aus den Tabellen *Schueler* und *Orte*, in denen der Primärschlüssel *OID* und der Fremdschlüssel *FK_OID* gleich ist

	Vorname	Nachname	Adresse	PLZ	Ort
1	Nevio	Kalkhoff	Looslistrasse	3800	Unterseen
2	Finn	Wehlau	Lagerhausweg	3292	Busswil BE
3	Livio	Sanders	Loryplatz	3855	Brienz BE
4	Silvan	Trein	Hochbühlweg	4932	Lotzwil
5	Mia	Sieverding	Hofweg	3414	Oberburg
6	Larissa	Pargmann	Aehrenweg	3065	Bolligen
7	Luisa	Grote	Bernstrasse	3363	Oberönz
8	Lena	Havekost	Colombstrasse	3073	Gümligen
9	Lia	Köster	Boehlenstrasse	3270	Aarberg
10	Levin	Trenkamp	Lerberstrasse	3128	Rümligen

Die Attribute Vorname, Nachname und Adresse stammen aus der Tabelle *Schueler*, die Attribute PLZ und Ort aus der Tabelle *Orte*. *s* ist das Alias der Tabelle *Schueler*, *o* das Alias der Tabelle *Orte*. Dieses Alias wird den Attributnamen vorangestellt, damit Konflikte vermieden werden (wenn zwei Tabellen Attribute haben, die gleich heissen) und damit klar ist, aus welchen Tabellen die Attribute stammen.

Hinweis: Beim INNER JOIN kann das Schlüsselwort INNER weggelassen werden. JOIN bedeutet also dasselbe wie INNER JOIN.

Beobachtung I

INNER JOIN retourniert Teilmengen von Tabellen, falls **nicht** alle Datensätze die ON-Klausel erfüllen.

Beobachtung II (Umkehrung von I)

Erfüllen alle Datensätze einer Tabelle die ON-Klausel, wird die Tabelle vollständig retourniert.

Aufgabe 1

Wie viele Datensätze werden mit dem SELECT oben zurückgegeben?

```
SELECT count(*)
FROM Schueler AS s
INNER JOIN Orte AS o
ON s.FK_OID = o.OID -270
```

Aufgabe 2

Geben Sie von allen Lernenden den Namen und die Firma zurück.

```
SELECT schueler.Nachname, firmen.Firma FROM schueler
INNER JOIN firmen ON schueler.FK_FID = firmen.FID
```

Wie viele Datensätze sind es?

254

Warum werden weniger Datensätze zurückgegeben als beim vorherigen SELECT?

Manche haben da ein leeres Feld... Null oder ""

Tabellenverbund mit mehr als 2 Tabellen

Wir wollen nun von allen Schülern Name, Adresse, Wohnort **und** Firma anzeigen. Es handelt sich dabei um einen Tabellenverbund mit 3 Tabellen:

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort, f.Firma
    FROM Schueler AS s
    INNER JOIN Orte AS o
    ON s.FK_OID = o.OID
    INNER JOIN Firmen AS f
    ON s.FK_FID = f.FID;
```

Führen Sie dieses SELECT aus und halten Sie Ihre Beobachtungen fest.

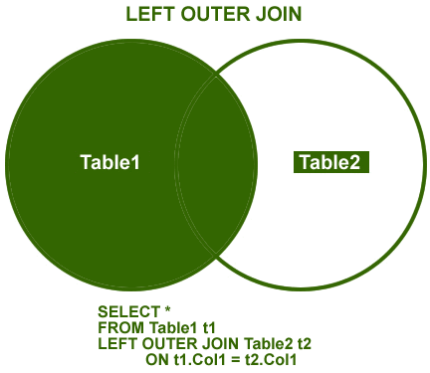
Ganz normale Abfrage?

Aufgabe 3

Zusätzlich zu Name, Adresse, Ort und Firma soll auch noch die Klasse ausgegeben werden. Formulieren Sie den notwendigen SELECT (Tabellenverbund mit 4 Tabellen).

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort, f.Firma, k.Klasse FROM
Schueler AS s INNER JOIN Orte AS o ON s.FK_OID = o.OID INNER JOIN Firmen AS f ON
s.FK_FID = f.FID INNER JOIN klassen AS k ON FK_KID = k.KID
```

LEFT JOIN



Bedeutung

Beim LEFT JOIN werden alle Datensätze der linken Tabelle zurückgegeben.

Definition

LEFT JOIN gibt die linke Tabelle vollständig aus und von der rechten Tabelle nur die Datensätze, welche die ON-Klausel erfüllen.

Testen Sie folgendes Statement:

```
SELECT s.Vorname, s.Nachname, s.Adresse, f.Firma
    FROM Schueler AS s
    LEFT JOIN Firmen AS f
    ON s.FK_FID = f.FID;
```

Ausgesprochen

Retourniere alle Zeilen der linken Tabelle *Schueler* und nur die Zeilen der rechten Tabelle *Firmen*, in denen der Primärschlüssel *FID* und der Fremdschlüssel *FK_FID* gleich ist.

	Vorname	Nachname	Adresse	Firma
1	Nevio	Kalkhoff	Looslistrasse	Informatik Service Center ISC-EJPD
2	Finn	Wehlau	Lagerhausweg	Eidg. Departement für Wirtschaft,
3	Livio	Sanders	Loryplatz	Noser Young Professionals AG
4	Silvan	Trein	Hochbühlweg	SPT Roth AG
5	Mia	Sieverding	Hofweg	Swisscom AG
6	Larissa	Pargmann	Aehrenweg	Glaux Soft AG
7	Luisa	Grote	Bernstrasse	Post CH AG
8	Lena	Havekost	Colombstrasse	Xplanis AG
9	Lia	Köster	Boehlenstrasse	login Berufsbildung AG
10	Levin	Trenkamp	Lerberstrasse	Adfinis SyGroup AG
11	Rafael	Emke	Bankgässchen	*NULL*

Aufgabe 4

Wie viele Datensätze werden mit diesem SELECT zurückgegeben und warum?

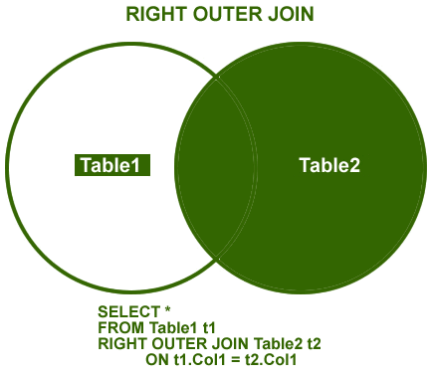
Was bedeutet der Attributwert NULL bei der Firma von Rafael Emke?

Aufgabe 5

Geben Sie wie in Aufgabe 3 von allen Schülern aus: Name, Adresse, Ort, Firma und Klasse. Diesmal sollen aber **alle** Schüler aufgeführt werden.

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort, f.Firma, k.Klasse FROM Schueler
AS s LEFT JOIN Orte AS o ON s.FK_OID = o.OID LEFT  JOIN Firmen AS f ON s.FK_FID =
f.FID LEFT JOIN klassen AS k ON FK_KID = k.KID
```

RIGHT JOIN



Bedeutung

Beim RIGHT JOIN werden alle Datensätze der rechten Tabelle zurückgegeben.

Definition

RIGHT JOIN gibt die rechte Tabelle vollständig aus, und von der linken Tabelle nur die Datensätze, welche die ON-Klausel erfüllen.

Testen Sie folgendes Statement:

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort
    FROM Schueler AS s
    RIGHT JOIN Orte AS o
    ON s.FK_OID = o.OID
    WHERE o.Kanton = "BE";
```

Ausgesprochen

Retourniere alle Zeilen der rechten Tabelle *Ort* und nur die Zeilen der linken Tabelle *Schueler*, in denen der Primärschlüssel *OID* und der Fremdschlüssel *FK_OID* gleich ist.

Ergebnis

Als Ergebnis erhalten wir folgende Fehlermeldung: "RIGHT and FULL OUTER JOINs are not currently supported". D.h. mit SQLite ist ein RIGHT JOIN nicht möglich.

Wenn wir trotzdem alle Orte und die Schüler, die jeweils im Ort wohnen, ausgeben wollen, müssen wir die beiden Tabellen vertauschen, siehe folgendes Kapitel.

Konversion (Umkehrung) RIGHT JOIN zu LEFT JOIN

RIGHT JOIN

```
SELECT *
   FROM Schueler AS s
   RIGHT JOIN Orte AS o
   ON s.FK_OID = o.OID;
```

Äquivalentes LEFT JOIN

```
SELECT s.*, o.*
   FROM Orte AS o
   LEFT JOIN Schueler AS s
   ON s.FK_OID = o.OID;
```

Regelschritte

1. Im *SELECT* wird *RIGHT JOIN* mit *LEFT JOIN* ersetzt.
2. Linke Tabelle (nach *FROM*) und rechte Tabelle (nach *LEFT JOIN*) werden vertauscht.
3. Falls für die Attributliste der Joker * angewandt wurde, muss diese Liste wie oben angepasst werden.

Aufgabe 6

Wieso muss die Attributliste (Regelschritt 3) im SELECT angepasst werden?

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort, f.Firma, k.Klasse FROM Schueler AS s
INNER JOIN Orte AS o ON s.FK_OID = o.OID INNER JOIN Firmen AS f ON s.FK_FID = f.FID
INNER JOIN klassen AS k ON FK_KID = k.KID
```

Testen Sie folgendes Statement:

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort
    FROM Orte AS o
    LEFT JOIN Schueler AS s
    ON s.FK_OID = o.OID
    ORDER BY o.Ort DESC;
```

	Vorname	Nachname	Adresse	PLZ	Ort
1	Elina	Grever	Elfenstrasse	3052	Zollikofen
2	Andrin	Freese	Kehrgasse	3436	Zollbrück
3	Diego	Wendt	Dammweg	3436	Zollbrück
4	Lina	Hellwig	Könizstrasse	3436	Zollbrück
5	*NULL*	*NULL*	*NULL*	3086	Zimmerwald
6	Laurin	Brockshus	Jubiläumsplatz	3184	Wünnewil
7	Leonie	Hoffmann	Amthausgässchen	3048	Worblaufen
8	Mia	Bremer	Grossackerstrasse	3252	Worben
9	*NULL*	*NULL*	*NULL*	3076	Worb
10	Anna	Wagner	Galgenfeldweg	3812	Wilderswil

Es werden alle Orte ausgegeben mit den Schülern, die da wohnen.

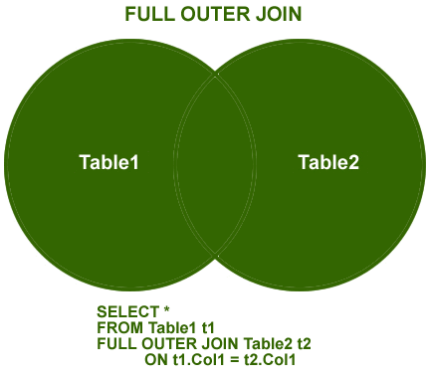
Aufgabe 7

Warum steht bei *Vorname*, *Nachname* und *Adresse* oft *NULL*, z.B. in Zeile 5?

Warum gibt es Orte, die mehr als 1x vorkommen, z.B. Zollbrück?

In der Tabelle Orte gibt es 187 Datensätze. Bei der soeben ausgeführten Abfrage erhalten wir 304 Datensätze zurück. Warum dieser Unterschied?

FULL OUTER JOIN



Bedeutung

Beim FULL OUTER JOIN werden alle Datensätze von beiden Tabelle zurückgegeben.

Definition

Der FULL OUTER JOIN liefert alle Datensätze beider Tabellen. Wenn Datensätze nach der Verknüpfungsbedingung zusammenpassen, werden sie in einer Zeile ausgegeben; wo es keinen "Partner" gibt, wird ein NULL-Wert angezeigt.

FULL OUTER JOIN ist in SQLite nicht implementiert!

Mit einem Trick können wir trotzdem eine Abfrage absetzen, die dem FULL OUTER JOIN entspricht:

```
SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort
    FROM Schueler AS s
    LEFT JOIN Orte AS o
    ON s.FK_OID = o.OID
UNION SELECT s.Vorname, s.Nachname, s.Adresse, o.PLZ, o.Ort
    FROM Orte AS o
    LEFT JOIN Schueler AS s
    ON s.FK_OID = o.OID
    ORDER BY Ort;
```

Wir hängen die beiden SELECT, die wir vorhin einzeln ausgeführt haben, mit UNION zusammen.
Wir könne also mit UNION zwei Abfragen zu einer vereinigen. Wichtig dabei:
Die Attributliste muss bei beiden SELECT identisch sein!

Wir werden UNION in diesem Modul nicht weiter anschauen. Führen Sie trotzdem die Abfrage oben aus und schauen Sie sich das Ergebnis an.

Weitere Aufgaben

Speichern Sie sämtliche SQL-Befehle in die Datei *AB105-03.sql* ab.

Vergeben Sie bei allen Aufgaben jeweils sinnvolle Attributtitel, wenn notwendig. Beispiel: Wenn Adresse und Ort von Schülern und Firmen aufgeführt werden sollen, dann könnten die Titel wie folgt heissen: "Adresse privat", "Ort privat", "Adresse Firma" und "Ort Firma".

Aufgabe 8

Analysieren Sie als Erstes die Datenbank. Ziehen Sie das Entity-Relationship-Diagramm bei und prüfen Sie alle Beziehungen und die Mengenangaben m bzw. mc und 1 bzw. c. Schauen Sie sich die Inhalte von allen Tabellen mit SELECT an, insbesondere die Fremdschlüssel.

Beantworten Sie folgende Fragen:

- Sind die Beziehungen im ERD korrekt?
- Was fällt Ihnen bezüglich Fremdschlüssel und NULL-Werten auf?
- Welche Tabellenverbunde können mit INNER JOIN realisiert werden?
- Welche benötigen ein LEFT JOIN?

Reicht die Prüfung auf Fremdschlüssel mit dem Wert NULL, um zu sehen, ob LEFT JOIN notwendig ist? Was ist, wenn wir alle Orte anzeigen wollen mit den Schülern, die da wohnen? Wie können wir überprüfen, ob bei dieser Abfrage ein LEFT JOIN notwendig ist?

Aufgabe 9

Führen Sie den Namen und die Klasse der Schüler auf, die sich in einer der Klassen INF.1C bis INF.1F befinden.

```
SELECT s.vorname, s.Nachname, k.Klasse, FROM schueler as S
INNER JOIN Klassen as k ON S.FK_KID = k.kid
WHERE k.Klasse BETWEEN
```

Aufgabe 10

Führen Sie alle Klassen auf mit der Anzahl Schüler, die in dieser Klasse sind. Vergeben Sie für die Anzahl einen entsprechenden Spaltentitel.

```
SELECT k.klasse , Count(s.FK_KID) as "Anzahl Schueler" FORM schueler as S JOIN
Klassen as K on s.FK_KID = k.kid
GROUP BY k.klasse;
```

Aufgabe 11

Geben Sie alle Schüler aus, die in einem Ort wohnen, wo die Postleitzahl mit 3 beginnt.
Attributliste: Nachname, Vorname, PLZ und Ort. Sortieren Sie das Ergebnis 1. nach Postleitzahl absteigend und 2. nach Nachname aufsteigend.

```
SELECT k.klasse , s.nachname, S.Vorname as FROM schueler as S JOIN
orte as o on s.FK_OID = o.OID
WHERE o.PLZ between 3000 and 3999
ORDER BY o.PLZ DESC, Nachname;
```

Aufgabe 12

Führen Sie alle Klassen auf mit den Modulen, die in diesen Klassen unterrichtet werden. Sortieren Sie das Ergebnis nach Klasse aufsteigend.

```
SELECT k.klasse, m.modul from KLASSEN as k
join module as m
join module m on l.FK_MID = m.MID
ORDER BY klasse;
```

Zusatzaufgaben

Diese Aufgaben sind schwieriger als die vorherigen und sollen als Herausforderung betrachtet werden!

Aufgabe 13

Listen Sie alle Schüler mit Vorname, Nachname, Adresse privat, Ort privat, Firma, Adresse Firma und Ort Firma auf. Vergeben Sie entsprechende Spaltentitel.

```
SELECT s.nachname, s,vorname, s.adresse as 'Adresse Privat'
o.ort as 'Ort Privat'
t.Ort as 'ort firma from shueler as s
left join orte as i in f s.fk_oid = o.oid
left join firmen as f on s.fk_fid = f.fid
left join orte as t on f.fk_oid = t.oid;
```

Aufgabe 14

Führen Sie alle Klassen auf mit den Modulen, die in diesen Klassen unterrichtet werden (Aufgabe 12). Zeigen Sie zudem die Lehrperson, die jeweils das Modul unterrichtet.

```
SELECT k.Klasse, m.Modul FROM Klassen k
JOIN Lektionen l ON l.FK_KID =  k.KID
JOIN module m on l.fK_MID = m.mid
order by k.klasse
```

Aufgabe 15

Geben Sie alle Schüler aus, bei denen Herr Maurer das Modul 105 unterrichtet.
Attributliste: Vor- und Nachname der Schüler, Klasse, Modul und Lehrperson.
Vor- und Nachname der Lehrperson sollen dabei als **ein** Attributwert angezeigt werden (d.h. Vor- und Nachname müssen mit dem Operator || verkettet werden).

```
SELECT s.vorname, s.nachname, k.klasse, m.modul,
p.vorname ||" || p.nachname as 'Lehrperson' from shueler
join Klassen as k on s.FK_KID = k.KID
join Lektionen as l on l.FK_KID = k.kid
join Lehrpersonen p on l.fk_PID = p.PID
JOin module m on l.fk_mid = m.Mid
WHERE m.modul like 'Modul 100%'
AND p.nachname = 'Maurer';
```

Aufgabe 16

Geben Sie von allen Schülern, die in Niederscherli wohnen, folgende Attribute aus:
Vorname, Nachname, PLZ, Ort, Firma, Klasse, Module und Lehrpersonen.
Da jeder Schüler mehrere Module besucht, erhalten Sie pro Schüler mehrere Einträge (1 Eintrag / Modul). D.h. die ersten Attribute (Name, Ort, Firma, Klasse) werden mehrfach aufgeführt.

Es handelt sich dabei um einen Tabellenverbund von allen 7 Tabellen!

	Vorname	Nachname	PLZ	Ort	Firma	Klasse	Modul	Lehrperson
1	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul 100 - Daten charakterisieren, aufbereiten und a...	Ralph Maurer
2	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul 117 - Informatik- und Netzinfrastruktur für klei...	Daniel Schär
3	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul 431 - Aufträge im IT-Umfeld selbstständig dur...	Daniel Schär
4	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul 403 - Programmabläufe prozedural implement...	Barbara Bielawski
5	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul 924 - Mathematik / Naturwissenschaften	Daniel Vitale
6	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul 945 - Englisch 1	Ines Ramseier
7	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul ABU - Allgemeinbildender Unterricht	Simone Gerber
8	Elin	Decker	3145	Niederscherli	Asetronics AG	INF.1l	Modul SP - Sport	Benjamin Adam
9	Raphael	Brüggemann	3145	Niederscherli	login Berufsbildung AG	INF.1l	Modul 100 - Daten charakterisieren, aufbereiten und a...	Ralph Maurer
10	Raphael	Brüggemann	3145	Niederscherli	login Berufsbildung AG	INF.1l	Modul 117 - Informatik- und Netzinfrastruktur für klei...	Daniel Schär

usw.

Notieren Sie den entsprechenden SELECT.

```
SELECT s.vorname, s.nachname, o.plz, o.ort, f.firma, k.klasse, m.modul,
p.vorname || ' ' || p.nachname as "Lehrperson" from schueler as s
left join orte o on s.fk_oid = o.oid
left join firmen f on s.fk_fid = f.fid
join Klassen k on s.fk_kid = k.kid
join Lektion l on l.fk_kid = k.kid
join module m on l.fk_mid = m.mid
```