Aufgabe 1 (Preliminaries) This task requires a working installation of SQLite. The way to install it depends on your platform, but it is usually very easy. For Ubuntu 20.04 it’s just sudo apt install sqlite3. Aufgabe 2 (University Schema) Laden Sie die gegebene SQLite-Datenbank university.db.

1. Schauen Sie sich das Schema an: .tables .schema --indent
2. Schauen Sie sich die Instanz an (die Daten in jeder Tabelle): select * from ;
3. Vergleichen Sie mit dem in der Vorlesung dazu gegebenen Schema. Verstehen Sie die Bedeutung der Datens¨atze in allen diesen Tabellen, die Primary Keys und die Foreign-Key-Beziehungen zwischen den Tabellen. Beantworten Sie insbesondere folgende Fragen: (a) Wieviele instructors kann ein student als advisor haben? Warum?

solution

0-n , weil es eine many-to-many beziehung von student to instrutor ist, die table advisor ist die zwischentabelle

(b) Wieviele students kann ein instructor als advisor haben? Warum?

solution

0-n , weil es eine many-to-many beziehung von student to instrutor ist, die table advisor ist die zwischentabelle

(c) Warum gibt es keine Foreign-Key-Beziehung von section auf time slot?

solution

weil es eine one to many beziehung ist, und ein time_slot kann von mehreren sections verwendet werden

Aufgabe 3 (Abfragen mit publications.db)

Es ist das umseitige Schema gegeben. Die zugeh¨orige SQLite-Datenbank ist in publications.db. Geben Sie SQL-Queries f¨ur alle folgenden Aufgaben an. Testen Sie die Queries mit dieser Datenbank.

1. Finden Sie alle Autoren, die in San Francisco wohnen.

solution

```
select * from authors where city = "San Francisco"
```

2. Wieviele Titel beginnen mit 'S' ?

solution

```
select * from titles where title like "S%";
```

3. Bestimmen Sie den durchschnittlichen Preis eines Titels.

solution

```
select avg(price) from titles;
```

avg(price)

14.76625 4. Geben Sie das Datum aller Verk¨aufe des Ladens 'Bookbeat' an.

solution

5. Geben Sie alle Titel aus, die im Laden 'Bookbeat' verkauft wurden.

solution

```
SELECT  *
FROM sales
JOIN stores
```

```
ON sales.stor_id = stores.stor_id
WHERE stores.stor_name = "Bookbeat";
```

6. Finden Sie Titel und Preis des teuersten Buches.

solution

```
SELECT  *
       ,MAX(price)
FROM titles;
```

7. Finden Sie s¨amtliche B¨ucher, die teurer als das billigste Psychologie-Buch sind.

solution

```
SELECT  *
FROM titles
WHERE price > (
SELECT  MIN(price)
FROM
(
    SELECT  *
    FROM titles
    WHERE type like '%psychology%'
));
```

8. Finden Sie die Autoren, die in einem Staat wohnen, in dem es keinen der erfassten L¨aden gibt.

solution

```
SELECT  *
FROM authors
WHERE state not IN (
SELECT  stores.state
FROM stores
JOIN authors
ON stores.state = authors.state
GROUP BY  stores.state);
```

9. Geben Sie die St¨adte an, in denen es sowohl Autoren wie auch Verleger gibt

```
SELECT  *
FROM stores
JOIN authors
ON stores.state = authors.state
GROUP BY  stores.state
```

solution

1 Datenbanken¨Ubungsblatt 5

1. Bestimmen Sie alle B¨ucher, die den gleichen Typ besitzen wie das Buch ’Net Etiquette’.

solution

```
SELECT  *
FROM titles
WHERE type = (
SELECT  type
```

```
FROM titles
WHERE title = "Net Etiquette");
```

11. Geben Sie einen SQL Ausdruck an, der die Bücher typen zusammen mit der Anzahl Bücher jedes Typs ausgibt.

solution

```
SELECT  COUNT(type)
       ,type
       ,title
FROM titles
GROUP BY  type;
```

12. Geben Sie einen SQL Ausdruck an, der die Bücher typen zusammen mit der Anzahl Bücher jedes Typs ausgibt, von denen es mehr als 2 verschiedene Bücher gibt.

solution

```
SELECT  *
FROM
(
    SELECT  COUNT(type) AS count_type
           ,type
           ,title
    FROM titles
    GROUP BY  type
)
WHERE count_type > 2
```

13. Geben Sie einen SQL Ausdruck an, der die Anzahl Autoren pro Staat auflistet, wobei die Ausgabe nach Anzahl Autoren sortiert sein soll.

solution

```
SELECT  *
       ,COUNT(state) AS count_state
FROM authors
GROUP BY  state
ORDER BY count_state;
```

14. Bestimmen Sie alle Publisher, welche weniger B¨ucher herausgegeben haben als der Durchschnitt.

solution

```
SELECT  *
FROM
(
    SELECT  *
           ,COUNT(title_id) AS count_title_id
    FROM publishers
    JOIN titles
    ON publishers.pub_id = titles.pub_id
    GROUP BY  titles.pub_id
)
WHERE count_title_id < (
SELECT  AVG(count_title_id) AS avg_count_title_id
FROM
(
    SELECT  *
           ,COUNT(title_id) AS count_title_id
    FROM publishers
    JOIN titles
    ON publishers.pub_id = titles.pub_id
    GROUP BY  titles.pub_id
```

```
))
```