

NULL-Werte

Wenn Datensätze in einer Spalte „keinen Wert“ enthalten, wird das als NULL-Wert bezeichnet. In manchen Anwendungen ist das sinnvoll. Wenn eine Spalte NULL-Werte enthalten soll, muss man das in der Definition der Spalte festlegen (Häkchen bei „Null“ setzen):

Name	Typ	Länge/Werte	Standard	Kollation	Attribute	Null	A	J	Kc
PEnde	DATE		NULL			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Der Standard-Wert muss nicht unbedingt NULL sein.

Um Zeilen mit NULL-Werten zu finden, gibt es die Vergleichsbefehle `IS NULL` bzw. `IS NOT NULL`:

SELECT * FROM presidents WHERE PTodestag IS NULL

→ alle noch lebenden (Ex-)Präsidenten

SELECT * FROM presidents

WHERE PTodestag IS NULL AND PEnde IS NOT NULL

→ alle noch lebenden Ex-Präsidenten, aber nicht der amtierende Präsident

Die anderen Vergleichsoperatoren liefern bei NULL-Werten immer „falsch“ zurück. Man bekommt die noch lebenden Präsidenten also weder mit

SELECT * FROM presidents WHERE PTodestag < '2000-01-01'

noch mit

SELECT * FROM presidents WHERE PTodestag >= '2000-01-01'

Beim (aufsteigenden) Sortieren werden NULL-Werte vor allen anderen Werten angezeigt.

Mondial-Datenbank

Erstellen Sie eine Datenbank „mondial“ und importieren Sie die Daten aus der Datei `T:\Schule\datenbanken\Mondial.sql`

Diese Datenbank enthält eine Vielzahl von Tabellen mit geographischen Informationen. Wir betrachten zunächst nur die Tabellen `city`, `country`, `river` und `mountain`. Später werden wir auch die anderen Tabellen verwenden.

Schreiben Sie SQL-Anfragen für die folgenden Fragen.

1. Die Namen aller Städte, die sich auf einer Insel befinden (434).
2. Die Namen aller Länder, die keine Hauptstadt haben (2).
3. Die Gesamtzahl der Einwohner aller Länder, die von Großbritannien (ID = 224) abhängig sind.
4. Die Namen, Einwohnerzahl, Fläche und Bevölkerungsdichte (Einwohnerzahl geteilt durch Fläche) aller Staaten, die von einem anderen Land abhängig sind. Sortieren Sie das Ergebnis absteigend nach der Bevölkerungsdichte (38).
5. Die Namen und Einwohnerzahlen aller Städte, die auf dem Äquator liegen (4).
6. Die Gesamtzahl der Einwohner in Städten auf der westlichen Erdhalbkugel.
7. Die Höhe des höchsten Bergs auf der Südhalbkugel.
8. Die Daten aller Flüsse, die den Äquator kreuzen. Sie erkennen dies (zugegeben etwas ungenau) daran, dass die Breitengrade (latitude) von Quelle (source) und Mündung (estuary) unterschiedliche Vorzeichen haben (14).
9. Die Namen aller Städte, für die weder die Einwohnerzahl, noch die geographische Lage eingetragen ist (364).

Die Zahlen in Klammer bedeuten, wie viele Datensätze die Abfrage ergeben sollte.

LEFT JOINS

Wenn zwei Tabellen zusammengeführt werden sollen und es in einer Tabelle Datensätze ohne Gegenstück in der anderen Tabelle gibt, werden diese bei einem JOIN nicht ausgegeben:

LNr	LNachname	LVorname	LDienstbez
1	Winzig	Willi	OStR
2	Meier	Anna	StR'in
3	Lohse	Heinrich	StD

Tabelle „lehrer“

**SELECT * FROM lehrer
JOIN schueler ON LNr = SKLNr
ORDER BY LNachname, LVorname**

Anna Meier (LNr = 2) ist keine Klassenlehrerin, ihr Primärschlüssel taucht in der Spalte „SKLNr“ nicht auf.

SNr	SKLNr	SNachname	SVorname	SKlasse
1	1	Bauer	Theo	8
2	3	Huber	Sara	12
3	1	Köhler	Karolina	8

Tabelle „schueler“

LNr	LNachname	LVorname	LDienstbez	SNr	SKLNr	SNachname	SVorname	SKlasse
3	Lohse	Heinrich	StD	2	3	Huber	Sara	12
1	Winzig	Willi	OStR	1	1	Bauer	Theo	8
1	Winzig	Willi	OStR	3	1	Köhler	Karolina	8

Es gibt allerdings Situationen, in denen man dennoch alle Datensätze aus der ersten Tabelle haben will. Hier kommt der LEFT JOIN zum Einsatz:

**SELECT * FROM lehrer LEFT JOIN schueler ON LNr = SKLNr
ORDER BY LNachname, LVorname**

LNr	LNachname	LVorname	LDienstbez	SNr	SKLNr	SNachname	SVorname	SKlasse
3	Lohse	Heinrich	StD	2	3	Huber	Sara	12
2	Meier	Anna	StR'in	NULL	NULL	NULL	NULL	NULL
1	Winzig	Willi	OStR	1	1	Bauer	Theo	8
1	Winzig	Willi	OStR	3	1	Köhler	Karolina	8

Datensätze aus der ersten („linken“) Tabelle, die kein Gegenstück in der zweiten („rechten“) Tabelle haben, werden ausgegeben. In den entsprechenden Spalten der rechten Tabelle stehen NULL-Werte.

Aufgabe:

Verwenden Sie wieder die Mondial-Datenbank aus der Datei T:\Schule\datenbanken\Mondial.sql

Die Datei T:\Schule\datenbanken\Mondial.html enthält Informationen über die Struktur der Datenbank, insbesondere darüber, welche Fremdschlüssel in welche Tabellen zeigen.

Schreiben Sie SQL-Abfragen, um die folgenden Informationen zu erhalten. Die Zahl in Klammer gibt die zu erwartende Anzahl der Ergebnisdatensätze an.

1. Die Namen aller Berge auf der Nordhalbkugel, ihre Höhe sowie den Namen des Gebirges (sofern zutreffend). Sortieren Sie das Ergebnis absteigend nach der Höhe des Berges (183).
2. Die Namen aller Inseln mit dem Namen der eventuell zugehörigen Inselgruppe sowie des Sees, in dem sie ggf. liegen. Sortieren Sie das Ergebnis zuerst nach dem Namen des Sees, dann nach dem der Inselgruppe und dann nach dem Namen der Insel (276).
3. Den Namen aller Länder, den ihrer Hauptstädte und den der Provinz, in der sich die Hauptstadt befindet. Sortieren Sie das Ergebnis nach dem Namen der Hauptstadt (238).
4. Geben Sie für alle Flüsse, die länger als 1000 km sind, den Namen, die Länge sowie den Namen des Sees bzw. des Meeres, in den/das der Fluss mündet. Sortieren Sie das Ergebnis absteigend zuerst nach dem Namen des Meeres und dann nach dem Namen des Sees (88).
5. Ein Fluss kann auch in einen anderen Fluss münden. Erweitern Sie die letzte Abfrage so, dass für jeden Fluss auch angezeigt wird, in welchen anderen Fluss er mündet (88).

Hierzu benötigen Sie einen sogenannten Self-JOIN. Eine Erklärung finden Sie unter <http://bit.ly/2bPhbzU> bzw.

https://de.wikibooks.org/wiki/Einf%C3%BChrung_in_SQL:_Mehr_zu_JOIN

Komplexe Abfragen auf der Mondial-Datenbank

Verwenden Sie wieder die Mondial-Datenbank unter `T:\Schule\datenbanken\Mondial.sql`. Schreiben Sie SQL-Anfragen, um die folgenden Fragen zu beantworten. Die Klammer hinter der Frage gibt die Anzahl der zu erwartenden Datensätze an.

1. Was sind die Namen aller Städte Griechenlands („Greece“)? Sortieren Sie die Liste alphabetisch (16).
2. Geben sie die Namen und geographischen Koordinaten aller Städte aus, die an der Elbe liegen. Sortieren Sie das Ergebnis von Süden nach Norden (3).
3. In welchen Ländern Asiens („Asia“) beträgt der Anteil der Muslime („Muslim“) über 50%? Geben Sie den Namen des Landes und den Anteil aus, sortiert absteigend nach dem Anteil und aufsteigend nach dem Ländername (25).
4. Wie heißt der höchste Berg Amerikas („America“)?
5. Wie heißt das Land, das die längste gemeinsame Grenze mit Russland (ID 176) hat?
6. Welche Wüsten befinden sich in den USA (ID 225) und welche Fläche haben diese jeweils? Sortieren Sie das Ergebnis nach dem Flächeninhalt (7). Achten Sie darauf, dass jede Wüste wirklich nur einmal im Ergebnis auftaucht.
7. In welchen Ländern liegt die Kalahari-Wüste? (5)
8. Wie lang ist die Außengrenze von Deutschland (ID 77)? Sie sollten auf eine Gesamtlänge von 3621 km kommen.
9. In welchem Land werden die meisten Sprachen gesprochen und wieviele sind es? Sie sollten auf 8 Sprachen kommen.
10. Wie viele Protestanten („Protestant“) gibt es auf der Welt? Sie sollten auf ca. 295 Millionen kommen.
11. Welcher Fluss fließt durch die meisten Länder (10 Länder)?
12. Wir wollen die Annahme überprüfen, ob die Städte in größeren Ländern auch mehr Einwohner haben. Also berechnen wir für die einzelnen Länder die durchschnittlichen Einwohnerzahlen der Städte. Wir dividieren diesen Wert durch die Einwohnerzahl des Landes (wir nennen das Ergebnis „Hilfswert“) und lassen uns diese Liste (Land, Einwohnerzahl, durchschnittliche Einwohnerzahl in den Städten, Hilfswert) absteigend nach Hilfswert ausgeben. (238)

Ein praktisches Sprachmittel in SQL ist der `IN`-Operator. Damit kann man überprüfen, ob ein Wert in einer Liste von Werten enthalten ist.

Beispiel:

```
SELECT * FROM Province WHERE Country IN (13,77,206)
```

Diese Abfrage liefert alle Provinzen (Bundesländer und Kantone) in Deutschland (77), Österreich (13) und der Schweiz (206).

Die Liste kann auch Zeichenketten enthalten:

```
SELECT * FROM City WHERE name IN ('Berlin','Stuttgart','Munich')
```

Dies liefert die Daten zu den Städten Berlin, Stuttgart und München.

Der `IN`-Operator kann mit `NOT` auch umgedreht werden:

```
SELECT * FROM City WHERE id NOT IN ('Berlin','Stuttgart','Munich')
```

Dies liefert die Daten zu allen Städte, *außer* Berlin, Stuttgart und München.

Interessant wird der `IN`-Operator aber erst mit *Subqueries* („Unterabfragen“). Man kann abfragen, ob ein Wert sich in der Ergebnisliste einer anderen Abfrage befindet.

Beispiel:

```
SELECT * FROM City WHERE Province IN  
(SELECT id FROM Province WHERE Country = 77)
```

Hier wird zuerst die Subquery in der Klammer ausgewertet und damit die `ids` aller Bundesländer in Deutschland bestimmt. In der äußeren Abfrage werden dann alle Städte gesucht, deren `Province`-Wert irgendeinem der gefundenen Werte entspricht.

Subqueries können beliebig geschachtelt werden:

```
SELECT * FROM City WHERE Province IN  
  (SELECT id FROM Province WHERE Country IN  
    (SELECT Country FROM encompasses WHERE Continent IN  
      (SELECT id FROM Continent WHERE name = 'America''))))
```

Damit werden z.B. alle Städte in Amerika gesucht.

Eine Subquery in Verbindung mit dem `IN`-Operator darf stets nur eine Spalte zurückgeben. Wenn eine Subquery keinen Datensatz zurückliefert, ergibt der `IN`-Vergleich natürlich immer falsch und die äußere Abfrage liefert ebenfalls keine Datensätze.

Verwenden Sie den `IN`-Operator für die folgenden Fragen:

13. Lösen Sie die Fragen 6 und 7 nochmals mit dem `IN`-Operator.
14. Berechnen Sie die Gesamtlänge aller Flüsse, die durch Deutschland fließen.
15. Geben Sie alle Binnenländer (d.h. Länder ohne direkten Zugang zum Meer) aus, sortiert nach dem Namen der Länder (45). Hilfe: Suchen Sie zunächst alle Länder, die Zugang zum Meer haben und kehren Sie Ihre Auswahl dann um.
16. Geben Sie alle Länder ohne Berge aus (144).