SQL - Beispiele

Studenten

MatrikelNr	Name	Vorname	Vorname2	Geburt	Ort	SgNr	Bafoeg
1001	Schmidt	Hans	Peter	24.2.1990	Nürnberg	2	200
1002	Meisel	Dirk	Helmut	17.8.1989	Fürth	3	500
1003	Schmidt	Amelie		19.9.1992	Wendelstein	1	0
1004	Krause	Christian	Johannes	3.5.1990	Nürnberg	1	100
1005	Schäfer	Julia		30.3.1993	Erlangen	5	0
1006	Rasch	Lara		30.3.1992	Nürnberg	3	0
1007	Bakowski	Juri		15.7.1988	Fürth	4	400

Studiengaenge

SgNr	Kuerzel	Name	Fak
1	IN	Informatik	IN
2	WIN	Wirtschaftsinformatik	IN
3	MIN	Medieninformatik	IN
4	BW	Betriebswirtschaftslehre	BW
5	ET	Elektrotechnik	EFI

Projektion (Spaltenauswahl) und Selektion

$\pi_{Name,Vorname,Ort}(\sigma_{Name='Schmidt'}(Studenten))$

```
SELECT Name, Vorname, Ort
FROM Studenten
WHERE Name='Schmidt'
ORDER BY Ort, Name, Vorname
```

Sortierung (ORDER BY) gibt es in der relationalen Algebra nicht.
Schlüsselwort DISTINCT nach SELECT eliminiert Duplikate.

```
SELECT DISTINCT Name
FROM Studenten
```

Projektion mit Funktionen

Für einen Übersicht der Funtionen in MySQL siehe:
http://dev.mysql.com/doc/refman/5.7/en/func-op-summary-ref.html

```
SELECT
  COALESCE(Vorname2, 'kein 2. Vorname'), -- Nullwert-Behandlung
  CONCAT(Vorname, ' ', Name) AS Name, -- SQL Standard
  Vorname + ' ' + Name AS Name2, -- nur MS SQL Server
  YEAR(GebDat) AS GebJahr,
  TIMESTAMPDIFF(YEAR, GebDat, CURRENT_DATE) AS AlterJahre,
  UPPER(Name) AS NameGross,
  CASE WHEN LENGTH(Vorname) > 5 THEN 'lang' ELSE 'kurz' END AS Länge
FROM Studenten
```

Selektionen mit komplexen Bedingungen

```
WHERE Bafoeg + 100 < 1000 AND Bafoeg != 0

WHERE Bafoeg BETWEEN 100 AND 500

WHERE SgNr IN (1, 2, 3)

WHERE Name LIKE 'M%' OR Name NOT LIKE '%M'

WHERE GebDat > '1990-01-01'

WHERE Vorname2 IS NULL;
```

Joins

Inner Join

$\pi_{s.Name,\ sg.Fak}$ (Studenten $\bowtie_{Studenten.SgNr=Studiengaenge.SgNr}$ Studiengaenge)

```
SELECT s.Name, sg.Fak
FROM Studenten s JOIN Studiengang sg ON s.SgNr = sg.SgNr
```

Äquivalente Version über kartesisches Produkt:
$\pi_{s.Name,\ sg.Fak}$ ($\sigma_{Studenten.SgNr=Studiengaenge.SgNr}$(Studenten x Studiengaenge))

```
SELECT s.Name, sg.Fak
FROM Studenten s, Studiengaenge sg
WHERE s.SgNr = sg.SgNr
```

Outer Join

```
SELECT s.Name, sg.Fak
FROM Studenten s RIGHT JOIN Studiengang sg ON s.SgNr = sg.SgNr
```

Right Join gibt alle Datensätze der rechten Tabelle aus, auch wenn kein passender Datensatz in linker Tabelle vorhanden. Bei Left Join ist es umgekehrt.

Tips zu Joins: Inner Joins sind performanter zu berechnen und meist ausreichend. Ein Outer Join wird nur benötigt, wenn explizit auch z.B. Studiengänge ohne Studenten ausgegeben werden sollen.

Aggregation

Anzahl der Studenten pro Ort:

```
SELECT Ort, COUNT(*)
FROM Studenten
GROUP BY Ort
```

Gesamtsumme des Bafög und Anzahl der verschiedenen Wohnorte pro Studiengang für alle Studiengänge mit mindestens 2 Studenten:

```
SELECT
  sg.Name, sg.Fak, SUM(s.Bafoeg), COUNT(DISTINCT s.Ort)
FROM Studenten s JOIN Studiengang sg ON s.SgNr = sg.SgNr
GROUP BY sg.Name, sg.Fak
HAVING COUNT(*) >= 2
```

Tips zu Aggregationsanfragen:

- Alle genannten Spalten in der SELECT-Klausel müssen entweder in einer Aggregatfunktion verwendet oder in die GROUP-BY-Klausel übernommen werden.
- In der GROUP-BY-Klausel tauchen meistens Attribute wie IDs, Namen, Bezeichnungen etc. auf. Numerische Attribute wie Mengen, Preise, Anzahlen usw. werden üblicherweise mit einer Aggregatfunktion wie SUM verrechnet. D.h. falls Sie in einer Anfrage "GROUP BY Menge" o.ä. stehen haben, ist wahrscheinlich etwas falsch.

Insert, Update, Delete

```
INSERT INTO Studiengaenge(SgNr, Kuerzel, Name, Fak)
VALUES (1, 'IN','Informatik','Inf')

UPDATE Studenten
SET Ort = 'Berlin', Bafoeg = 0
WHERE Name = 'Meier' AND Vorname = 'Hans'

DELETE
FROM Studenten
WHERE MatrikelNr = 1235
```

Unterabfragen

Unterabfragen in der WHERE-Klausel

Alle Studiengänge ohne Studenten:

```
SELECT *
FROM Studiengaenge
WHERE SgNr NOT IN (SELECT SgNr FROM Studenten)
```

Alle Studiengänge mit Studenten

```
SELECT *
FROM Studiengaenge sg
WHERE EXISTS (SELECT * FROM Studenten s WHERE s.SgNr = sg.SgNr)
```

Unterabfragen in der FROM-Klausel

Können wie eigene Tabellen verwendet werden:

```
SELECT MAX(Gesamtbafoeg_pro_SG)
FROM (
  SELECT SgNr, SUM(Baefoeg) AS Gesamtbafoeg_pro_SG)
  FROM Studenten
  GROUP BY SgNr
) AS tmp
```

Mengenoperationen

Für Mengenoperationen müssen die beiden eingehenden Tabellen (gebildet durch entsprechende SELECT-Anweisungen) die gleiche Anzahl von Spalten mit kompatiblen Datentypen haben.

Vereinigung

Alle Sätze aus beiden Tabellen.

```
SELECT Name, Vorname, Email
FROM Leser
UNION
SELECT Name, Vorname, Email
FROM Autoren
```

UNION eliminiert automatisch Duplikate. UNION ALL behält Duplikate und ist daher aus Performance-Aspekten vorzuziehen, solange eine Duplikat-Elminierung nicht erforderlich ist.

Durchschnitt

Alle Sätze, die sowohl in der ersten als auch in der zweiten Tabelle vorkommen, Duplikate werden eliminiert.

```
SELECT Name, Vorname, Email
FROM Leser
INTERSECT
SELECT Name, Vorname, Email
FROM Autoren
```

Differenz

Nur die Sätze aus der ersten Tabelle, die nicht in der zweiten enthalten sind, Duplikate werden eliminiert.

```
SELECT Name, Vorname, Email
FROM Leser
EXCEPT     -- bei den meisten DBS MINUS
SELECT Name, Vorname, Email
FROM Autoren
```