SQL-Grundlagen Spickzettel

LearnSQL.de

SQL (*Structured Query Language*) ist eine Sprache zur Kommunikation mit Datenbanken. Sie ermöglicht es, bestimmte Daten auszuwählen und komplexe Berichte zu erstellen. Heutzutage ist SQL eine universelle Sprache der Daten. Sie wird in praktisch allen Technologien verwendet, die Daten verarbeiten.

BEISPIELDATEN

LAND

Id	Name	Einwohner	Flaeche
1	Frankreich	66600000	640680
2	Deutschland	80700000	357000
...	...	...	...

STADT

Id	Name	Land_Id	Einwohner	Bewertung
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

ABFRAGE EINER EINZELNEN TABELLE

Zeige alle Spalten aus der Tabelle Land an:

```
SELECT *
FROM Land;
```

Abrufen der Spalten Id und Name aus der Tabelle Stadt:

```
SELECT Id, Name
FROM Stadt;
```

Abrufen von Städtenamen sortiert nach der Spalte Bewertung in der Standardreihenfolge ASCending (aufsteigend):

```
SELECT Name
FROM Stadt
ORDER BY Bewertung [ASC];
```

Abrufen von Städtenamen sortiert nach der Spalte Bewertung in der Reihenfolge DESCending (absteigend):

```
SELECT Name
FROM Stadt
ORDER BY Bewertung DESC;
```

ALIASES

SPALTEN

```
SELECT Name AS Stadt_Name
FROM Stadt;
```

TABELLEN

```
SELECT La.Name, St.Name
FROM Stadt AS St
JOIN Land AS La
  ON St.Land_Id = La.Id;
```

FILTERN DER AUSGABE

VERGLEICHSOPERATOREN

Holt die Namen von Städten, die eine Bewertung über 3 haben:

```
SELECT Name
FROM Stadt
WHERE Bewertung > 3;
```

Holt die Namen von Städten, die weder Berlin noch Madrid sind:

```
SELECT Name
FROM Stadt
WHERE Name != 'Berlin'
  AND Name != 'Madrid';
```

TEXT-OPERATOREN

Sucht Namen von Städten, die mit einem 'P' beginnen oder mit einem 's' enden:

```
SELECT Name
FROM Stadt
WHERE Name LIKE 'P%'
  OR Name LIKE '%s';
```

Sucht Namen von Städten, die mit einem beliebigen Buchstaben beginnen, gefolgt von "ublin" (wie Dublin in Irland oder Lublin in Polen):

```
SELECT Name
FROM Stadt
WHERE Name LIKE '_ublin';
```

ANDERE OPERATOREN

Holt die Namen von Städten mit einer Einwohnerzahl zwischen 500K und 5M:

```
SELECT Name
FROM Stadt
WHERE Einwohner BETWEEN 500000 AND 5000000;
```

Holt die Namen von Städten, die keinen fehlenden Bewertungswert haben:

```
SELECT Name
FROM Stadt
WHERE Bewertung IS NOT NULL;
```

Holt Namen von Städten, die in Ländern mit den IDs 1, 4, 7 oder 8 liegen:

```
SELECT Name
FROM Stadt
WHERE Land_Id IN (1, 4, 7, 8);
```

ABFRAGE MEHRERER TABELLEN

INNER JOIN

JOIN (oder explizit **INNER JOIN**) gibt Zeilen zurück, die übereinstimmende Werte in beiden Tabellen haben.

```
SELECT Stadt.Name, Land.Name
FROM Stadt
[INNER] JOIN Land
  ON Stadt.Land_Id = Land.Id;
```

STADT			LAND	
Id	Name	Land_Id	Id	Name
1	Paris	1	1	Frankreich
2	Berlin	2	2	Deutschland
3	Warschau	4	3	Island

LEFT JOIN

LEFT JOIN gibt alle Zeilen aus der linken Tabelle mit den entsprechenden Zeilen aus der rechten Tabelle zurück. Wenn es keine übereinstimmende Zeile gibt, werden **NULL**s als Werte aus der zweiten Tabelle zurückgegeben.

```
SELECT Stadt.Name, Land.Name
FROM Stadt
LEFT JOIN Land
  ON Stadt.Land_Id = Land.Id;
```

STADT			LAND	
Id	Name	Land_Id	Id	Name
1	Paris	1	1	Frankreich
2	Berlin	2	2	Deutschland
3	Warschau	4	NULL	NULL

RIGHT JOIN

RIGHT JOIN gibt alle Zeilen aus der rechten Tabelle mit den entsprechenden Zeilen aus der linken Tabelle zurück. Wenn es keine übereinstimmende Zeile gibt, werden **NULL**s als Werte aus der linken Tabelle zurückgegeben.

```
SELECT Stadt.Name, Land.Name
FROM Stadt
RIGHT JOIN Land
  ON Stadt.Land_Id = Land.Id;
```

STADT			LAND	
Id	Name	Land_Id	Id	Name
1	Paris	1	1	Frankreich
2	Berlin	2	2	Deutschland
NULL	NULL	NULL	3	Island

FULL JOIN

FULL JOIN (oder explizit **FULL OUTER JOIN**) gibt alle Zeilen aus beiden Tabellen zurück – wenn es keine passende Zeile in der zweiten Tabelle gibt, werden **NULL**s als Werte aus dieser Tabelle zurückgegeben.

```
SELECT Stadt.Name, Land.Name
FROM Stadt
FULL [OUTER] JOIN Land
  ON Stadt.Land_Id = Land.Id;
```

STADT			LAND	
Id	Name	Land_Id	Id	Name
1	Paris	1	1	Frankreich
2	Berlin	2	2	Deutschland
3	Warschau	4	NULL	NULL
NULL	NULL	NULL	3	Island

CROSS JOIN

CROSS JOIN gibt alle möglichen Kombinationen von Zeilen aus beiden Tabellen zurück. Es sind zwei Syntaxen verfügbar.

```
SELECT Stadt.Name, Land.Name
FROM Stadt
CROSS JOIN Land;
```

```
SELECT Stadt.Name, Land.Name
FROM Stadt, Land;
```

STADT			LAND	
Id	Name	Land_Id	Id	Name
1	Paris	1	1	Frankreich
1	Paris	1	2	Deutschland
2	Berlin	2	1	Frankreich
2	Berlin	2	2	Deutschland

NATURAL JOIN

NATURAL JOIN verbindet Tabellen durch alle Spalten mit demselben Namen.

```
SELECT Stadt.Name, Land.Name
FROM Stadt
NATURAL JOIN Land;
```

STADT			LAND	
Land_Id	Id	Name	Name	Id
6	6	San Marino	San Marino	6
7	7	Vatikanstadt	Vatikanstadt	7
5	9	Griechenland	Griechenland	9
10	11	Monaco	Monaco	10

NATURAL JOIN verwendet diese Spalten, um Zeilen abzugleichen:
Stadt.Id, **Stadt.Name**, **Land.Id**, **Land.Name**.
NATURAL JOIN wird in der Praxis sehr selten verwendet.

SQL-Grundlagen Spickzettel

LearnSQL.de

AGGREGATION UND GRUPPIERUNG

GROUP BY **fasst** Zeilen zusammen, die in bestimmten Spalten die gleichen Werte aufweisen. Es werden Zusammenfassungen (Aggregate) für jede eindeutige Kombination von Werten berechnet.

STADT		
Id	**Name**	**Land_Id**
1	Paris	1
101	Marseille	1
102	Lyon	1
2	Berlin	2
103	Hamburg	2
104	München	2
3	Warschau	4
105	Krakau	4

→

STADT	
Land_Id	**Anzahl**
1	3
2	3
4	2

AGGREGAT-FUNKTIONEN

- `avg(expr)` – Durchschnittswert für Zeilen innerhalb der Gruppe
- `count(expr)` – Anzahl der Werte für Zeilen innerhalb der Gruppe
- `max(expr)` – Maximalwert innerhalb der Gruppe
- `min(expr)` – Mindestwert innerhalb der Gruppe
- `sum(expr)` – Summe der Werte innerhalb der Gruppe

BEISPIELABFRAGEN

Ermittle die Anzahl der Städte:

```
SELECT COUNT(*)
FROM Stadt;
```

Ermittle die Anzahl der Städte mit Nicht-Null-Bewertungen:

```
SELECT COUNT(Bewertung)
FROM Stadt;
```

Ermitteln der Anzahl der eindeutigen Länderwerte:

```
SELECT COUNT(DISTINCT Land_Id)
FROM Stadt;
```

Ermittle die kleinste und die größte Länderbevölkerung:

```
SELECT MIN(Einwohner), MAX(Einwohner)
FROM Land;
```

Ermittle die Gesamtbevölkerung der Städte in den jeweiligen Ländern:

```
SELECT Land_Id, SUM(Einwohner)
FROM Stadt
GROUP BY Land_Id;
```

Ermittle die durchschnittliche Bewertung der Städte in den jeweiligen Ländern, wenn der Durchschnitt über 3,0 liegt:

```
SELECT Land_Id, AVG(Bewertung)
FROM Stadt
GROUP BY Land_Id
HAVING AVG(Bewertung) > 3.0;
```

UNTERABFRAGEN

Eine Unterabfrage ist eine Abfrage, die in einer anderen Abfrage oder in einer anderen Unterabfrage verschachtelt ist. Es gibt verschiedene Arten von Unterabfragen.

EINZELNER WERT

Die einfachste Unterabfrage gibt genau eine Spalte und genau eine Zeile zurück. Sie kann mit den Vergleichsoperatoren =, <, <=, >, oder >= verwendet werden.
Diese Abfrage findet Städte mit der gleichen Bewertung wie Paris:

```
SELECT Name
FROM Stadt
WHERE Bewertung = (
  SELECT Bewertung
  FROM Stadt
  WHERE Name = 'Paris'
);
```

MEHRERE WERTE

Eine Unterabfrage kann auch mehrere Spalten oder mehrere Zeilen zurückgeben. Solche Unterabfragen können mit den Operatoren `IN`, `EXISTS`, `ALL`, oder `ANY` verwendet werden.
Diese Abfrage findet Städte in Ländern, die mehr als 20 Millionen Einwohner haben:

```
SELECT Name
FROM Stadt
WHERE Land_Id IN (
  SELECT Land_Id
  FROM Land
  WHERE Einwohner > 20000000
);
```

KORRELIERTE UNTERABFRAGEN

Eine korrelierte Unterabfrage bezieht sich auf die in der äußeren Abfrage eingeführten Tabellen. Eine korrelierte Unterabfrage hängt von der äußeren Abfrage ab. Sie kann nicht unabhängig von der äußeren Abfrage ausgeführt werden.
Diese Abfrage findet Städte mit einer Bevölkerung, die größer als die durchschnittliche Bevölkerung des Landes ist:

```
SELECT *
FROM Stadt Grosse_Stadt
WHERE Einwohner > (
  SELECT AVG(Einwohner)
  FROM Stadt Durchschnitt_Stadt
  WHERE Durchschnitt_Stadt.Land_Id
        = Grosse_Stadt.Land_Id
);
```

Diese Abfrage findet Länder, die mindestens eine Stadt haben:

```
SELECT Name
FROM Land
WHERE EXISTS (
  SELECT *
  FROM Stadt
  WHERE Land_Id = Land.Id
);
```

SET-OPERATIONEN

Mengenoperationen werden verwendet, um die Ergebnisse von zwei oder mehr Abfragen zu einem einzigen Ergebnis zu kombinieren. Die kombinierten Abfragen müssen die gleiche Anzahl von Spalten und kompatible Datentypen zurückgeben. Die Namen der entsprechenden Spalten können unterschiedlich sein.

RADSPORT		
Id	**Name**	**Land**
1	YK	DE
2	ZG	DE
3	WT	PL
...	...	...

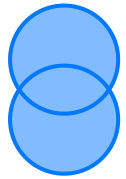
EISSCHNELLLAUF		
Id	**Name**	**Land**
1	YK	DE
2	DF	DE
3	AK	PL
...	...	...

UNION

`UNION` kombiniert die Ergebnisse von zwei Ergebnismengen und entfernt Duplikate.
`UNION ALL` entfernt keine doppelten Zeilen.

Diese Abfrage zeigt deutsche Radfahrer zusammen mit deutschen Eisläufer an:

```
SELECT Name
FROM Radsport
WHERE Land = 'DE'
UNION / UNION ALL
SELECT Name
FROM Eisschnelllauf
WHERE Land = 'DE';
```

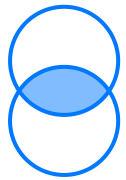


INTERSECT

`INTERSECT` gibt nur Zeilen zurück, die in beiden Ergebnismengen vorkommen.

Diese Abfrage zeigt deutsche Radfahrer an, die gleichzeitig auch deutsche Eisläufer sind:

```
SELECT Name
FROM Radsport
WHERE Land = 'DE'
INTERSECT
SELECT Name
FROM Eisschnelllauf
WHERE Land = 'DE';
```



EXCEPT

`EXCEPT` gibt nur die Zeilen zurück, die in der ersten Ergebnismenge erscheinen, aber nicht in der zweiten Ergebnismenge.

Diese Abfrage zeigt deutsche Radsportler an, es sei denn, sie sind gleichzeitig auch deutsche Eisläufer:

```
SELECT Name
FROM Radsport
WHERE Land = 'DE'
EXCEPT / MINUS
SELECT Name
FROM Eisschnelllauf
WHERE Land = 'DE';
```