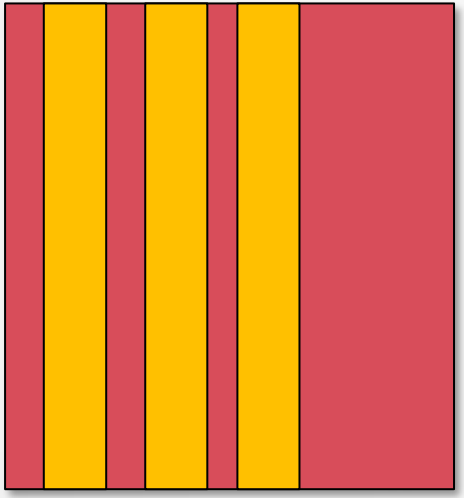
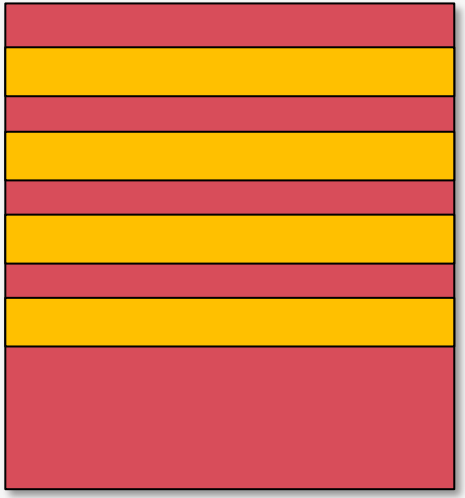
SQL

Übersicht

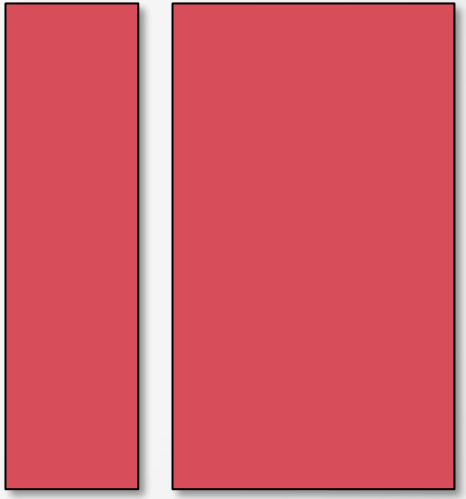
Projektion:



Selektion:



Verbund:



SELECT: Syntaxübersicht

```
SELECT [Auswahl]
FROM [Tabelle]
WHERE [Bedingung]
GROUP BY [Gruppierungsbedingung]
HAVING [Filterbedingung]
ORDER BY [Feldliste]
```

SELECT: Projektion - Wahl der Spalten

Beschreibung:
Gibt die Spalten an, die als Ergebnis angezeigt werden

Gibt alle Spalten (aller Datensätze) der Tabelle Personal zurück
SELECT ALL FROM Personal;
SELECT * FROM Personal;

Gibt alle Spalten Nachname, Vorname (aller Datensätze) der Tabelle Personal zurück
SELECT Nachname, Vorname FROM Personal

SELECT: Anzahl der Datensätze

Beschreibung:
Beschränkt die Anzahl der Datensätze, die als Ergebnis angezeigt werden

Gibt die ersten 25 Namen der Studenten der Tabelle Personal zurück
SELECT Vorname, Nachname FROM Studenten LIMIT 25;
Gibt die ersten 25 Namen der Studenten der Tabelle Personal zurück
SELECT Vorname, Nachname FROM Studenten ORDER BY punkte DESC LIMIT 25;

SELECT: WHERE I

Beschreibung:
Gibt an, welche Datensätze der im FROM-Abschnitt aufgeführten Tabellen von einer SELECT-Anweisung betroffen sind.

Die Felder Nachname und Vorname aller Datensätz, in denen das Feld Nachname den Wert King hat:

```
SELECT Nachname, Vorname FROM Personal WHERE Nachname = 'King';
```

Die Felder Nachname und Vorname für alle Angestellten, deren Nachname mit einem "S" beginnt

```
SELECT Nachname, Vorname FROM Personal WHERE Nachname Like 'S*';
```

Alle Artikel mit einem Einzelpreis zwischen 30 und 75 aus

```
SELECT Artikelname, Einzelpreis FROM Artikel WHERE (Einzelpreis >=30.00 And Einzelpreis <= 75.00);
```

SELECT: WHERE II

Beschreibung:
Gibt an, welche Datensätze der im FROM-Abschnitt aufgeführten Tabellen von einer SELECT-Anweisung betroffen sind.

Alle Artikel aus, deren Artikelnamen zwischen "Cha" and "Out":
SELECT Artikelname, Einzelpreis FROM Artikel WHERE Artikelname Between 'Cha' And 'Out';

Alle Bestellungen, die im ersten Halbjahr 2020 eingegangen sind:
SELECT [Bestell-Nr], Bestelldatum FROM Bestellungen WHERE Bestelldatum Between #1-1-2020# And #6-30-2020#;

Alle Bestellungen, aus den Regionen Idaho, Oregon oder Washington :
SELECT [Bestell-Nr], Region FROM Bestellungen WHERE Region In ('ID', 'OR', 'WA');

SELECT: ORDER BY

Beschreibung:
Gibt die Sortierung der Datensätze an!

Sortiert die Namen der Angestellten nach ihren Nachnamen
SELECT Nachname, Vorname FROM Personal ORDER BY Nachname;
SELECT Nachname, Vorname FROM Personal ORDER BY Nachname ASC;

Sortiert die Gehälter in absteigender Reihenfolge:
SELECT Nachname, Gehalt FROM Personal ORDER BY Gehalt DESC, Nachname;

ORDER ist der letzte Eintrag in einer einfachen SQL-Anweisung
Mehrere Felder können angegeben werden, dann werden alle Datensätze, die im ersten Feld denselben Wert haben, nach dem zweiten Wert sortiert!

TOOL: SqlFormat.org

Type your SQL here:

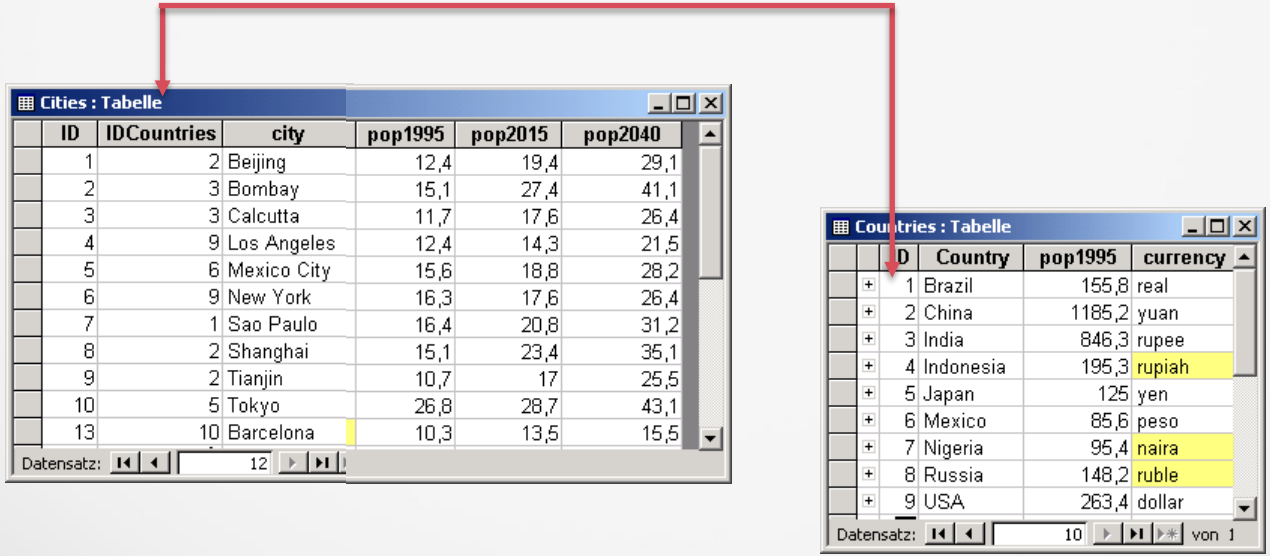
```
SELECT [Bestell-Nr], Region FROM Bestellungen WHERE Region In ('ID',
'OR', 'WA');
```

```
SELECT [Bestell-Nr],
       Region
FROM Bestellungen
WHERE Region In ('ID',
                 'OR',
                 'WA');
```

JOIN: JOIN I

Beschreibung:

Führt mehrere Tabellen zu einer virtuellen Tabelle zusammen. Die Verbindung ist unabhängig von einer Verknüpfung der Tabellen, erfolgt aber über zwei Schlüssel; in der Regel ein Primär- und ein Fremdschlüssel!



Cities : Tabelle

ID	IDCountries	city	pop1995	pop2015	pop2040
1	2	Beijing	12,4	19,4	29,1
2	3	Bombay	15,1	27,4	41,1
3	3	Calcutta	11,7	17,6	26,4
4	9	Los Angeles	12,4	14,3	21,5
5	6	Mexico City	15,6	18,8	28,2
6	9	New York	16,3	17,6	26,4
7	1	Sao Paulo	16,4	20,8	31,2
8	2	Shanghai	15,1	23,4	35,1
9	2	Tianjin	10,7	17	25,5
10	5	Tokyo	26,8	28,7	43,1
13	10	Barcelona	10,3	13,5	15,5

Datensatz: 12

Countries : Tabelle

ID	Country	pop1995	currency
1	Brazil	155,8	real
2	China	1185,2	yuan
3	India	846,3	rupee
4	Indonesia	195,3	rupiah
5	Japan	125	yen
6	Mexico	85,6	peso
7	Nigeria	95,4	naira
8	Russia	148,2	ruble
9	USA	263,4	dollar

Datensatz: 10 von 1

JOIN : JOIN II

1: Beinhaltet nur die Datensätze, bei denen die Inhalte der verknüpften Felder beider Tabellen gleich sind.

```
SELECT City, currency
FROM Cities
INNER JOIN Countries ON Cities.IDCountries = Countries.ID
```

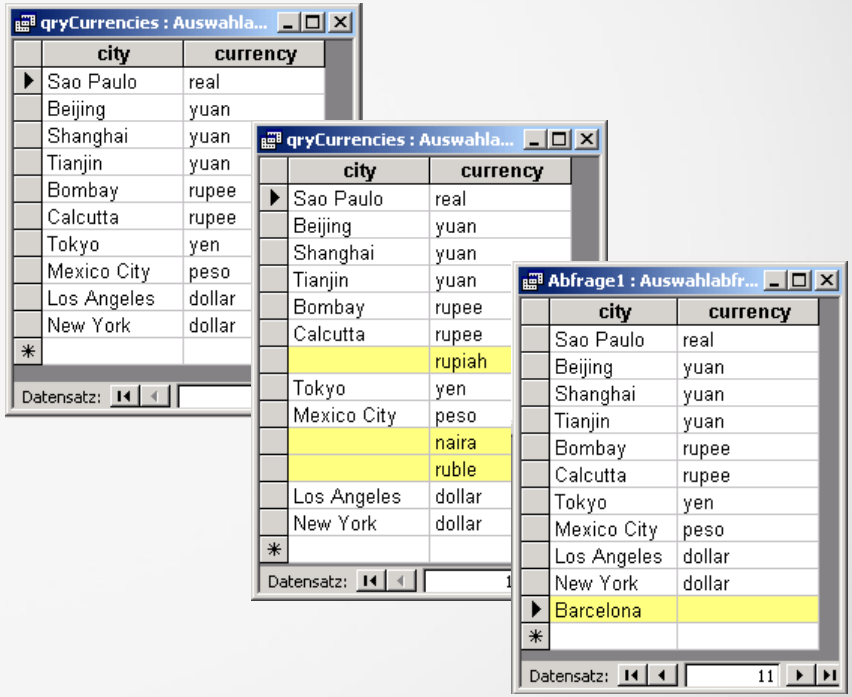
```
SELECT Cities.City, Countries.currency
FROM Cities
INNER JOIN Countries ON Cities.IDCountries = Countries.ID
```

2: Beinhaltet ALLE Datensätze aus 'Countries' und nur die Datensätze aus 'Cities', bei denen die Inhalte der verknüpften Felder beider Tabellen gleich sind.

```
SELECT City, currency
FROM Cities
RIGHT JOIN Countries ON Cities.IDCountries = Countries.ID
```

3: Beinhaltet ALLE Datensätze aus 'Cities' und nur die Datensätze aus 'Countries', bei denen die Inhalte der verknüpften Felder beider Tabellen gleich sind.

```
SELECT City, currency
FROM Cities
LEFT JOIN Countries ON Cities.IDCountries = Countries.ID
```



GROUP BY - HAVING: GROUP BY

Beschreibung:
Fasst Datensätze, die in der angegebenen Feldliste dieselben Werte enthalten, zu einem einzelnen Datensatz zusammen.

Für jeden Lieferanten wird der durchschnittliche Einzelpreis aller Artikel ermittelt(und nach Lieferanten gruppiert)!

```
SELECT [Lieferanten-Nr], Avg(Einzelpreis) AS DurchschnittEinzelpreis
FROM Artikel GROUP BY [Lieferanten-Nr];
```

GROUP BY - HAVING: GROUP BY II

In jeder Kategorie den wird der höchste Einzelpreis eines Artikels ermittelt!

```
SELECT [Kategorie-Nr], Max(Einzelpreis) AS MaxEinzelpreis
FROM Artikel GROUP BY [Kategorie-Nr];
```

Wie viele Bestellungen sind jedem Angestellten in der Datenbank zugeordnet:

```
SELECT [Personal-Nr], Count([Bestell-Nr]) AS AnzahlBestellungen
FROM Bestellungen GROUP BY [Personal-Nr];
```

GROUP BY - HAVING: HAVING I

Beschreibung:
Filtert die Datensätze für die gruppierten Abschnitte des GROUP BY Befehls

Alle Lieferanten, deren Artikel einen durchschnittlichen Einzelpreis über 40(€) aufweisen:

```
SELECT LieferantenNr, Avg(Einzelpreis) AS DurchschnittEinzelpreis
FROM Artikel GROUP BY LieferantenNr HAVING (Avg(Einzelpreis)>40);
```

Alle Angestellten, die mehr als 100 Bestellungen entgegengenommen haben:

```
SELECT PersonalNr, Count(BestellNr) AS AnzahlBestellungen FROM
Bestellungen GROUP BY PersonalNr HAVING Count(BestellNr) > 100;
```

GROUP BY - HAVING: Funktionen

Mit den Befehlen HAVING und GROUP BY können verschiedene Gruppenfunktionen eingesetzt werden:

AVG(): Durchschnittlicher Wert einer Spalte
MAX(): Maximaler Wert einer Spalte
SUM(): Summe aller Einträge einer Spalte
COUNT(): Durchschnittlicher Wert einer Spalte
MIN(): Minimum einer Spalte

GROUP BY: Funktionen in Kombination mit GROUP BY I

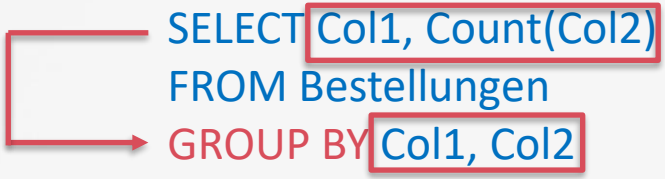
kann

```
SELECT  Amount, Count(Amount * 100)
FROM    Penalties
GROUP BY Amount;
```

Alle Felder aus dem GROUP BY-Teil können auch in dem SELECT-Teil vorkommen(als Felder oder in Ausdrücken)!

GROUP BY: Funktionen in Kombination mit GROUP BY II

muss



Alle Felder aus dem SELECT-Teil müssen auch in dem GROUP BY Teil vorkommen!

Subqueries – Definition und Regeln

- ✓ Eine Unterabfrage kann vorkommen in der
  - ✓ SELECT-Klausel
  - ✓ FROM-Klausel
  - ✓ WHERE-Klausel
- ✓ Möglich ist sie grundsätzlich in SELECT, INSERT, DELETE und UPDATE
- ✓ Fast immer ist es eine SELECT-Abfrage in der WHERE-Klausel einer anderen WHERE Klausel
- ✓ Verwendet werden kann
  - ✓ >, <, =
  - ✓ IN
  - ✓ ANY
  - ✓ ALL
  - ✓ EXISTS

Subqueries – Beispiel

ID	Name	Punkte
1	Peter	95
2	Mike	80
3	Paul	74
4	Laura	81

Aufgabenstellung: Alle Namen aller Kursteilnehmer von der Tabelle Ergebnisse, die mehr Punkte als Mike haben!

1. Schritt: Wie viele Punkte hat Mike(80)?

```
SELECT punkte FROM ergebnisse WHERE Name='Mike';
```

2. Schritt: Welche Kursteilnehmer haben mehr als 80 Punkte(Peter, Laura)?

```
SELECT Name FROM ergebnisse WHERE Punkte>80;
```

Kombination beider Schritte:

```
SELECT Name FROM ergebnisse WHERE Punkte>
        (SELECT punkte FROM ergebnisse WHERE Name='Mike');
```

Subqueries – Beispiel II

ID	Name	Punkte
1	Peter	95
2	Mike	80
3	Paul	74
4	Laura	81

Liste von Einträgen: IN(oder NOT IN)

```
SELECT Punkte
FROM ergebnisse
WHERE Name IN
     (SELECT Name
      FROM ergebnisse
      ORDER BY punkte DESC
      LIMIT 2);
```

Befehlsgruppen in SQL

DDL
Data Definition Language

- ✓ CREATE: Tabellen anlegen
- ✓ DROP: Tabellen löschen
- ✓ ALTER: Tabellenstruktur ändern

DML
Data Manipulation Language

- ✓ SELECT: Datensätze wählen
- ✓ INSERT: Datensätze zufügen
- ✓ UPDATE: Datensätze ändern
- ✓ DELETE: Datensätze löschen

DSL
Data Security Language

- ✓ BEGINTRANS: Transaktion beginnen
- ✓ COMMITTRANS: Transaktion erfolgreich beenden
- ✓ ROLLBACK: Transaktion abbrechen

Üben von SQL – speziel SELECT-Klausel

https://www.w3resource.com/sql-exercises/sql-subqueries-exercises.php

Your Code ...

```
1  create table calc(x int, y int);
2
3  insert into calc values(10, 25);
4
5  select x,y, (x+y) from calc;
6
```

Ergebnis:

```
10|25|35
```

Üben von DDL und DML

https://www.jdoodle.com/execute-sql-online

Your Code ...

```
1  create table calc(x int, y int);
2
3  insert into calc values(10, 25);
4
5  select x,y, (x+y) from calc;
6
```

Ergebnis:

```
10|25|35
```

DDL Befehl: Create

Funktion:

Erstellt eine Tabelle

Syntax:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Beispiel:

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

DDL Befehl: Create II

Datentype für CREATE TABLE :

CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data

DDL Befehl: Alter

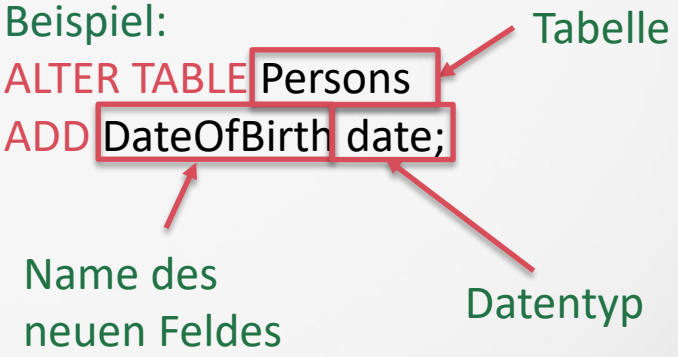
Funktion:
Ändert die Tabellenstruktur: Fügt zu oder löscht Spalten oder ändert diese

Syntax:
ALTER TABLE *table_name*
DROP COLUMN *column_name*;

Syntax:
ALTER TABLE *table_name*
ADD *column_name datatype*;

Syntax:
ALTER TABLE *table_name*
MODIFY *column_name datatype*;

Beispiel:
ALTER TABLE Persons
ADD DateOfBirth date;



DDL Befehl: DROP

Funktion:

Löscht die Tabelle

Syntax:

```
DROP TABLE table_name;
```

Beispiel:

```
DROP TABLE Shippers;
```

DML Befehl: INSERT INTO

Funktion:

Fügt Datensätze in die Tabelle ein

Syntax:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Syntax:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Diese Syntax kann verwendet werden, wenn sie alle Spalten mit Werten auffüllen

DML Befehl: INSERT INTO II

Beispiel 1:

```
INSERT INTO Personal (Vorname, Nachname, Position)
VALUES ('Harry', 'Washington', 'Trainee');
```

Beispiel 2:

```
INSERT INTO Personal
VALUES ('Harry', 'Washington', 'Trainee');
```

Beispiel 3:

```
INSERT INTO Kunden SELECT * FROM NeuKunden;
```

Fügt alle Datensätze von der Tabelle NeuKunden zu Kunden hinzu

DML Befehl: UPDATE

Funktion:
Ändert Werte in Feldern aufgrund Kriterien

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Beispiel:

```
UPDATE Bestellungen
SET Bestellmenge = Bestellmenge * 1.1, Fracht = Fracht * 1.03
WHERE Bestimmungsland = 'USA';
```

DML Befehl: UPDATE II

- ✓ UPDATE erzeugt keine Ergebnismenge.
- ✓ Die Operation kann nicht rückgängig gemacht werden.
- ✓ Welche Datensätze werden aktualisiert:
  - Zunächst Ergebnisse einer Auswahlabfrage(SELECT) untersuchen, die dieselben Auswahlkriterien verwendet
  - Aktualisierungsabfrage(UPDATE) ausführen.
- ✓ Erstellen Sie immer Sicherungskopien Ihrer Daten.
- ✓ Wenn Sie einen falschen Datensatz aktualisieren, können Sie diesen mit Hilfe der Sicherungskopien wiederherstellen

DML Befehl: DELETE

Funktion:

Löscht Datensätze aus einer Tabelle, die den Kriterien in dem WHERE Abschnitt entsprechen

Syntax:

```
DELETE FROM table_name
WHERE condition;
```

Beispiel:

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

DML Befehl: DELETE II

- ✓ Es werden nur die Daten gelöscht, die Tabellenstruktur(Feldattribute und Indizes) bleiben erhalten!
- ✓ Bei 1:n-Beziehung: Operationen mit Löschweitergabe in MS-Access löschen zusätzlich die Datensätze auf der n-Seite einer Beziehung, die mit dem Datensatz auf der 1-Seite, der durch die Abfrage gelöscht wird, in Beziehung stehen.