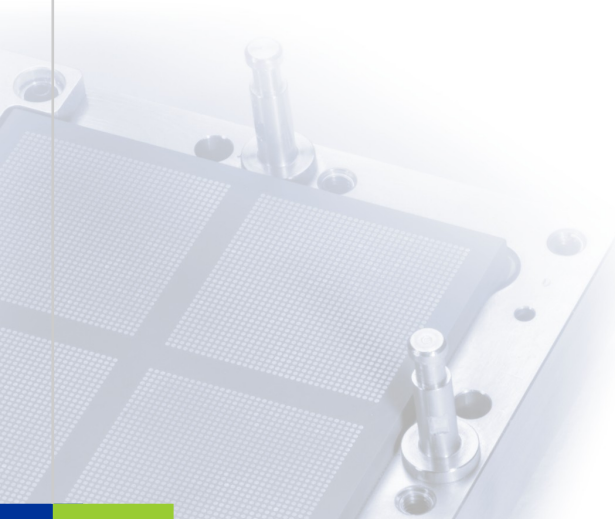


SQL (Structured Query Language)

Einführung



- RRZN-Handbuch "SQL, Grundlagen und Datenbank-Design"
- Alan Beaulieu: Einführung in SQL;
O'Reilly-Verlag; ISBN-Nr. 3897214431

- Strukturierte Abfragesprache für relationale Datenbanken.
- Datendefinition, -manipulation, -abfrage.
- ... definiert Kriterien, um nach Daten zu suchen.
- ... kann eine Menge von Datensätze automatisch aktualisieren oder löschen.
- ... besteht aus sehr wenigen Kommandos, sehr vielen Schlüsselwörter und einfachen Funktionen.
- In SQL sind keine Schleifen, bedingte Anweisungen oder die Nutzung von Variablen implementiert.

- DDL (Data Definition Language)
 - Erstellen von Datenbanken, Tabellen und Indizes.
 - Wird in Access nur im Zusammenhang mit VBA benutzt.
- DQL (Data Query Language)
 - Abfragen von Daten.
- DML (Data Manipulation Language)
 - Anlegen, ändern und löschen von Datensätzen.
- DCL (Data Controlling Language)
 - Anlegen von Benutzern und Vergabe von Zugriffsrechten.
 - Wird in Access nicht genutzt.

- .. können in einer Zeile oder in mehreren Zeilen eingegeben werden.
- ... werden mit einem Semikolon abgeschlossen.
- ... werden immer von links nach rechts gelesen.
- ... bestehen aus Befehlen und Ausdrücken.
- SQL ignoriert Groß- und Kleinschreibung bei Schlüsselwörtern, Tabellennamen und Spaltennamen.
- Schlüsselwörter werden in Großbuchstaben dargestellt.
- Das Ergebnis wird in einer temporären Tabelle dargestellt.

- Falls SQL in Abfragen oder Formularen / Steuerelementen genutzt wird, werden Zeichenketten durch Anführungsstriche begrenzt.
- Falls SQL in einer VBA-Anweisung genutzt wird, werden die Zeichenketten durch Apostroph begrenzt.
- Leerzeichen oder Zeichen, die in SQL vordefiniert sind, müssen in eckige Klammern eingeschlossen werden.
- Ein Datum wird durch Hash (#) begrenzt.

tabelle.feld
tabMitarbeiter.nachname

- Dem Tabellennamen folgt der Feldname. Der Feldname symbolisiert ein Feld in der angegebenen Tabelle.
- Der Tabellennamen wird durch ein Punkt von dem Feld getrennt.

- Die Abfrage ist in der Entwurfsansicht geöffnet. Das SQL-Eingabefenster kann über
 - ,, das Menü Ansicht - SQL-Ansicht,
 - ... das Symbol Ansicht und den Menüeintrag SQL-Ansicht
 - ... das Kontextmenü des Abfragefenster SQL-Ansicht geöffnet werden.
- In der SQL-Ansicht
 - ... wird eine vorhanden Abfrage in SQL dargestellt.
 - ... kann eine neue Abfrage erstellt werden.

- Datenherkunft (RecordSource) in Formularen und Berichten festlegen.
- Datensatzherkunft (RowSource) in Listenfeldern, Kombinationsfeldern und Nachschlagefeldern.

Argument	Beschreibung
SELECT	Leitet eine SQL-Anweisung ein.
Prädikat	Die Anzahl der Datensätze einschränken.
Tabelle.Feld	Der Name der Tabelle aus der Datensätze ausgewählt werden sollen. Feld beschreibt die Spalte innerhalb dieser Tabelle. Tabelle und Feld werden durch ein Punkt getrennt.
FROM	Aus welcher Tabelle kommen die Daten?
Tabelle	Name der Tabelle mit den Daten, die aufgerufen werden.
WHERE	Welche Bedingung muss ein Datensatz erfüllen?
GROUP BY	Datensätze werden zu Gruppen zusammengefasst.
HAVING	Welche Bedingung muss die Gruppe aus Datensätze erfüllen?
ORDER BY	Das Ergebnis wird auf- oder absteigend sortiert.

```
SELECT alle Felder FROM Tabelle;  
SELECT * FROM land;
```

- **SELECT** leitet die Anweisung an.
- Das Sternchen symbolisiert alle Felder der Tabelle
- **FROM** gibt Auskunft darüber, woher die Daten kommen.

```
SELECT Feldname, Feldname FROM Tabelle;  
SELECT [land].[IDland], [land].[land] FROM land;
```

- Hier werden bestimmte Felder angezeigt. Die Elemente in der Auflistung werden durch Kommata getrennt.
- Die angegebenen Felder sind alle in der Tabelle *land* enthalten.
- Um die Feld- und Tabellennamen werden mit Hilfe von eckigen Klammern zusammengefasst, wenn der Name Leerzeichen oder Sonderzeichen enthält.

```
SELECT land.IDland, land.land FROM land  
ORDER BY land.land;
```

- ORDER BY wählt ein Feld aus, nach dem die Daten sortiert werden.
- Wenn keine Angaben gemacht werden, werden die Daten aufsteigend von A bis Z sortiert.
- ORDER BY land.land DESC sortiert die Daten absteigend von Z bis A.

```
SELECT feld, feld FROM tabelle  
WHERE Bedingung;
```

```
SELECT land.IDland, land.land FROM land  
WHERE (land.land = "Irland");
```

- Es werden alle Datensätze angezeigt, die der Bedingung entsprechen.
- Die Bedingung oder das Kriterium für die Auswahl beginnt mit WHERE.
- Eine Bedingung kann sich folgendermaßen aufbauen:
 - feld Vergleichsoperator Konstante
 - feld Vergleichsoperator Steuerelement-Inhalt
- Mehrere Bedingungen können verknüpft werden.

- ... sind Ausdrücke, die einen boolschen Wert zurückliefern.
- ... vergleichen mit Hilfe von bestimmten Operatoren zwei Werte.
- ... sind zum Beispiel:
 - Wenn die Bestellmenge eine gewisse Höchstmenge überschreitet...
 - Wenn der Kontostand dem Dispo entspricht...
 - Wenn die Strecke A doppelt so lang ist wie Strecke B...
 - Wenn die Warenmenge eine Mindestmenge unterschreitet...

Operator	Rechenart	Beispiel
=	ist gleich	$(7 = 3) \Rightarrow \text{False}$
<	ist kleiner als	$(7 < 3) \Rightarrow \text{False}$
<=	ist kleiner gleich als	$(7 <= 3) \Rightarrow \text{False}$
>	ist größer als	$(7 > 3) \Rightarrow \text{True}$
>=	ist größer gleich als	$(7 >= 3) \Rightarrow \text{True}$
<>	ist ungleich	$(7 <> 3) \Rightarrow \text{True}$

```
sqlAnweisung = "SELECT artikelName, artikelPreis FROM tblArtikel"  
sqlAnweisung = sqlAnweisung & "WHERE tblArtikel.artikelPreis > 10"
```

```
SELECT land.IDland, land.land FROM land  
WHERE (land.land = "Irland");
```

```
sqlAnweisung = "SELECT artikelName, artikelmenge FROM tblBestellung"  
sqlAnweisung = sqlAnweisung & "WHERE tblArtikel.menge <> 1"
```

- ... oder relationale Operatoren.
- ... verknüpfen zwei oder mehr Bedingungen miteinander.
- Folgende Möglichkeiten sind vorhanden:
 - AND (Und, Konjunktion) ist nur wahr, wenn alle Bedingungen wahr sind.
 - OR (Oder, Disjunktion) ist wahr, sobald eine der Bedingungen wahr ist.
 - NOT (Negation) invertiert den booleschen Wert der Bedingung.

Bedingung				
a	b	Not a	a AND b	a OR b
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

```
sqlAnweisung = "SELECT artikelName, artikelPreis FROM tblArtikel"  
sqlAnweisung = sqlAnweisung & "WHERE"  
sqlAnweisung = sqlAnweisung & "(tblArtikel.artikelPreis > 10) AND"  
sqlAnweisung = sqlAnweisung & "(tblArtikel.artikelPreis < 100)"
```

```
SELECT land.IDland, land.land FROM land  
WHERE (land.land = "Irland") OR (land.land = "Finnland");
```

```
sqlAnweisung = "SELECT artikelName, artikelPreis FROM tblArtikel"  
sqlAnweisung = sqlAnweisung & "WHERE"  
sqlAnweisung = sqlAnweisung & "tblArtikel.artikelPreis BETWEEN 10 AND 100"
```

- BETWEEN Untergrenze AND Obergrenze
- Die Werte liegen in einem bestimmten Zahlenraum.
- Die Werte können zwischen zwei Werten liegen.

```
SELECT land.IDland, land.land FROM land  
WHERE (land.land) IN ("Finnland", "Irland", "Dänemark");
```

- IN folgt eine Liste, die durch Klammern begrenzt ist.
- Die Elemente der Liste werden durch Kommata getrennt.

- Mit Hilfe von LIKE statt dem Gleichheitszeichen können Strings miteinander verglichen werden.
- Die Strings werden auf Gleichheit geprüft.
- Im Suchmuster können folgende Platzhalter an beliebiger Position genutzt werden:
 - Sternchen, um kein, ein oder mehrere Zeichen zu ersetzen.
 - Fragezeichen, um ein Zeichen zu ersetzen.

Dim sqlAnweisung As String

```
sqlAnweisung = "SELECT * FROM tblMitarbeiter"
```

```
sqlAnweisung = sqlAnweisung & "WHERE tblMitarbeiter.Nachname LIKE 'Meier'"
```

```
Set rs = db.OpenRecordset(sqlAnweisung)
```

Dim sqlAnweisung As String

```
sqlAnweisung = "SELECT * FROM tblMitarbeiter"
```

```
sqlAnweisung = sqlAnweisung & "WHERE tblMitarbeiter.Nachname LIKE 'M*er'"
```

```
Set rs = db.OpenRecordset(sqlAnweisung)
```

Dim sqlAnweisung As String

```
sqlAnweisung = "SELECT * FROM tblMitarbeiter"
```

```
sqlAnweisung = sqlAnweisung & "WHERE tblMitarbeiter.Nachname LIKE 'Me?er'"
```

```
Set rs = db.OpenRecordset(sqlAnweisung)
```

```
SELECT artikelName, artikelPreis, verpackung  
FROM tblLieferung  
GROUP BY artikelName, artikelPreis, verpackung;
```

- GROUP BY folgen die Felder, nach dem die Datensätze gruppiert werden sollen.
- Alle Datensätze, die den gleichen Feldinhalt besitzen, werden zusammengefasst.

Milchprodukte	Yoghurt	1,99 €
Käse	Tilsiter	0,99 €
Milchprodukte	Quark	0,69 €
Obst	Äpfel	1,99 €
Käse	Butterkäse	1,02 €
Obst	Ananas	2,02 €
Käse	Butterkäse	1,12 €
Obst	Äpfel	2,20 €
Milchprodukte	Yoghurt	0,49 €

Milchprodukte	Yoghurt	1,99 €
		0,49 €
	Quark	0,69 €
	Äpfel	1,99 €
Obst	Äpfel	2,20 €
		Ananas
Käse	Butterkäse	1,02 €
		1,12 €
	Tilsiter	0,499 €

```
SELECT artikelName, artikelPreis, verpackung  
FROM tblLieferung  
GROUP BY artikelName, artikelPreis, verpackung  
HAVING (artikelPreis > 10);
```

- HAVING leitet eine Bedingung für die Filterung von Datensätzen ein, nach denen gruppiert werden soll.

```
SELECT artikelName, artikelPreis, menge, (artikelPreis * menge)  
FROM tblLieferung;
```

- Mit Hilfe folgender mathematischer Operatoren können Werte in einer SQL-Anweisung berechnet werden:
 - Sternchen für die Multiplikation
 - Schrägstrich für die Division
 - Pluszeichen für die Addition
 - Minuszeichen für die Subtraktion
- Mit Hilfe des kaufmännischen Unds (&) können Text verknüpft werden.

- In SQL können viele vordefinierte Funktionen für die mathematische Berechnung genutzt werden.
- Die Funktionen werden mit Hilfe ihrer englischen Bezeichnung aufgerufen.
- Sämtliche im Abfrageentwurf oder VBA enthalten Funktionen können genutzt werden.

```
SELECT preis, tage, buchung,  
        DateDiff('d', tabBuchung.buchungsDate, tabBuchung.bezahltAm)  
FROM tabBuchung  
WHERE (DateDiff('d', tabBuchung.buchungsDate, tabBuchung.bezahltAm) > 10);
```

```
SELECT pidArtikel, artikelName, artikelBeschreibung  
FROM tabArtikel  
WHERE ((tabArtikel.artikelBeschreibung Is Null);
```

```
SELECT pidArtikel, artikelName, artikelBeschreibung  
FROM tabArtikel  
WHERE ((tabArtikel.artikelBeschreibung Is Null);
```

Funktion	Bedeutung
Avg([Spalte])	Mittelwert aller Spaltenwerte verschieden von <i>NULL</i>
Count([Spalte])	Anzahl der Spaltenwerte verschieden von <i>NULL</i>
Count(*)	Anzahl der Spaltenwerte inklusive der Nullwerte
Min([Spalte])	Kleinsten Spaltenwert verschieden von <i>NULL</i>
Max([Spalte])	Maximaler Spaltenwert verschieden von <i>NULL</i>

Funktion	Bedeutung
First([Spalte])	Spaltenwert der ersten Zeile des Ergebnisses, kann <i>Null</i> sein.
Last([Spalte])	Spaltenwert der letzten Zeile des Ergebnisses.
StDev([Spalte])	Standardabweichungen einer Stichprobe de Spaltenwerte
StDevP([Spalte])	Standardabweichung der Grundgesamtheit der Spaltenwerte
Var([Spalte])	Varianz der Stichprobe der Spaltenwerte
VarP([Spalte])	Varianz der Grundgesamtheit der Spaltenwerte

```
SELECT anzahlTage, preisProNacht,  
       Avg(anzahlTage * preisProNacht) AS Endpreis  
FROM tblBuchung  
GROUP BY anzahlTage, preisProNacht;
```

- In diesem Beispiel wird der Mittelwert aller gebuchten Reisen ausgerechnet.
- Der Mittelwert wird in das Feld *Endpreis* gespeichert.
- In einer SQL-Anweisung
 - ... wird auf ein Feld eine Aggregatfunktion angewandt. Andere Felder sind nicht vorhanden. Eine Gruppierung ist nicht nötig.
 - ... , die aus mehreren Felder besteht, wird auf ein Feld eine Aggregatfunktion angewandt. Die Felder müssen gruppiert werden.

```
SELECT DISTINCT artikelName, artikelPreis, menge, (artikelPreis * menge)  
FROM tblLieferung;
```

- Es werden Datensätze mit gleichen Inhalt nicht angezeigt.
- Jeder Datensatz kommt exakt einmal vor.

```
SELECT TOP 2 artikelName, artikelPreis, menge, (artikelPreis * menge)  
FROM tblLieferung  
ORDER BY (artikelPreis * menge);
```

- Diese Datensätze werden nach dem Endpreis sortiert.
- Mit Hilfe von TOP 2 werden die zwei niedrigsten Lieferpreise angezeigt werden.
- Wenn mit Hilfe von ORDER BY (artikelPreis * menge) DESC sortiert wird, können die Lieferungen mit den höchsten Endpreisen gefiltert werden.

Europa	Frankreich
Europa	Spanien
Asien	China
Asien	Indien
	Antarktis
Afrika	Südafrika
	Arktis

Europa
Asien
Amerika
Australien
Afrika

*Inner
Join*

Europa	Frankreich
Europa	Spanien
Asien	China
Asien	Indien
Afrika	Südafrika

```
SELECT tabLand.land, tabKontinent..kontinent  
FROM tabKontinent  
INNER JOIN tabLand ON tabKontinent.idKontinent = tabLand.Kontinent;
```

- Die Tabellen *tabKontinent* wird mit der Tabelle *tabLand* verknüpft.
- Die Bedingung für die Verknüpfung wird durch das Schlüsselwort ON festgelegt. Es werden Felder angegeben, die übereinstimmende Werte enthalten können.
- Es wird eine 1 : n – Beziehung nachgebildet.
- Es werden nur die Datensätze angezeigt, deren verknüpfte Werte in beiden Tabellen vorkommen.
- Diese Art wird als Exklusionsverknüpfung bezeichnet.
- Das Ergebnis ist ein Dynaset. Das Ergebnis dieser SQL-Anweisung kann bearbeitet werden.

Europa	Frankreich
Europa	Spanien
Asien	China
Asien	Indien
	Antarktis
Afrika	Südafrika
	Arktis

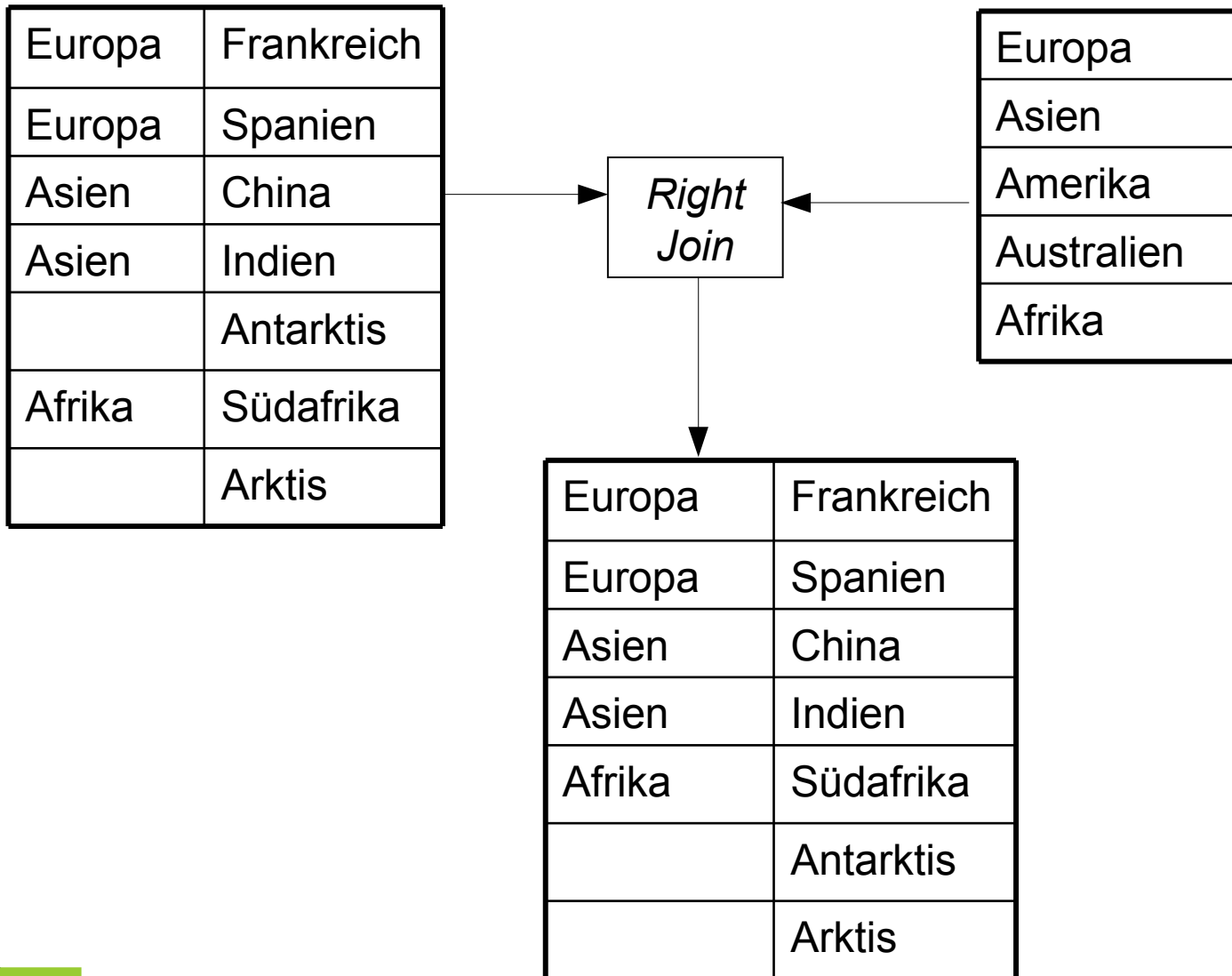
*Left
Join*

Europa
Asien
Amerika
Australien
Afrika

Europa	Frankreich
Europa	Spanien
Asien	China
Asien	Indien
Afrika	Südafrika
Amerika	
Australien	

```
SELECT tabLand.land, tabKontinent..kontinent  
FROM tabKontinent  
LEFT JOIN tabLand ON tabKontinent.idKontinent = tabLand.Kontinent;
```

- Die Tabellen *tabKontinent* wird mit der Tabelle *tabLand* verknüpft.
- Die Bedingung für die Verknüpfung wird durch das Schlüsselwort ON festgelegt. Es werden Felder angegeben, die übereinstimmende Werte enthalten können.
- Es werden alle Datensätze aus der linken Tabelle angezeigt. Es werden nur die Datensätze aus der rechten Tabelle angezeigt, die eine Beziehung zu der linken Tabelle besitzen.



```
SELECT tabLand.land, tabKontinent..kontinent  
FROM tabKontinent  
RIGHT JOIN tabLand ON tabKontinent.idKontinent = tabLand.Kontinent;
```

- Die Tabellen *tabKontinent* wird mit der Tabelle *tabLand* verknüpft.
- Die Bedingung für die Verknüpfung wird durch das Schlüsselwort ON festgelegt. Es werden Felder angegeben, die übereinstimmende Werte enthalten können.
- Es werden alle Datensätze aus der rechten Tabelle angezeigt. Es werden nur die Datensätze aus der linkenTabelle angezeigt, die eine Beziehung zu der linken Tabelle besitzen.