SQL
Informationen anzeigen und filtern

Alle Kunden, die im Land x wohnen

Alle Umsätze größer 10.000

Welche Bestellung wurden zwischen dem „01.03.2006“ und dem 31.03.2006“ aufgegeben?

Auswahlabfragen

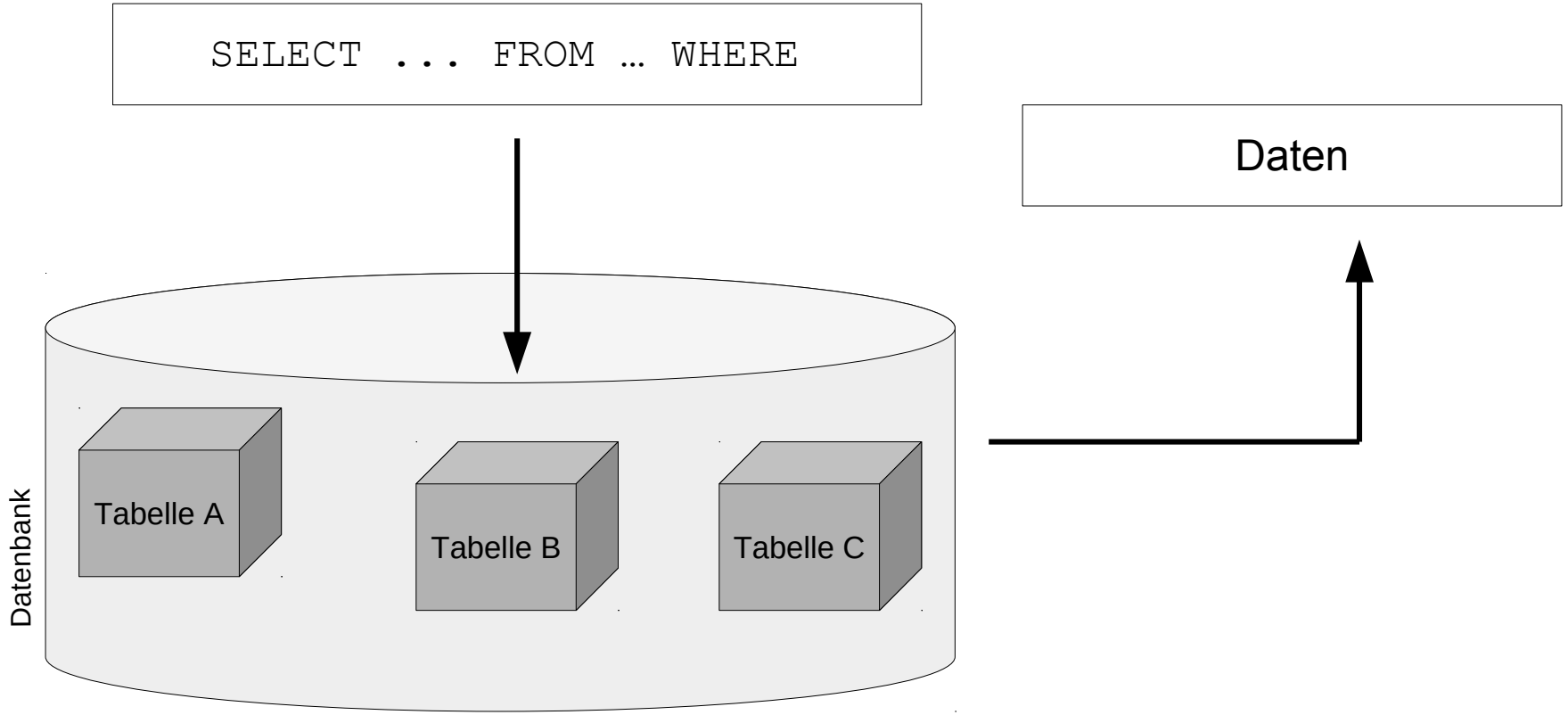
- Beginn mit `SELECT`.
- Anzeige von allen oder ausgewählten Datenfeldern.
- Filterung der Daten mit Hilfe von Bedingungen.
- Aufsteigende oder absteigende Sortierung von Datenfeldern.

Beispiele

```
-- Kunden aus Kanada.
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country Like 'Canada')
ORDER BY LastName, FirstName;
```

```
-- Tracks länger als 300000.
SELECT tracks.Name AS Trackname, Milliseconds
FROM tracks
WHERE (Milliseconds > 300000)
Order BY Milliseconds DESC;
```

Arbeitsweise



Ergebnis einer Auswahlabfrage

- Speicherung in einer temporären Ergebnistabelle.
- Die Ergebnistabelle basiert auf die, in der SQL-Anweisung angegebenen Datenfeldern aus einer definierten Datenquelle.
- Das Ergebnis ist von den, in der Datenquelle momentan gespeicherten Informationen abhängig.
- Die Sicht auf die Datensätze in der Ergebnistabelle kann mit Hilfe eines Filters eingeschränkt werden. Es werden nur die Datensätze angezeigt, die der Filter durchlässt. Falls der Filter keine Datensätze durchlässt, ist die Ergebnistabelle leer.

Datentypen in den Datenfeldern

- Alphanumerische Daten. Alle Zeichen eines Unicode-Zeichensatzes. Ziffern, Buchstaben, Satzzeichen etc.
- Numerische Daten. Ziffern aus denen neue Werte berechnet werden können.
- Datums- und Zeitwerte werden als alphanumerische Daten in SQLite dargestellt. In anderen Datenbanksystemen gibt es für diese Art von Daten eigene Datentypen.
- Binäre Daten, die von SQL nicht interpretiert werden können. Zum Beispiel Grafiken haben eine Struktur, die von SQL nicht interpretiert werden kann.

SQL-Befehle

- Beschreibung einer Aktivität. Zum Beispiel symbolisiert der Befehl SELECT die Tätigkeit „Wähle aus“.
- SQL-Befehle beginnen immer mit einem Buchstaben.
- Um die Lesbarkeit zu erhöhen, werden die Befehle häufig groß geschrieben.

Syntax

```
SELECT [Feld], [Feld]
FROM [Tabelle]
WHERE ([Bedingung])
ORDER BY [Feld] ASC|DESC, [Feld] ASC|DESC;
```

```
AUSWAHL VON [Datenfeld a], [Datenfeld b]
AUS DER [Tabelle / Datenquelle]
WO DIE ([Bedingung]) ERFÜLLT IST
SORTIERT NACH
[Feld a] aufsteigend|absteigend,
[Feld c] aufsteigend|absteigend;
```

Filterung von Daten

- In Abhängigkeit von bestimmten Kriterien werden die Daten gefiltert. Wenn das Auswahlkriterium zutrifft, passieren die Daten den Filter und werden angezeigt.
- In SQL-Anweisungen werden Bedingungen beschrieben, die der Wert in einem Datenfeld erfüllen muss. Andernfalls wird der dazugehörige Datensatz nicht in der Ergebnistabelle angezeigt.

Beispiele

- Alle Kunden aus Kanada.
- Alle Tracks, deren Spieldauer in Millisekunden größer als 300000 ist.
- Alle Tracks, die aus dem Genre „Easy Listing“, „Reggae“ und „Latin“ kommen.
- Alle Mitarbeiter, die nach dem Jahr 2015 angestellt wurden.
- Alle Mitarbeiter, die zwischen dem 1.1.2014 und dem 31.12.2014 eingestellt wurden.

Kriterien / Bedingungen

```
(Country Like 'Canada')
(Milliseconds > 300000)
(HireDate BETWEEN '2004-01-01' AND '2004-12-31')
```

- Ja / Nein-Fragen. Das Kriterium ist erfüllt (ja, wahr, true) oder nicht (nein, falsch, false).
- Wenn das Kriterium nicht zutrifft, ist die Ergebnistabelle der SQL-Anweisung leer.
- Filtermuster, die mit Hilfe von Operatoren und Operanden zusammengestellt werden.
- Bedingungen können zu einem komplexen Kriterium verknüpft werden.

Operatoren

```
(Country Like 'Canada')
(Milliseconds > 300000)
(HireDate BETWEEN '2004-01-01' AND '2004-12-31')
(GenreID IN(7, 8, 12))
```

- Vergleichsoperatoren. Wie werden die Daten aus den Feldern mit dem Muster verglichen?
- Werte müssen innerhalb einer unteren und oberen Grenze liegen.
- Werte müssen Werte in einer Liste entsprechen.

Vergleichsoperatoren

ist ...	Operator		
gleich	=		
ungleich	<>		
	!=		
kleiner	<		
kleiner gleich	<=		
größer	>		
größer gleich	>=		

Operanden

```
(Country Like 'Canada')
(Milliseconds > 300000)
(HireDate BETWEEN '2004-01-01' AND '2004-12-31')
(GenreID IN(7, 8, 12))
```

- Feldnamen. Welche Daten sollen untersucht werden? Feldnamen werden links vom Operator genutzt.
- Statische Werte. Vergleichsmuster für die, in einem Feld gespeicherten Daten. Literale, die direkt in die Bedingung eingegeben werden. Statische Werte stehen immer rechts vom Operator.

… vom Datentyp

- Kategorisierung von Daten in Abhängigkeit des Typs.
- Wie können die Daten verarbeitet werden?
- Wie viel Platz wird für die Ablage der Information benötigt? Die Angabe erfolgt in Bytes.

Mögliche Datentypen

- In Abhängigkeit des Datenbanksystems sind die verschiedenen Datentypen implementiert.
- SQLite hat zum Beispiel nur Storage Classes, die wie Datentypen behandelt werden.

Verarbeitung von Daten

- Mathematische Rechnungen mit Hilfe von Gleitkomma- und Ganzzahlen.
- Einige Datenbanksysteme bieten Datums- und Zeitberechnungen an.
- Speicherung von Daten ohne weitere Berechnung. Die Daten werden binär oder als Text gespeichert. Postleitzahlen bestehen zum Beispiel aus Ganzzahlen, die aber für keine weitere Berechnungen benötigt werden. Postleitzahlen werde als Text gespeichert.

Storage Classes in SQLite

- SQLite hat keine Datentypen, sondern nur Storage Classes.
- Die Klassen beschreiben einen Datentyp allgemein.
- Informationen im Web: https://www.sqlite.org/datatype3.html.

Möglichkeiten

- `INTEGER`. Ganzzahl.
- `REAL`. Zahlen in diesem Format nähren sich immer einen Wert von doppelter Genauigkeit an. Gleitkommazahlen belegen 8 Byte (IEEE-Format).
- `TEXT`. String. Alphanumerische und numerische Zeichen im Format UTF-8, UTF-16.
- `BLOB`. Binary Large Object. Binäre Daten wie zum Beispiel Grafiken etc.
- `NULL`. Das Datenfeld hat einen undefinierten Wert.

Ganzzahlen

- Ganzzahlen sind vorzeichenbehaftet.
- Positive Ganzzahlen als Literale: 3, +13456 und so weiter.
- Negative Ganzzahlen als Literale: -5, -3456 und so weiter.
- Ein Datenfeld mit der Angabe `integer` oder `int` speichert eine Ganzzahl in der Storage Class `INTEGER`.

Gleitkommazahl

- Zahlen mit Nachkommastellen. Als Dezimaltrennzeichen wird der Punkt genutzt.
- Zahlen, die sich einen bestimmten Wert nähren. Gleitkommazahlen sollten nicht in kaufmännischen Berechnungen genutzt werden.
- Zahlen wie zum Beispiel 3.5, 0.345667, 2.0e+24 und so weiter sind Gleitkommazahlen.

Datentyp „float“

- Datenfelder mit der Angabe `float(12)` speichern Gleitkommazahlen in der Storage Class `REAL`.
- Die Bezeichnung `float` hat eine Parameterliste. Die runden Klammern folgen direkt dem Namen.
- In der runden Klammern wird mit Hilfe einer Ganzzahl die Genauigkeit der Gleitkommazahl angegeben. In diesem Beispiel ist die Zahl auf 12 Stellen genau.

Dezimalzahlen

- Zahlen mit Nachkommastellen. Als Dezimaltrennzeichen wird der Punkt genutzt.
- Zahlen, die für die Angabe von Preisen in Datenbanken genutzt werden.
- Zahlen wie zum Beispiel 3.55, 0.99 und so weiter sind Dezimalzahlen.

Datentyp „numeric“

- Datenfelder mit der Angabe `numeric(10, 2)` speichern Dezimalzahlen in der Storage Class `REAL`.
- Die Bezeichnung `float` hat eine Parameterliste. Die runden Klammern folgen direkt dem Namen. Die Parameter in der Liste werden durch ein Komma getrennt.
- Der erste Parameter legt die Genauigkeit der Zahl fest.
- Der zweite Parameter definiert die Anzahl der Nachkommastellen.

Boolean

- Beantwortung von Ja- / Nein-Fragen.
- Interpretation in SQLite als Ganzzahl (`INTEGER`).
- Das Kriterium ist erfüllt: true, wahr, 1.
- Das Kriterium ist nicht erfüllt: false, falsch, 0.

Vergleich von Zahlen

```
(Milliseconds > 300000)
(Total = 5.94)
```

- Mit Hilfe von Vergleichsoperatoren können in Datenfeldern gespeicherte Informationen mit einer Zahl verglichen werden.
- Wenn der Vergleich zutrifft, wird der Datensatz angezeigt.
- Hinweis: Gleitkommazahlen sollten nicht auf Gleichheit geprüft werden.

Vergleich von Zahlenwerten

ist ...	Operator	Kriterium	Ergebnis
gleich	=	3 = 4	Falsch
ungleich	<>	3 <> 4	Wahr
	!=	3 != 4	Wahr
kleiner	<	3 < 4	Wahr
kleiner gleich	<=	3 <= 4	Wahr
größer	>	3 > 4	Falsch
größer gleich	>=	3 >= 4	Falsch

Text

- Alphanumerische und numerische Zeichen.
- Zeichenketten. String.
- Als Text wird zum Beispiel die Länderbezeichnung „Canada“, die Postleitzahl „30159“ oder die Verpackungsangabe „24 - 12 oz bottles“ gespeichert.
- In Filterkriterien wird Text am Anfang und Ende immer durch ein Apostroph begrenzt. Zum Beispiel: 'Canada', '30159'.

Datentypen

- Die Angabe `nvarchar(160)` speichert in diesem Beispiel maximal 160 Unicode-Zeichen.
- Die Angabe `varchar(15)` speichert in diesem Beispiel maximal 15 Zeichen. Der Typ varchar nutzt immer eine 8 Bit Zeichenkodierung wie zum Beispiel die ASCII-Zeichenkodierung.
- Die Angabe `text` stellt einen Container bereit, um beliebigen langen Text zu speichern.

Hinweise

- Zeichen werden in Abhängigkeit ihrer Zeichenkodierung verglichen.
- SQLite kann nur ASCII-Zeichen unabhängig von der Groß- und Kleinschreibung miteinander vergleichen.
- Unicode-Zeichen werden in Abhängigkeit der Groß- und Kleinschreibung verglichen.

Vergleich von Text

ist ...	Operator	Kriterium	Ergebnis
gleich	=	'Calgary' = 'Edmonton'	Falsch
ungleich	<>	'Calgary' <> 'Edmonton'	Wahr
	!=	'Calgary' != 'Edmonton'	Wahr
kleiner	<	'Calgary' < 'Edmonton'	Wahr
kleiner gleich	<=	'Calgary' <= 'Edmonton'	Wahr
größer	>	'Calgary' > 'Edmonton'	Falsch
größer gleich	>=	'Calgary' >= 'Edmonton'	Falsch

Operator „ist gleich“

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country = 'Canada')
ORDER BY LastName, FirstName;
```

- Bei einem Vergleich mit Hilfe des Gleichheitszeichens muss der zu vergleichende Text und der Vergleichstext Zeichen für Zeichen übereinstimmen.
- Es werden nur Daten angezeigt, die dem angegebenen Textmuster entsprechen.

Operator „LIKE“

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country LIKE 'Canada')
ORDER BY LastName, FirstName;
```

- Der Operator `LIKE` entspricht dem Gleichheitszeichen.
- Bei einem Vergleich mit dem Operator `LIKE` können einzelne Zeichen im Vergleichstext durch Wildcards ersetzt werden.

Wildcards

- Platzhalter für ein oder mehrere Zeichen.
- Welche Wildcards genutzt werden können, ist abhängig vom gewählten Datenbanksystem.
- Wildcards können an jeder beliebigen Position in dem Vergleichsmuster vorkommen.
- Wildcards können beliebig miteinander kombiniert werden.

Wildcard %

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country LIKE 'C%')
ORDER BY LastName, FirstName;
```

- Das Prozentzeichen ist ein SQL-standardkonformer Platzhalter.
- Das Prozentzeichen steht für kein, ein oder mehrere Zeichen.

Wildcard _

```
SELECT LastName, FirstName, Company, Country, State
FROM customers
WHERE (State LIKE '__')
ORDER BY LastName, FirstName;
```

- Der Unterstrich ist ein SQL-standardkonformer Platzhalter.
- Der Unterstrich steht für ein beliebiges alphanumerisches oder numerisches Zeichen.

Operator „Glob“

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country GLOB 'Canada')
ORDER BY LastName, FirstName;
```

- Der Operator `GLOB` entspricht dem Gleichheitszeichen.
- Bei einem Vergleich mit dem Operator `GLOB` können einzelne Zeichen im Vergleichstext durch Wildcards ersetzt werden.
- Der Operator beachtet die Groß- und Kleinschreibung.
- Der Vergleich entspricht der Syntax des Betriebssystems UNIX.

Wildcard *

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country GLOB 'C*')
ORDER BY LastName, FirstName;
```

- Das Sternchen steht für kein, ein oder mehrere Zeichen.

Wildcard ?

```
SELECT LastName, FirstName, Company, Country, State
FROM customers
WHERE (State LIKE '??')
ORDER BY LastName, FirstName;
```

- Das Fragezeichen steht für ein beliebiges alphanumerisches oder numerisches Zeichen.

Wertebereich

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country GLOB '[A-D]*')
ORDER BY LastName, FirstName;
```

- In den eckigen Klammern wird ein Bereich von Werten definiert.
- In diesem Beispiel werden alle Länder angezeigt, die mit A, B, C oder D beginnen.

Wertebereich

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country GLOB '[A-D]*')
ORDER BY LastName, FirstName;
```

- In den eckigen Klammern wird ein Bereich von Werten definiert.
- Die Untergrenze wird von der Obergrenze durch den Bindestrich getrennt.
- In diesem Beispiel werden alle Länder angezeigt, die mit A, B, C oder D beginnen.

Werteliste

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country GLOB '[ADF]*')
ORDER BY Country, LastName, FirstName;
```

- In den eckigen Klammern wird eine Liste von Zeichen definiert.
- Die einzelnen Elemente in der Liste werden direkt hintereinander geschrieben
- In diesem Beispiel werden alle Länder angezeigt, die mit A, D oder F beginnen.

Negation eines Wertebereichs oder -liste

```
SELECT LastName, FirstName, Company, Country
FROM customers
WHERE (Country GLOB '[^ADF]*')
ORDER BY Country, LastName, FirstName;
```

- Das Zirkumflex ^ negiert die Werte in einer Liste oder Bereich.
- In diesem Beispiel werden alle Länder angezeigt, die nicht mit A, D oder F beginnen.

GLOB

```
select * from tab where col_nocase = 'One' collate binary;
```

- Wird die Groß- und Kleinschreibung bei einem Vergleich mit `LIKE` beachtet?
- Bei der Nutzung von ASCII-Zeichen ist der Wert standardmäßig `false`.

Pragma-Anweisungen

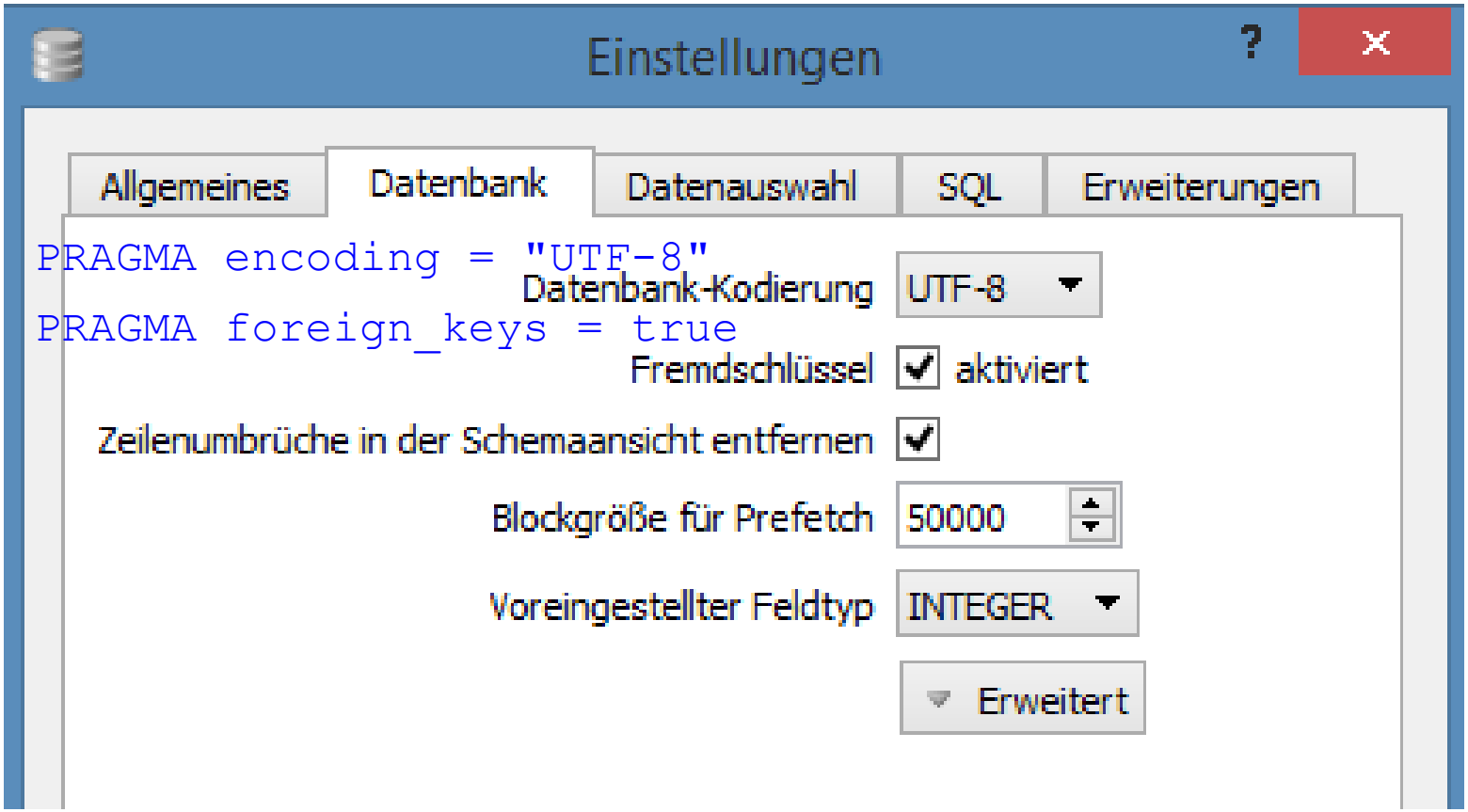
```
PRAGMA case_sensitive_like = false;
```

- Globale Einstellungen für das Datenbanksystem.
- Einstellung die, die genutzte Textkodierung oder den Vergleich von Texten betreffen.
- Einstellungsmöglichkeiten in SQLite: https://www.sqlite.org/pragma.html

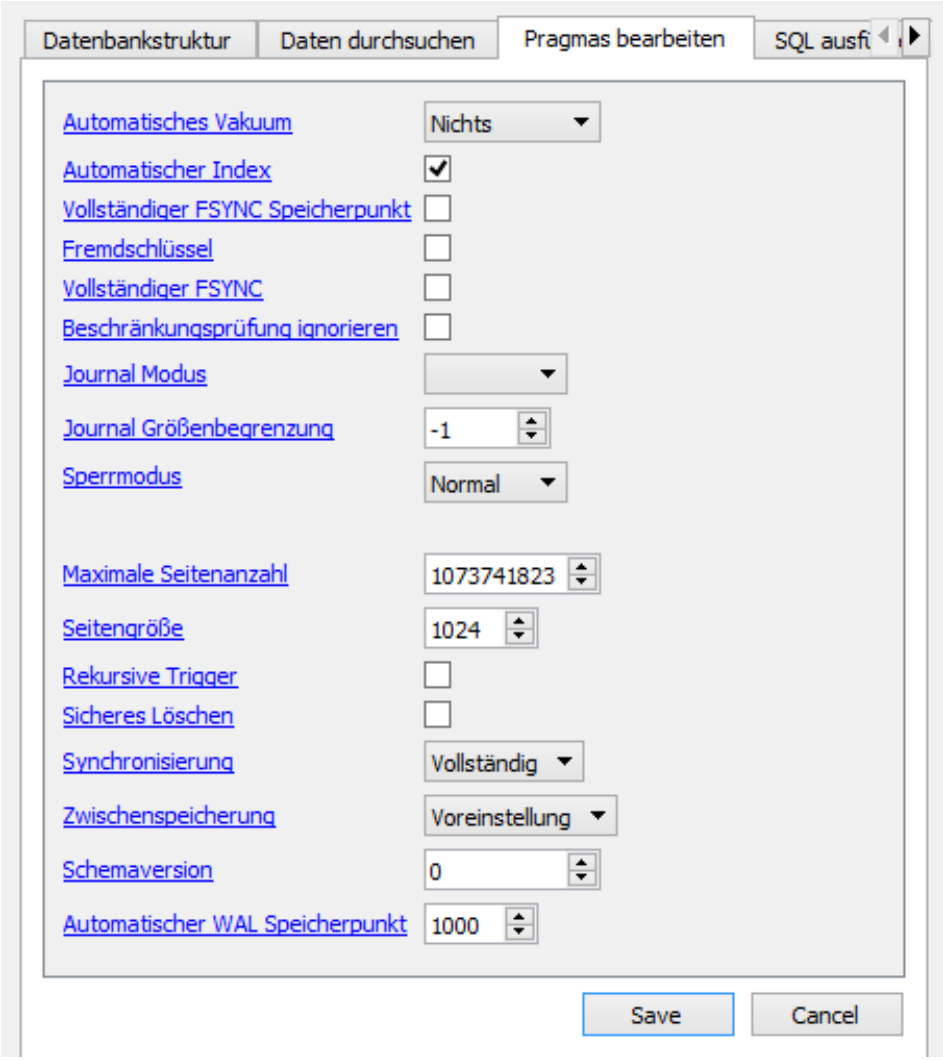
… im DB Browser

- Registerkarte *Pragmas bearbeiten*.
- *Ansicht – Einstellungen*. Registerkarte *Datenbank*.

Einstellungen für die Datenbank



Pragmas bearbeiten



Beachtung der Groß- und Kleinschreibung

```
PRAGMA case_sensitive_like = true;
```

- Beachtung der Groß- und Kleinschreibung in Bezug auf die Nutzung des Operators `LIKE`.
- Standardmäßig wird die Groß- und Kleinschreibung nicht beachtet.

Datums- und Zeitformate im SQL-Standard

- Datum (Date): '1968-01-09'. 'yyyy-mm-dd'
- Zeit (Time): '20:12:00 +01:00'. 'hh:mm:ss'. Die Stunden werden in einem 24-Stunden-Format angegeben. Eine Zeitverschiebung in Abhängigkeit der UTC (Universal Time Coordinate) kann angegeben werden. In diesem Beispiel wird die Sommerzeit berücksichtigt.
- Zeitstempel (Timestamp): '20:12:00.123 +01:00'. 'hh:mm:ss.dddd +|-hh:mm'. Die Bruchteile einer Sekunde wird mit Hilfe des Punktes an die Zeitangaben angehängt. Diese Angaben werden häufig für Log-Einträge genutzt.

Datums- und Zeitwerte in SQLite

- Eine Tabellenspalte mit dem Datentyp `datetime` oder `timestamp` wird in SQLite als ISO 8601 - String gespeichert.
- Der angegebene Datentyp wird in der Storage Class `TEXT` abgelegt.
- Datums- und Zeitangaben werden als Vergleichswerte in einer Bedingung in dem Typ „Text“ angegeben. Die Reihenfolge der Tages- und Monatsangaben ist abhängig von dem genutzten Format in dem zu filternden Datenfeld.

Beispiele für Operanden

- Die Angabe '1968-01-09 00:00:00' entspricht dem SQL-Standard. Das Datum wird in der Form yyyy-mm-dd angegeben.
- Die Angabe '8/14/1992 12:00:00 AM' entspricht nicht dem SQL-Standardformat. Das Datumsformat wird in der Form m/d/yyyy gespeichert. Bei einstelligen Monats- und Tagesangaben wird keine Null hinzugefügt. Zeitangaben werden im 12-Stunden-Format gespeichert.

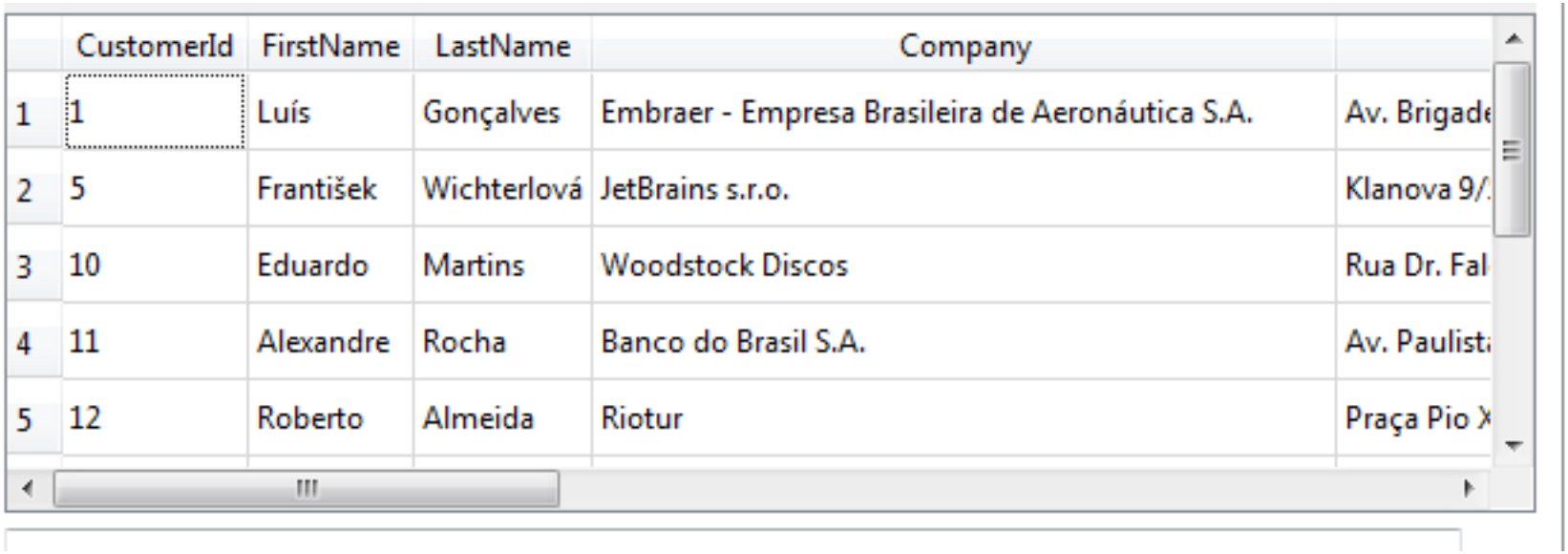
Operator „Like“

```
SELECT CustomerID, Total, InvoiceDate
FROM invoices
WHERE(InvoiceDate LIKE '2009-01-__ %');
```

- Mit Hilfe des Operators `LIKE` und deren Wildcards können auch Datums- und / oder Zeitangaben gefiltert werden.
- In diesem Beispiel werden alle Bestellung im Monat Januar 2009 angezeigt.

Definierter Wert in einem Datenfeld

```
SELECT * FROM customers
WHERE (Company IS NOT NULL);
```



	CustomerId	FirstName	LastName	Company	
1	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigade
2	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/
3	10	Eduardo	Martins	Woodstock Discos	Rua Dr. Fal
4	11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulist
5	12	Roberto	Almeida	Riotur	Praça Pio X

Erläuterung

- Das Datenfeld `Company` ist (`IS`) definiert (`NOT NULL`).
- Die Information ist in dem Datensatz vorhanden. Das Attribut `Company` ist für das zu beschreibende Objekt gesetzt.
- In der Ergebnistabelle werden in diesem Beispiel alle Kunden angezeigt, denen eine Firma zugeordnet ist.

Undefinierter Wert in einem Datenfeld

```
SELECT * FROM customers
WHERE (Company IS NULL);
```

CustomerId	FirstName	LastName	Company	Address	Ci
2	Leonie	Köhler	NULL	Theodor-Heuss-Straße 34	Stuttgar
3	François	Tremblay	NULL	1498 rue Bélanger	Montré
4	Bjørn	Hansen	NULL	Ullevålsveien 14	Oslo
6	Helena	Holý	NULL	Rilská 3174/6	Prague
7	Astrid	Gruber	NULL	Rotenturmstraße 4, 1010 Innere Stadt	Vienne

Erläuterung

- Das Datenfeld `Company` ist (`IS`) nicht definiert (`NULL`).
- Die Information liegt für diesen Datensatz nicht vor. Ob das Attribut `Company` für das beschriebene Objekt existiert, ist nicht bekannt.
- SQLite zeigt in der Ergebnistabelle `NULL` an. Der Nutzer hat nie Eingaben in dieses Feld gemacht.

Leerer Wert

```
SELECT * FROM customers
WHERE (State = '');
```

	CustomerId	FirstName	LastName	Company	Address	City	State	Country
1	11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo		Brazil

Erläuterung

- Das Datenfeld `State` ist leer (`''`).
- Die beiden direkt aufeinanderfolgenden Apostrophen kennzeichnen eine leere Zeichenfolge. 0 oder 0.0 kennzeichnen bei Zahlen ein leeres Feld.
- Die Information ist momentan nicht vorhanden. Es existiert kein Attributwert `State` zu dem beschriebenen Objekt.
- SQLite zeigt in der Ergebnistabelle eine leere Zelle an. Der Nutzer hat in das Feld einen Wert eingetragen, aber diesen gelöscht.

Nicht(Bedingungen)

```
SELECT * FROM customers
WHERE NOT(EMail LIKE '%@gmail.com');
```

- Negation der Bedingung.
- `NOT`([Bedingung] == true) = false.
- `NOT`([Bedingung] == false) = true

Bedingung oder Bedingungen

```
SELECT * FROM customers
WHERE (City LIKE 'São Paulo')
      OR (City LIKE 'Rio de Janeiro')
      OR (City LIKE 'Brasília');
```

- Eine der angegebenen Bedingungen muss wahr sein.
- Wenn die erste Bedingung wahr ist, werden alle anderen nicht mehr überprüft.

Bedingung und Bedingungen

```
SELECT Name, GenreID, Milliseconds
FROM tracks
WHERE ((GenreID = 1) AND (Milliseconds > 800000));
```

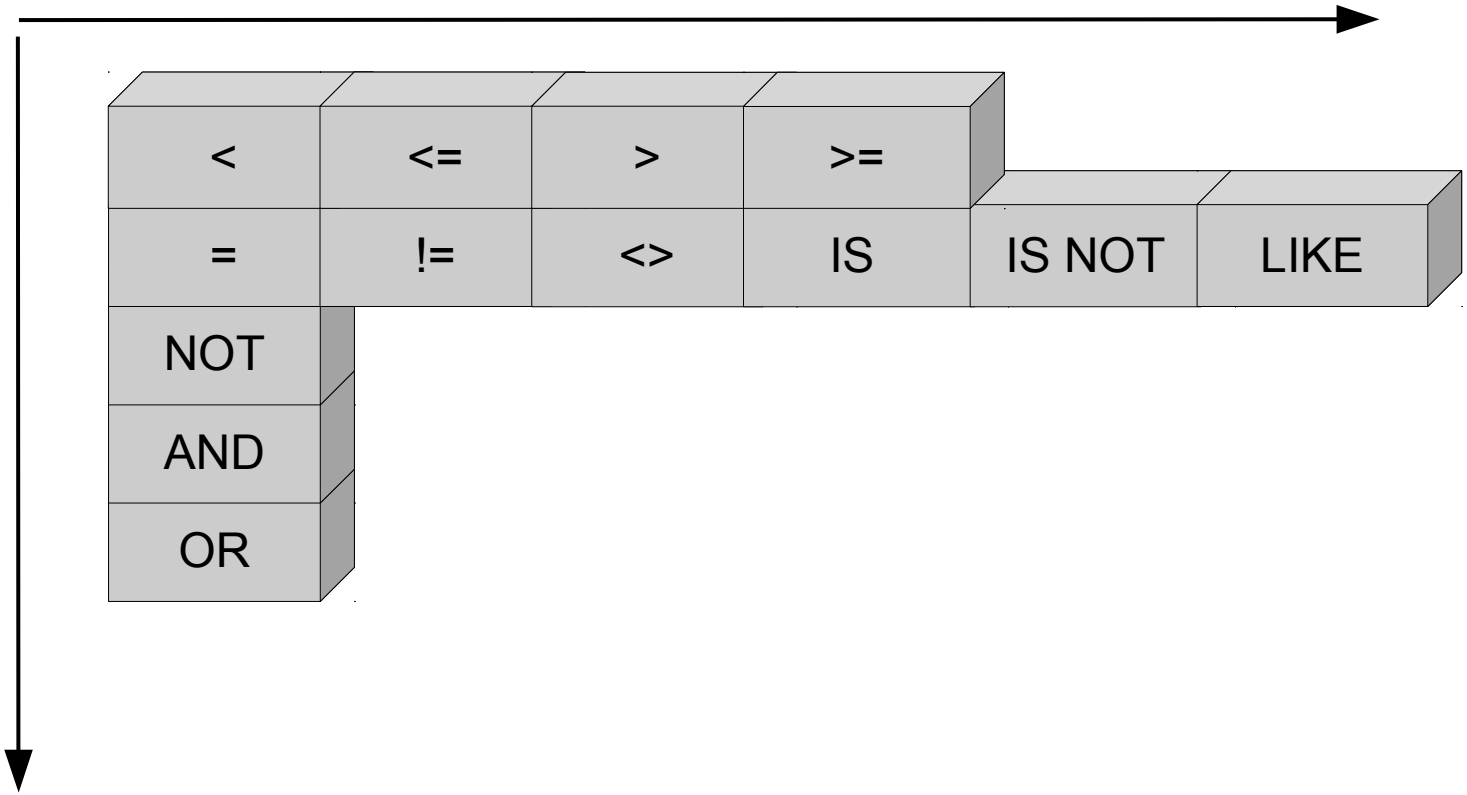
- Alle, die mit `AND` verknüpften Bedingungen müssen wahr sein.
- Wenn die erste Bedingung falsch ist, werden alle anderen nicht mehr überprüft.

Kombination von Verknüpfungen

```
SELECT Name, GenreID, Milliseconds
FROM tracks
WHERE (((GenreID = 3) OR (GenreID = 7))
      AND (Milliseconds > 400000));
```

- Die runden Klammern dienen nur der besseren Lesbarkeit.

Gewichtung der Operatoren



Liste von Werten

```
SELECT * FROM customers
WHERE (State IN ('CA', 'WA', 'NY'));

SELECT * FROM tracks
WHERE (genreID IN(2, 4, 6));

SELECT * FROM tracks
WHERE ((genreID = 2)
      OR (genreID = 4) OR (genreID = 6));
```

Erläuterung

- Der Inhalt des Datenfeldes muss einem Element in der Liste entsprechen.
- Mit Hilfe von `IN()` wird eine Liste von erlaubten Elementen definiert.
- Die Liste der erlaubten Elemente beginnt und endet mit den runden Klammern. Die Liste kann beliebig viele Elemente enthalten.
- Die Elemente in der Liste werden durch ein Komma getrennt.
- In SQLite können Zahlen oder Strings als Vergleichswerte genutzt werden.

Zwischen … und ...

```
SELECT * FROM tracks
WHERE (Bytes BETWEEN 6000000 AND 7000000);

SELECT * FROM tracks
WHERE ((Bytes >=  6000000) AND (Bytes <= 7000000));
```

- Das Datenfeld hat einen Wert zwischen (`BETWEEN`) der Unter- und (`AND`) der Obergrenze.
- Der Wert des Datenfeldes ist größer gleich der Untergrenze und kleiner gleich der angegebenen Obergrenze.

… für Datumsangaben

```
SELECT employees.*
FROM employees
WHERE (HireDate
      BETWEEN '2004-01-01' AND '2004-12-31');
```

- In SQLite werden Datumsangaben als String angegeben.