# S(tructured)Q(uery)L(anguage) 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

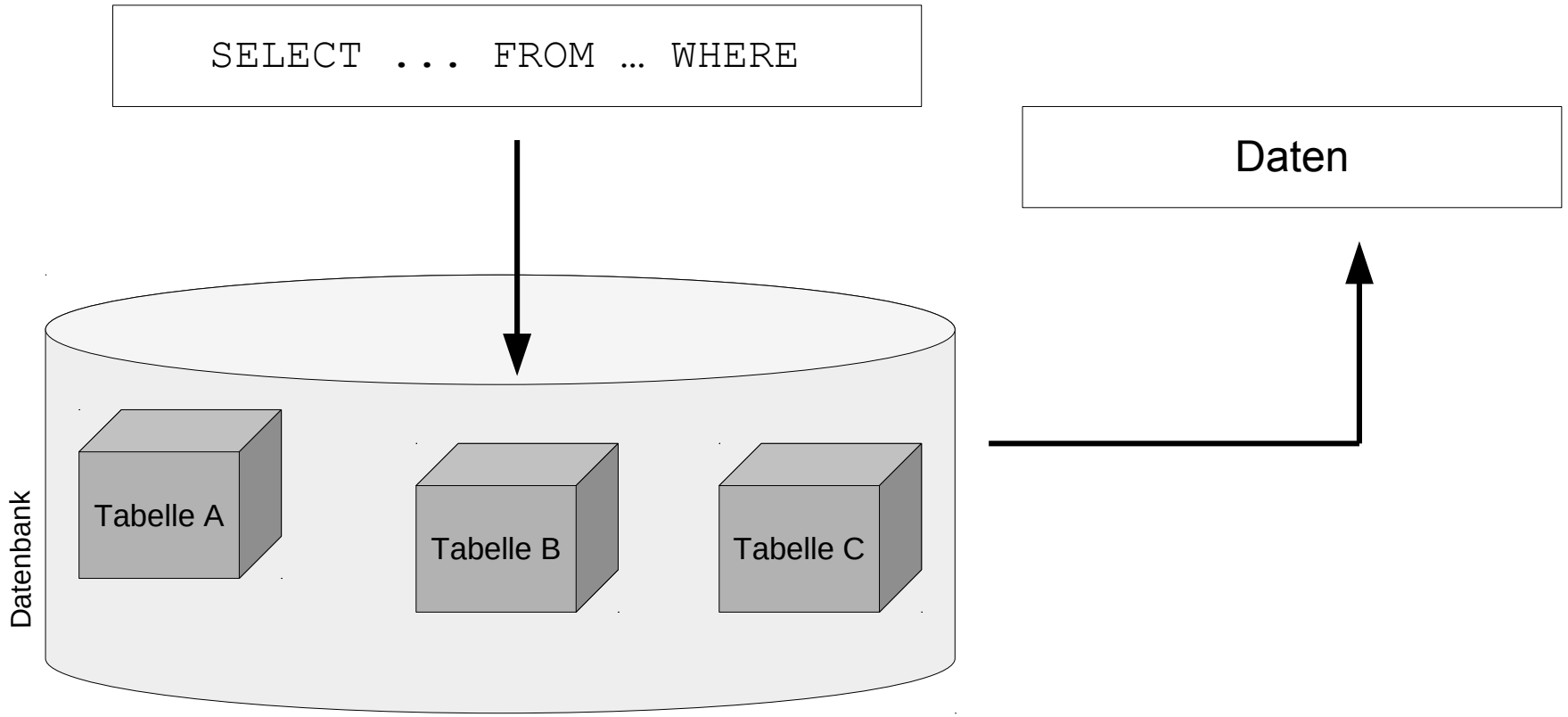
- Beginn mit `SELECT`.
- Anzeige von allen oder ausgewählten Datenfeldern.
- Filterung der Daten mit Hilfe von Bedingungen.
- Aufsteigende oder absteigende Sortierung von Datenfeldern.

# Beispiele

```
-- Kunden aus Kanada.
SELECT "LastName", "FirstName",
FROM "Customer"
WHERE ("Country" LIKE 'Canada')
ORDER BY "LastName", "FirstName";
```

```
-- Tracks länger als 300000.
SELECT "Track"."Name" AS Trackname, "Milliseconds"
FROM "Track"
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.

# Filterung von Daten

- Die Sicht auf die Datensätze in der Ergebnistabelle kann mit Hilfe eines Filters eingeschränkt werden.
- Filter werden in SQL-Anweisungen mit Hilfe von Bedingungen definiert, die ein Datensatz erfüllen muss.
- Falls keine Datensätze entsprechend des Filters vorhanden sind, ist die Ergebnistabelle der Abfrage leer.

# 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.

# Syntax einer Auswahlabfrage

```
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;
```

# Kriterien / Bedingungen

```
("Total" > 5.5)
("Milliseconds" > 300000)
("Bytes" BETWEEN 6000000 AND 7000000)
```

- Bedingungen bestehen aus Operatoren und Operanden.
- Vergleich von Werten und einem Datenfeld.
- Filter, die mit Hilfe von Operatoren und Operanden definiert werden.
- Bedingungen können zu einem komplexen Kriterium verknüpft werden.

# Operatoren

```
("Milliseconds" > 300000)
("Country" LIKE 'Canada')
("GenreID" IN(2, 4, 6))
("Bytes" BETWEEN 6000000 AND 7000000)
```

- Vergleichsoperatoren.
- Vergleich von Text mit einem Muster.
- Vergleich mit Werten in einer Liste.
- Vergleich in Abhängigkeit eines Wertebereichs.

# Operanden

```
("Milliseconds" > 300000)
("Country" LIKE 'Canada')
("GenreID" IN(2, 4, 6))
("Bytes" BETWEEN 6000000 AND 7000000)
```

- Feldnamen, deren Inhalt mit einem Wert verglichen werden sollen.
- Literale oder Textmuster mit den der Inhalt in einem Datenfeld verglichen werden soll.

# Vergleichsoperatoren

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

# Vergleich von Zahlen

```
("Milliseconds" > 300000)
("Total" > 5.5)
```

- Datenfelder, die eine Gleitkommazahl oder Ganzzahl speichern, können mit Hilfe von Vergleichsoperatoren gefiltert werden.
- Hinweis: Gleitkommazahlen nähren sich nur einem Wert an. Aus diesen Grund sollten diese Art von Zahlen nie 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 |

# Vergleich von Text

```
("City" <> 'Calgary')
("City" LIKE 'Calgary')
```

- Zeichenketten werden durch ein Apostroph am Anfang und Ende begrenzt.
- Text kann mit Hilfe von Vergleichsoperatoren oder Textmustern verglichen werden.

# 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 „Like“

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

- Der Operator `LIKE` entspricht dem Gleichheitszeichen. Das Datenfeld muss exakt dem angegebenen Textmuster entsprechen.
- Das Datenfeld wird mit einem Textmuster verglichen. Zeichen in dem Textmuster werden durch Wildcards ersetzt.

# Wildcards

```
("Country" LIKE 'C%')
("State" LIKE '__')
```

- Wildcards sind Platzhalter für eine bestimmte Anzahl von Zeichen in einem Text.
- Wildcards können an jeder beliebigen Position in einem Textmuster vorkommen.
- Wildcards können beliebig häufig in einem Textmuster vorkommen.

# 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.

# Maskierung von Wildcards

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

- Das Prozentzeichen und der Unterstrich werden in einem Textmuster als Wildcards interpretiert.
- In PostgreSQL können Wildcards mit dem Backslash maskiert werden. Mit dem oben angegebenen Beispiel würden Texte wie zum Beispiel HB_1 oder XC_A345 herausgefiltert.

# Definierter Wert in einem Datenfeld

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

- 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 "Customer"
WHERE ("Company" IS NULL);
```

- 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.
- In der Ergebnistabelle wird in dem Datenfeld kein Wert angezeigt. Das Feld ist leer.

# Leere Zeichenfolge

```
SELECT * FROM "Customer"
WHERE ("Company"  = '');
```

- Das Datenfeld `State` enthält eine leere Zeichenfolge (`''`).
- Das Datenfeld ist definiert, aber leer.
- In der Ergebnistabelle wird in dem Datenfeld kein Wert angezeigt. In der Anzeige kann das Feld nicht von `IS NOT NULL` nicht unterschieden werden.

# Nicht(Bedingungen)

```
SELECT * FROM "Customer"
WHERE NOT("Email" LIKE '%@gmail.com');
```

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

# Bedingung oder Bedingungen

```
SELECT *
FROM "Customer"
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

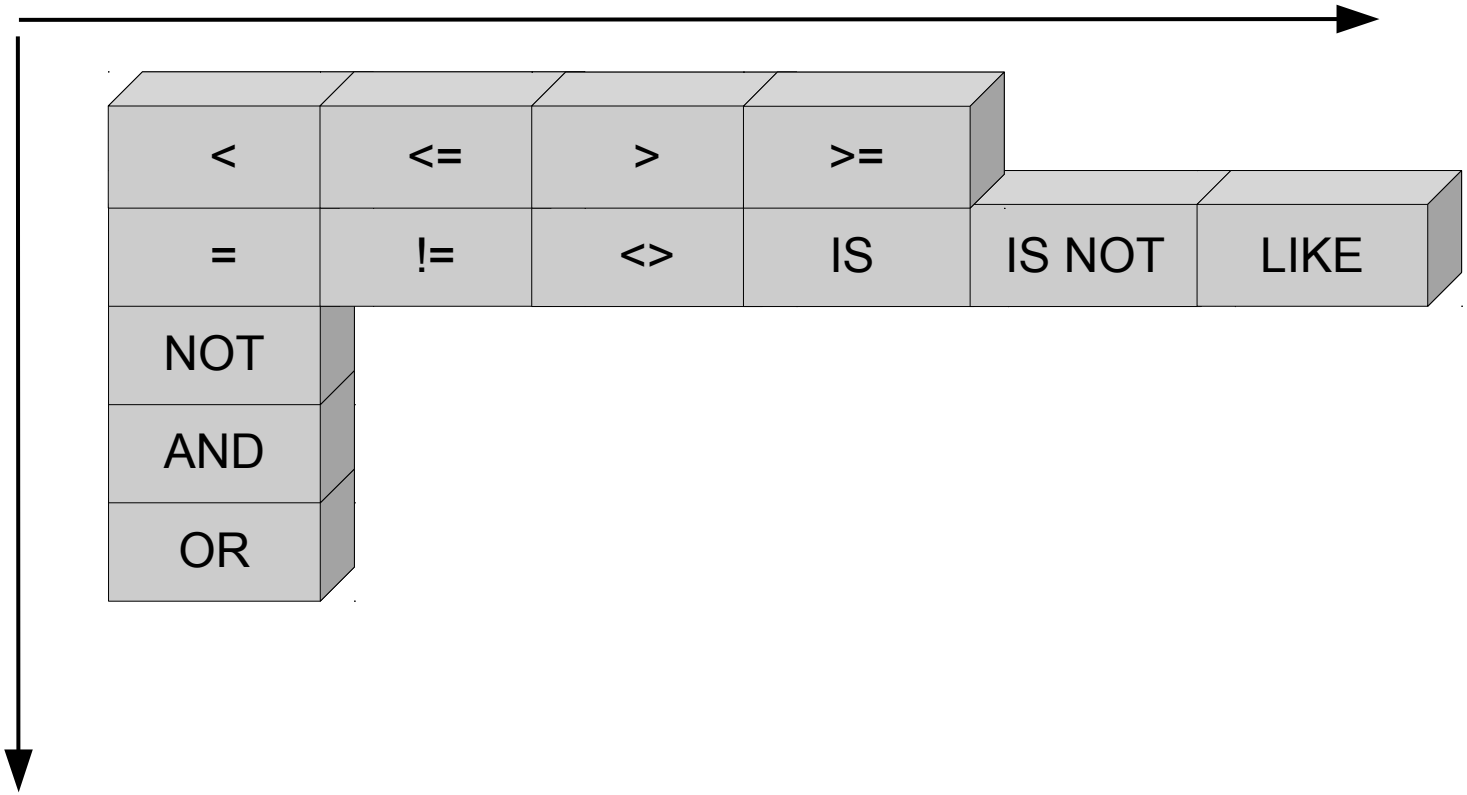
```
SELECT "Name", "GenreId", "Milliseconds"
FROM "Track"
WHERE (("GenreId" = 1)
       AND ("Milliseconds" > 300000));
```

- 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 "Track"
WHERE ((("GenreId" = 3)
       OR ("GenreId" = 7))
       AND ("Milliseconds" > 400000));
```

# Gewichtung der Operatoren



# Klammerung von Bedingungen

```
SELECT "Name", "GenreId", "Milliseconds"
FROM "Track"
WHERE ((("GenreId" = 3)
       OR ("GenreId" = 7))
       AND ("Milliseconds" > 400000));
```

- Mit Hilfe der runden Klammern werden Bedingungen zusammengefasst.
- Die Rangfolge der Operatoren kann durch die Klammerung verändert werden.
- Die runden Klammern dienen der besseren Lesbarkeit von Ausdrücken.

# Liste von Werten

```
("State" IN ('CA', 'WA', 'NY'))

(("State" LIKE 'CA')
  OR ("State" LIKE 'WA')
  OR ("State" LIKE 'NY'))
```

```
  ("GenreId" IN(2, 4, 6));

  (("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 Elemente in der Liste werden durch ein Kommata getrennt.

# Zwischen … und ...

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

SELECT * FROM "Track"
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.

# Datumsformate im SQL-Standard

- Eingabe: '1968-01-09'. 'yyyy-mm-dd'.
- Eingabe: '01/09/1968'. 'mm/dd/yyyy'.
- In der Ergebnistabelle: '1968-01-09'. 'yyyy-mm-dd'.

# Zeitformate im SQL-Standard

- '20:12:00'. 'hh:mm:ss'.
  Die Stunden werden in einem 24-Stunden-Format angegeben.
- '09:12:00 AM'.
  Als Stunde wird 9 Uhr vormittags gespeichert.
- '09:12:00 PM'.
  Als Stunde wird 9 Uhr nach dem Mittag (21:00) gespeichert.

# Beispiel

```
SELECT "LastName", "HireDate"
FROM "Employee"
WHERE ("HireDate" BETWEEN '2002-01-01'
                      AND '2002-01-31');
```

- Das Datenfeld hat einen Wert zwischen (`BETWEEN`) der Unter- und (`AND`) der Obergrenze.
- Die Datumswerte werden durch ein Apostroph am Anfang und Ende begrenzt.
- Die Literale können mit Hilfe von Funktionen berechnet werden.