# SQL Unterabfragen

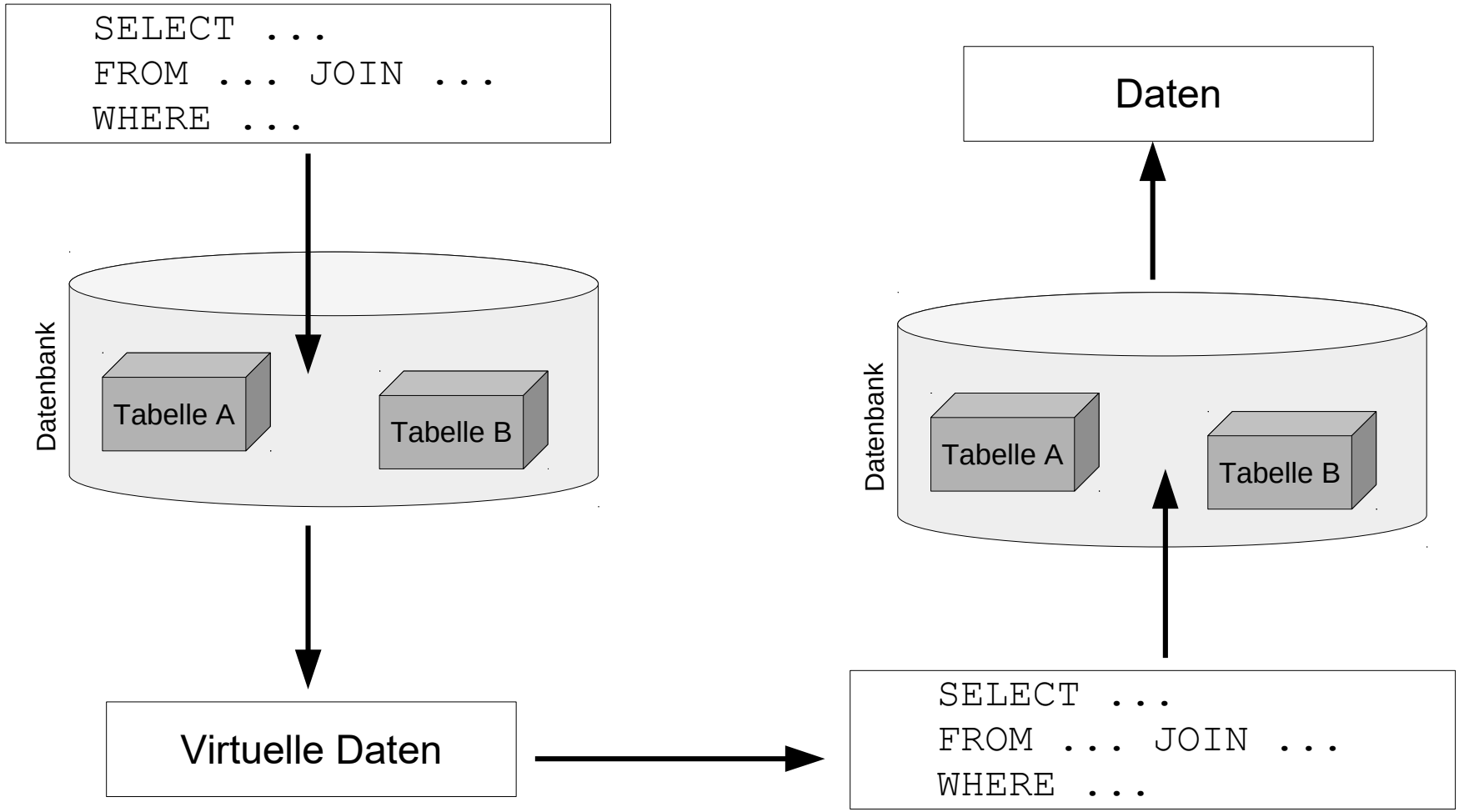
Welcher Kunde hat Bestellungen über die durchschnittliche Bestellmenge aufgegeben?

Welche Produkte sind den Kategorien a, b, … Zugeordnet?

# Unterabfragen

- Auswahlabfragen werden zur Prüfung von Werten in Bedingungen genutzt.
- Auswahlabfragen werden als Ersatz für Tabellennamen in der FROM- oder JOIN-Klausel genutzt.
- Je nach Nutzung darf die Unterabfrage exakt einen Wert oder beliebig viele Werte zurückgeben.

# Arbeitsweise



# Beispiel für die Nutzung

```
SELECT tracks.Name, tracks.Milliseconds
FROM tracks
WHERE (
  GenreID IN (

    SELECT genres.GenreID
    FROM genres
    WHERE (genres.Name IN ('Rock', 'Jazz'))
  )
);
```

# Hauptabfrage

```
SELECT tracks.Name, tracks.Milliseconds
FROM tracks;
```

- Die Hauptabfrage kann unabhängig von der Unterabfrage ausgeführt werden.
- Die Hauptabfrage liefert ein nicht gefiltertes Ergebnis
- Datensätze in der Hauptabfrage können sortiert werden.

# Unterabfrage

```
SELECT genres.GenreID
FROM genres
WHERE (genres.Name IN ('Rock', 'Jazz'))
```

- Die Unterabfrage kann als Auswahlabfrage ausgeführt werden.
- Das Ergebnis der Unterabfrage ist immer unsortiert. Eine Sortierung wird nicht benötigt.
- Unterabfragen geben entsprechend der Nutzung x Datenfelder zurück.
- Unterabfragen werden immer durch runde Klammern begrenzt.

# … in einer WHERE-Klausel

```
SELECT tracks.Name, tracks.Milliseconds
FROM tracks
WHERE (
  GenreID IN (
     SELECT genres.GenreID
     FROM genres
     WHERE (genres.Name IN ('Rock', 'Jazz'))
  )
);
```

# Erläuterung

- Die Unterabfrage liefert die Elemente der Liste In() in einer Bedingung.
- Die Unterabfrage liefert x Datensätze. Jeder Datensatz der Unterabfrage stellt einen Wert in der Liste dar.
- Pro Datensatz wird der Wert eines Datenfeldes benötigt. In diesem Beispiel wird der Primärschlüssel zurückgegeben.

# Duplikatsuche

```
SELECT artists.Name
FROM artists
WHERE (artists.artistId IN
        (
         SELECT albums.ArtistId
         FROM albums
         GROUP BY albums.ArtistId
         HAVING Count(*) > 1
        )
);
```

# Erläuterung

- Die Liste `IN()` wird durch eine Auswahlabfrage gebildet.
- Als Listenelement wird ein Wert pro Datensatz benötigt. Die Anzahl der Elemente in der Liste ist abhängig von der Anzahl der Datensätze in der Ergebnistabelle der Unterabfrage.
- In der Unterabfrage werden die Elemente gruppiert. In Abhängigkeit des Datenfeldes `albums.ArtistId` werden die Daten zusammengefasst.
- Es werden nicht alle gruppierten Datensätze in der Liste `IN()` genutzt. Der Künstler muss mindestens zwei Alben produziert haben (`Count(*) > 1`).

# Fehler: Schachtelung von Aggregatfunktionen

```
SELECT avg(sum(tracks.Milliseconds))
FROM tracks
GROUP BY tracks.AlbumId;
```

- Eine Schachtelung von Aggregatfunktionen ist nicht möglich.
- Das Ergebnis einer Aggregatfunktion kann in der Ergebnistabelle angezeigt werden, aber nicht als Grundlage für weitere Berechnungen genutzt werden.

# Lösung

```
SELECT avg(musikstueck.dauer)
FROM (
      SELECT sum(tracks.Milliseconds) AS dauer
      FROM tracks
      GROUP BY tracks.AlbumId
)
AS musikstueck
```

# Erläuterung

- Die Anweisung `FROM tracks` gibt die Datenquelle an. Der Tabellenname `tracks` wird durch eine SQL-Anweisung ersetzt.
- Die Unterabfrage wird durch die runden Klammern geklammert.
- Mit Hilfe des Schlüsselwortes `AS` wird der Unterabfrage ein Alias zugewiesen.
- Die Daten aus der Ergebnistabelle werden in der Hauptabfrage weiter verarbeitet.

# Unterabfrage

```
SELECT sum(tracks.Milliseconds) AS dauer
FROM tracks
GROUP BY tracks.AlbumId
```

- In diesem Beispiel wird die Summe von allen Tracks für ein Album berechnet. Die Datensätze werden nach der `AlbumId` gruppiert.
- Mit Hilfe des Schlüsselwortes `AS` wird dem berechneten Feld ein Alias zugewiesen.

# Hauptabfrage

```
SELECT avg(musikstueck.dauer)
FROM (
      SELECT sum(tracks.Milliseconds) AS dauer
      FROM tracks
      GROUP BY tracks.AlbumId
)
AS musikstueck
```

- Der Durchschnitt von der Gesamtdauer pro Album wird berechnet.
- Das Feld, welches zur Berechnung genutzt wird, wird mit Hilfe eines qualifizierten Namens angegeben. Als Datenquelle wird der Alias der Unterabfrage genutzt. Die Angabe des Datenfeldes bezieht sich auf das Alias des Datenfeldes in der Unterabfrage.

# Weiteres Beispiel

```
SELECT
    albums.Title,
    MIN(t.millis) AS LaengeAlbum
FROM (
    SELECT
       SUM(tracks.Milliseconds) AS millis,
       tracks.AlbumID
    FROM tracks
    GROUP BY tracks.AlbumId )t

INNER JOIN albums
ON (t.AlbumId = albums.AlbumId)

ORDER BY MIN(t.millis) DESC;
```

# Erläuterung

- Dem Schlüsselwort `FROM` folgt eine Unterabfrage als Datenquelle.
- Die Unterabfrage wird durch die runden Klammern geklammert.
- Der Unterabfrage wird der Alias `t` zugewiesen. Der runden Klammer folgt direkt der Name. Das Schlüsselwort `AS` wird nicht benötigt.
- Die Daten aus der Ergebnistabelle werden in der Hauptabfrage weiter verarbeitet.

# Unterabfrage

```
SELECT
      SUM(tracks.Milliseconds) AS millis,
      tracks.AlbumID
FROM tracks
GROUP BY tracks.AlbumId
```

- In diesem Beispiel wird die Summe von allen Tracks für ein Album berechnet. Mit Hilfe einer Aggregatfunktion werden die Werte des Feldes `tracks.Milliseconds` addiert.
- Mit Hilfe des Schlüsselwortes `AS` wird dem berechneten Feld ein Alias zugewiesen.
- Die Datensätze werden nach der `AlbumId` gruppiert.

# Hauptabfrage

```
SELECT
    albums.Title,
    MIN(t.millis) AS LaengeAlbum
FROM ( ... )t

INNER JOIN albums
ON (t.AlbumId = albums.AlbumId)

ORDER BY MIN(t.millis) DESC;
```

# Nutzung der Aggregatfunktion

```
SELECT
    albums.Title,
    MIN(t.millis) AS LaengeAlbum
```

- In der Unterabfrage wird für das berechnete Datenfeld ein Alias vergeben. Dieser Alias wird der Aggregatfunktion `Min()` in der Hauptabfrage als Parameter in den runden Klammern übergeben.
- Das, in der Hauptabfrage berechnete Feld wird wiederum ein Alias mit dem Schlüsselwort `AS` zugewiesen.

# Verknüpfung der Tabellen

```
FROM ( ... )t

INNER JOIN albums
ON (t.AlbumId = albums.AlbumId)
```

- In diesem Beispiel wird eine INNER JOIN – Verknüpfung genutzt.
- Die Unterabfrage wird mit der Tabelle `albums` verknüpft. Diese Tabelle könnte auch durch eine weitere Unterabfrage ersetzt werden.
- Das Feld `t.AlbumId` in der Verknüpfungsbedingung muss in der Unterabfrage definiert werden. Der qualifizierte Name setzt sich aus dem Alias der Unterabfrage und dem Namen des angegebenen Fremdschlüssels in der Unterabfrage zusammen.

# Sortierung nach ...

```
ORDER BY MIN(t.millis) DESC;
```

- Falls das Feld mit einer Aggregatfunktion zusammengefasst wurde, muss dieses entsprechend in der Feldliste angegeben werden.

# Synchronisierte Unterabfrage

```
SELECT t2.Name, t2.Milliseconds
FROM tracks AS t2
WHERE (
  t2.Milliseconds > (

    SELECT AVG(t1.Milliseconds) AS Durchschnitt
    FROM tracks AS t1
  )
)
ORDER BY t2.Milliseconds DESC;
```

# Erläuterung

- Die Hauptabfrage und die Unterabfrage nutzen dieselbe Tabelle.
- Für die Tabelle in der Haupt- und Unterabfrage wird mit Hilfe des Schlüsselwortes `AS` jeweils ein Alias vergeben. In Abhängigkeit des Alias werden der Haupt- und der Unterabfrage die entsprechenden Datenfelder zugeordnet.
- Bei der Nutzung von gleichen Tabellen-Bezeichnung in der Haupt- und Unterabfrage werden Ausdrücke, die in der Unterabfrage nicht aufgelöst werden können, an die Hauptabfrage weitergereicht.

# Korrelierte (verbundene) Unterabfrage

```
SELECT Kunde.CustomerID, Kunde.Company, Bestellung.Total

FROM customers AS Kunde INNER JOIN invoices AS Bestellung
ON (Kunde.CustomerID = Bestellung.CustomerID)

WHERE (
   (NOT(Kunde.Company Is Null))
   AND
   (
      Bestellung.Total >
      (
         SELECT AVG(MaxBestellung.Total) AS MaxGesamt
         FROM invoices AS MaxBestellung
         WHERE ( MaxBestellung.CustomerID = Kunde.CustomerID)
      )
   )
)

ORDER BY Kunde.Company, Bestellung.Total DESC;
```

# Hauptabfrage

```
SELECT Kunde.CustomerID, Kunde.Company, Bestellung.Total

FROM customers AS Kunde

INNER JOIN invoices AS Bestellung
ON (Kunde.CustomerID = Bestellung.CustomerID)

ORDER BY Kunde.Company, Bestellung.Total DESC;
```

- In der Hauptabfrage werden in der Feldliste qualifizierte Namen genutzt.
- Die Tabellen in der Verknüpfung bekommen ein Alias zugewiesen.
- Die Daten der Ergebnistabelle werden sortiert.

# WHERE-Bedingung der Hauptabfrage

```
WHERE (
   (NOT(Kunde.Company Is Null))
   AND
   (
      Bestellung.Total > ()
   )
)
```

- Mit Hilfe von `AND` werden zwei Bedingungen verknüpft. Beide Bedingungen müssen wahr sein.
- In diesen Beispiel müssen den Kunden eine Firma zugeordnet sein (`NOT(Kunde.Company Is Null)` oder `NOT(Kunde.Company Is NOT Null)`)
- Die Gesamtsumme der Bestellung muss größer als ein, in der Unterabfrage berechneter Wert sein.

# Unterabfrage

```
    Bestellung.Total >
    (
       SELECT AVG(MaxBestellung.Total) AS MaxGesamt
       FROM invoices AS MaxBestellung

       WHERE ( MaxBestellung.CustomerID =
               Kunde.CustomerID)
    )
```

- In dieser Unterabfrage wird der Aggregatfunktion `AVG()` ein Datenfeld aus der Datenquelle. Der berechnete Wert wird ein Alias zugewiesen.
- Die Datenquelle folgt dem Schlüsselwort `FROM`. Die Datenquelle bekommt ein Alias zugewiesen.

# WHERE-Bedingung in der Unterabfrage

```
    Bestellung.Total >
    (
       SELECT AVG(MaxBestellung.Total) AS MaxGesamt
       FROM invoices AS MaxBestellung

       WHERE ( MaxBestellung.CustomerID =
               Kunde.CustomerID)
    )
```

- Die Berechnung wird ausgeführt, wenn die ID des Kunden aus der Datenquelle der Unterabfrage gleich der ID des Kundens aus der Hauptabfrage ist.
- Die Unterabfrage kann nicht ohne die Hauptabfrage ausgeführt werden.

# Berechnung von neuen Schlüsselwerten

```
SELECT

(SELECT COUNT(*)
 FROM customers AS anzahlKunde
 WHERE anzahlKunde.LastName <= newTable.LastName)
|| '_' || lower(substr(newTable.LastName,1, 3))
AS row_Num,

newTable.CustomerId,

newTable.FirstName || ' ' || newTable.LastName
AS customerName

FROM Customers AS newTable
ORDER BY newTable.LastName, newTable.FirstName;;
```

# Erläuterung

- Ein Datenfeld in der Feldliste wird durch eine Unterabfrage ersetzt.
- Mit Hilfe der Unterabfrage wird ein Wert berechnet. Dieser Wert wird in der Tabelle in der angegebenen Spalte angezeigt.
- Mit Hilfe des Schlüsselwortes `AS` wird dem Datenfeld ein passender Alias zugewiesen.

# Berechnetes Feld

```
(    )
|| '_' || lower(substr(newTable.LastName,1, 3))
AS row_Num
```

- Der berechnete Wert in Abhängigkeit der Unterabfrage wird mit dem Unterstrich und den ersten drei Buchstaben des Nachnames des Kunden verknüpft.
- Dem berechneten Feld wird ein Alias zugewiesen.

# Unterabfrage

```
SELECT COUNT(*)
FROM customers AS anzahlKunde

WHERE anzahlKunde.LastName <= newTable.LastName
```

- Die Aggregatfunktion `Count(*)` gibt die Anzahl der Datensätze zurück.
- Der Name des Kunden in der Zusammenfassung müssen vor dem anzuzeigenden Namen liegen. Falls Unicode-Zeichen verwendet werden, liegt A vor B und A vor a und so weiter.