



# Datenbanken und SQL

(Woche 3 - Tag 5)

# Agenda

## Subselect (Unterabfrage)

- Einzelner Rückgabewert
  - Definition + Motivation
  - Beispiel
- Mehrere Rückgabewerte
  - Motivation
  - IN-Operator
  - Beispiel
- Übungen

# Subselect

(Einzelner Rückgabewert)

# Definition + Motivation

- Eine Abfrage **A** heißt „verschachtelt“ (oder auch „äußere Abfrage“), wenn zu deren Abarbeitung zunächst das Ergebnis einer weiteren Abfrage **B** notwendig ist.
- Eine solche Abfrage **B** wird als **Unterabfrage**, bzw. **Subselect** bezeichnet.
- Um den Lösungsansatz zu erläutern, werden wir zunächst das Resultat von **B** ermitteln, um dieses anschließend im Code von **A** als **Konstante** zu nutzen.
- Da sich der Rückgabewert von **B** jedoch nach einem **Update der Datenbank** ändern könnte, werden wir abschließend die Konstante durch den Subselect **B** ersetzen.

# Beispiel

**Aufgabenstellung:** (wie üblich beziehen wir uns auf die Datenbank „Geld\_her“)

Gesucht werden die Namen jener Produkte, die billiger als der Durchschnittspreis (aller Produkte) sind.

1. Schritt -> Wir ermitteln zunächst den Durchschnittspreis aller Produkte (Subselect):

```
SELECT AVG(Euro_Preis) FROM Produkt;
```

| AVG(produkt.Euro_Preis) |
|-------------------------|
| 202.245000              |

2. Schritt -> Anschließend können wir alle Produkte ermitteln, die weniger als **202,245000** kosten:

```
SELECT Produkt_Name FROM Produkt  
WHERE Euro_Preis < 202.245000;
```

3. Schritt -> Abschließend ersetzen wir die **Konstante** durch den **Subselect** (der in Klammern und ohne Semikolon notiert werden muss!)

```
SELECT Produkt_Name FROM Produkt  
WHERE Euro_Preis < (SELECT AVG(Euro_Preis) FROM Produkt);
```

| Produkt_Name    |
|-----------------|
| tool 2.0        |
| tool 3.1        |
| solver 1000     |
| solver premium  |
| Do IT edition 1 |

# Subselect

(Mehrere Rückgabewerte)

# Motivation

- Gelegentlich haben wir es mit verschachtelten Abfragen zu tun, bei denen die äußere Abfrage nicht bezüglich eines einzigen Wertes formuliert werden kann, sondern nur bezüglich einer **Menge von Werten** (des selben Attributs).
- Entsprechend werden in diesem Fall Subselects benötigt, die nicht nur einen einzigen, sondern **mehrere Werte** (des selben Attributs) ausgeben.
- Ferner wird ein Operator benötigt, der die aus der Mengenlehre bekannte Relation „*ist Element von*“ zur Verfügung stellt. Dies wird der „**IN**“-Operator sein, den wir zunächst kurz vorstellen, bevor wir ihn im Zusammenhang mit verschachtelten Abfragen verwenden werden.

# IN-Operator

Der IN-Operator kann als Abkürzung für mehrgliedrige OR-Verknüpfungen verwendet werden, was zum einen komfortabel ist, und zum anderen die Lesbarkeit des Codes verbessern kann – ein **Beispiel**:

Nachname aller Kunden mit ID=**1** oder **2** oder **5** oder **7**:

Lösung mit den bereits bekannten Techniken:

```
SELECT Nachname From Kunde  
WHERE Kunde_ID=1 OR Kunde_ID=2 OR Kunde_ID=5 OR Kunde_ID=7;
```

Lösung mittels IN-Operator:

```
SELECT Nachname From Kunde  
WHERE Kunde_ID IN(1,2,5,7);
```

| Nachname |
|----------|
| Rot      |
| Deise    |
| Hahn     |
| Kaufnix  |



# Beispiel

**Aufgabenstellung:** (wie üblich beziehen wir uns auf die Datenbank „Geld\_her“)

Gesucht werden alle Kalenderdaten, an denen mindestens 1 Produkt bestellt wurde, das auch schon auf der Abrechnung 1 bestellt (bzw. gekauft) worden ist. Ausgabe chronologisch sortiert nach Kalenderdatum.


Wir formulieren zunächst den entsprechenden **Subselect**:

```
SELECT DISTINCT Produkt_ID  
FROM Abrechnung_Produkt  
WHERE Abrechnung_ID=1;
```

Für die verschachtelte Abfrage benötigen wir dann den folgenden Code, bei dem der **IN-Operator** nun alternativlos ist:

```
SELECT DISTINCT Datum FROM Abrechnung, Abrechnung_Produkt  
WHERE Abrechnung.Abrechnung_ID = Abrechnung_Produkt.Abrechnung_ID  
AND Produkt_ID IN  
    (  
        SELECT DISTINCT Produkt_ID  
        FROM Abrechnung_Produkt  
        WHERE Abrechnung_ID=1  
    )  
ORDER BY Datum;
```

# Gemeinsame Übung („Live-Coding“) -> A\_03\_05\_01



## Aufgabe\_03\_05\_01

Formulieren Sie bitte entsprechende SQL-Anweisungen für folgende Aufgabestellungen:

- Es sollen alle Kunden (ID, Vorname, Nachname) ausgegeben werden, die den gleichen Nachnamen wie Kunde 3 haben.
- Identische Ausgabe wie in a), nun aber mit Ausnahme von Kunde 3.
- Es sollen alle „teuersten Produkte“ (ID, Produktname und Preis) ausgegeben werden.

**Hinweise:**


- Es sei noch einmal daran erinnert, dass es mehrere Produkte geben kann, die sich gemeinsam den Titel „teuerstes Produkt“ teilen, da sie untereinander den identischen Preis haben, aber teurer als alle anderen Produkte sind.
- Diese Aufgabe konnten wir bisher weder mittels **GROUP BY** lösen (denn wir konnten dem maximalen Preis nicht „dem“ entsprechenden Produktnamen zuordnen) ...  
... noch mittels **ORDER BY Euro\_Preis DESC LIMIT x** (denn wir kannten x nicht).

- Es sollen alle Kunden (Vorname, Nachname) ausgegeben werden, die bisher weniger Produkte bestellten, als alleine auf der Abrechnung 3 bestellt worden sind.
- Es soll die Anzahl der Produkte ermittelt werden, die billiger sind als die Gesamtbestellsumme von Kunde 5.

WBS TRAINING AG  
Lorenzweg 5  
D-12099 Berlin  
Amtsgericht Berlin HRB 68531  
Sitz der Gesellschaft: Berlin

Vorstand:  
Heinrich Kronbichler,  
Joachim Giese  
Aufsichtsrat (Vorsitz): Dr. Daniel Stadler  
USt-IDNr.: DE 209 768 248

GLS Gemeinschaftsbank eG  
IBAN: DE18 4306 0967 1146 1814 00  
BIC: GENODEM33GLS



GLS zertifiziert nach  
DIN EN ISO 9001:2015 Reg. Nr. 01100010001  
Zertifizierung nach ISO 14001 Reg. Nr. 01100010001

# Vielen Dank für Ihre Aufmerksamkeit!

