10. Datenabfrage mit Unterabfragen

In diesem Abschnitt zeigen wir die Möglichkeit, Ergebnisse von Abfragen direkt in anderen Anweisungen zu verwenden.

Wir werden folgende Varianten kennen lernen:

- Unterabfragen, die einen Wert liefern.
- ALL-, ANY und IN-Unterabfragen
- EXIST-Unterabfragen

10.1. Unterabfragen, die einen Wert liefern

Wir benötigen Unterabfragen, um Anweisungen mit Werten zu formulieren, die selbst erst über eine Datenbankabfrage ermittelt werden. Ein Beispiel: „Zeige die Kunden, die in derselben Stadt wohnen, wie der Kunde mit der Kundennummer 106." Bei der Formulierung der Abfrage ist der Ort nicht bekannt. Um ihn zu ermitteln, könnten wir zunächst die Abfrage formulieren:

```
SELECT Ort
    FROM T_Kunden
    WHERE p_kunden_nr = 106;
```

Den Ortsnamen könnten wir dann als Konstante in der Abfrage einsetzen, die die eigentlich erwünschten Ergebnisse liefert:

```
SELECT p_kunden_nr, nname, vname
    FROM T_Kunden
    WHERE ort = 'Kayhude';
```

An jeder Stelle in einer SQL-Anweisung, an der ein Ausdruck stehen kann, der einen Wert berechnet, kann statt dessen eine spezielle Unterabfrage eingesetzt werden, die als Ergebnis einen Wert hat. Eine Unterabfrage, die für einen Wert stehen darf, ist hierbei eine in Klammern gesetzte Abfrage-Anweisung, die zwei Bedingungen zu erfüllen hat:

1. Die SELECT-Klausel muss genau ein Attribut oder virtuelles Attribut enthalten.
2. Das Ergebnis der SELECT -Klausel darf nicht mehr als ein Tupel enthalten. Somit können wir unser obiges Problem wie folgt formulieren:

```
SELECT p_kunden_nr, nname, vname
   FROM T_Kunden
   WHERE ort =
   (
        SELECT ort
        FROM T_Kunden
        WHERE p_unden_nr = 106
   );
```

Beispiele

Zeige die Abweichungen vom durchschnittlichen Listenpreis

```
SELECT p_artikel_nr, Listenpreis, Listenpreis - (SELECT AVG (listenpreis) FROM T_Artikel) AS Abweichung
    FROM T_Artikel;
```

Zeige alle Artikel, die teurer sind als der Durchschnitt

```
SELECT p_artikel_nr, listenpreis FROM T_Artikel
    WHERE listenpreis > (SELECT AVG (listenpreis) FROM T_Artikel);
```

Zeige den (oder die) teuersten Artikel

```
SELECT p_rtikel_nr, listenpreis FROM T_Artikel
    WHERE listenpreis = (SELECT MAX (listenpreis) FROM T_Artikel);
```

C01-1

Ermitteln Sie den günstigsten Artikel!

(Ausgabe: Artikelnummer, Bezeichnung, Listenpreis)

C01-2

Welcher Kunde hat die letzte Bestellung aufgegeben?

Hinweis: Zur Lösung benötigen Sie in der Hauptabfrage einen Verbund von T_Kunden und T_Bestellungen.

(Ausgabe: Kundennummer, Nachname, Vorname, Bestellnummer)

10.2. **Unterabfragen**, die mehr als ein Tupel liefern

Liefert die Unterabfrage eine Ergebnisrelation mit mehr als einem Tupel, so ist der Einsatz von Mengenoperatoren in der WHERE-Klausel der Hauptabfrage erforderlich. Folgende Operatoren werden zur Formulierung von Bedingungen angeboten, wobei ✶ für einen der Vergleichsoperatoren (=, < >, >, > =, < =, <) steht:

IN	prüft, ob ein Wert in der Ergebnismenge der Unterabfrage enthalten ist.
EXISTS	prüft, ob die Unterabfrage mindestens ein Tupel erbringt, das der Bedingung genügt.
✶ **ANY**	prüft, ob die Bedingung für irgendein Tupel der Unterabfrage zutrifft.
✶ **SOME**	SOME ist ein anderer Name für ANY.
✶ **ALL**	ALL prüft, ob die Bedingung für alle Tupel der Unterabfrage zutrifft.

Wir erläutern im Folgenden die einzelnen Operatoren an Beispielen.

► IN

Gesucht sind die Bestellungen, in denen der Artikel mit der Nummer K003 betroffen ist. Wir ermitteln in der Unterabfrage die Bestellnummern der Positionen, bei denen die Artikelnummer den Wert K003 hat. Dies können mehrere Positionen sein.

```
SELECT p_bestell_nr, bestelldatum
    FROM T_Bestellungen
    WHERE p_bestell_nr IN (SELECT p_f_Bestell_nr FROM T_artikel_Bestellungen WHERE
    p_f_artikel_nr = 'K003');
```

Zur Erläuterung betrachten wir die Unterabfrage einmal für sich:

```
SELECT p_f_bestell_nr
    FROM T_artikel_Bestellungen
    WHERE p_f_artikel_nr = 'K003'
```

Sie liefert mehrere (zwei) Ergebnisse: 960151 und 960152

Der IN-Operator ist bereits im Zusammenhang mit einfachen Abfragen an eine Relation vorgestellt worden. Bei der dort vorgestellten Form ist aber vorausgesetzt, dass die Menge der Vergleichswerte bekannt ist. In diesem Fall hätten wir die Hauptabfrage so formulieren können:

```
SELECT p_bestell_nr, bestelldatum
    FROM T_Bestellungen
    WHERE p_bestell_nr IN (960151, 960152);
```

Wenn die Unterabfrage eine leere Menge als Ergebnis liefert, dann ist die in der WHERE-Klausel der Hauptabfrage formulierte Bedingung nicht erfüllt. Sie erhält dann den Wahrheitswert FALSE.

► ANY

ANY muss in Verbindung mit den Vergleichsoperatoren (=, < >, < =, > =, <, >) eingesetzt werden. Der Test auf Gleichheit liefert dabei dasselbe Ergebnis wie der IN-Operator.

Wir suchen noch einmal die Bestellungen, in denen der Artikel K003 betroffen ist. Die Formulierung mit ANY lautet

```
SELECT p_bestell_nr, bestelldatum
    FROM T_Bestellungen
    WHERE p_bestell_nr = ANY (SELECT p_f_bestell_nr FROM T_Artikel_Bestellungen WHERE
    p_f_artikel_nr = 'K003');
```

Mit ANY können aber auch Abfragen formuliert werden, in denen nicht die Gleichheit von Werten geprüft wird. Wir fragen nach den Artikeln, die nicht auf dem Lagerplatz 2 gelagert sind und teurer sind als irgendein Artikel von Lagerplatz 2.

```
SELECT p_artikel_nr, listenpreis
    FROM T_Artikel
    WHERE  lagerplatz <> 2
            AND listenpreis > ANY (SELECT listenpreis FROM T_Artikel WHERE lagerplatz = 2);
```

Zum besseren Verständnis hier das Ergebnis der Unterabfrage:

```
SELECT listenpreis FROM T_Artikel WHERE lagerplatz = 2;
```

Sie liefert 6 Ergebnisse: 98,50 | 112,80 | 65,70 | 76,00 | 0,98 | 1,72

Die Bedingung der Hauptabfrage ist für alle Artikel erfüllt, die nicht auf Lagerplatz 2 liegen und teurer sind als 98 Cent, denn dies ist der kleinste Listenpreis bezogen auf Lagerplatz 2. Es reicht also aus, wenn die Bedingung der Hauptabfrage für irgendein (ANY) Ergebnis der Unterabfrage den Wahrheitswert TRUE liefert.

► ALL

Der ALL-Operator wird eingesetzt, wenn die Bedingung für alle Ergebnisse der Unterabfrage erfüllt sein muss.

Wir suchen die Artikel, deren Listenpreis größer ist als die Listenpreise aller Artikel aus Lager 5. Die Unterabfrage ist identisch mit der aus dem letzten Beispiel.

```
SELECT p_artikel_nr, listenpreis
    FROM T_Artikel
    WHERE  listenpreis > ALL (SELECT listenpreis FROM T_Artikel WHERE lagerplatz = 5);
```

Zum besseren Verständnis hier das Ergebnis der Unterabfrage:

```
SELECT Listenpreis FROM T_Artikel WHERE lagerplatz = 5;
```

Sie liefert 2 Ergebnisse: 6,35 | 8,35

Die Bedingung der Hauptabfrage ist für alle Artikel erfüllt, die teurer sind als 8,35 €, denn dies ist der höchste Listenpreis bezogen auf Lagerplatz 5. Die Bedingung der Hauptabfrage muss für alle (ALL) Ergebnisse der Unterabfrage den Wahrheitswert TRUE liefert.

Die letzte Abfrage ist im übrigen auch ohne ALL formulierbar. Wenn ein Artikel teurer ist als alle Artikel von Lager 5, dann ist sein Listenpreis größer als der größte Listenpreis von Artikeln aus Lager 5.

```
SELECT p_artikel_nr, listenpreis
    FROM T_Artikel
    WHERE  listenpreis > (SELECT MAX(listenpreis) FROM T_Artikel WHERE lagerplatz = 5);
```

Abfragen unter Verwendung von Unterabfragen können meist auf verschiedene Weise formuliert werden. Im Allgemeinen wird man die kürzere Formulierung bevorzugen, die meist auch die verständlichere ist.

► EXISTS

Der EXISTS-Operator prüft, ob in der Ergebnismenge der Unterabfrage mindestens ein Tupel enthalten ist.

Wir greifen noch einmal eine im Zusammenhang mit dem IN-Operator bereits behandelte Abfrage auf: „*Gesucht sind die Bestellungen, in denen Artikel mit der Nummer K003 enthalten ist*“.

Mit dem EXISTS-Operator kann dieselbe Frage beantwortet werden, wenn wir sie etwas anders formulieren: „*Gesucht sind die Bestellungen, in denen eine Position existiert, in der die Artikelnummer den Wert K003 hat*“.

```
SELECT p_bestell_nr, bestelldatum
    FROM T_Bestellungen AS B
    WHERE EXISTS
    (SELECT * FROM T_Artikel_Bestellungen AS P WHERE P.p_f_artikel_nr = 'K003' AND
    P.p_f_bestell_nr = B.p_bestell_nr);
```

Um diese Aufgabe mit dem IN-Operator zu lösen, hatten wir in der Unterabfrage die Bestellnummern der Positionen ermittelt, bei denen die Artikelnummer den Wert K003 hat. In der Hauptabfrage wurden die Bestellungen selektiert, deren Bestellnummern in der Ergebnismenge enthalten waren. Die Unterabfrage war unabhängig von der Hauptabfrage formulierbar.

Im Unterschied dazu muss bei der Verwendung von EXISTS in der Unterabfrage eine Bedingung formuliert werden, die ein bestimmtes in der Hauptabfrage behandeltes Tupel mit einbezieht. Anders ausgedrückt: Die Frage

```
EXISTS (SELECT * FROM T_Artikel_Bestellungen WHERE p_f_artikel_nr = 'K003')
```

liefert immer dann TRUE, wenn der Artikel K003 in irgendeiner Position auftritt. Also müssen wir die Untersuchung in der Unterabfrage auf die Positionen beschränken, die zu einer gerade in der Hauptabfrage behandelten Bestellung gehören. Die Bedingung lautet daher:

```
EXISTS (SELECT * FROM T_Artikel_Bestellungen WHERE p_f_artikel_nr = 'K003' AND
P.p_f_bestell_nr = B.p_bestell_nr)
```

Eine Unterabfrage wie im letzten Beispiel wird korreliert genannt, da sie nicht unabhängig von der Hauptabfrage bearbeitet werden kann. Die Bedingung in der Unterabfrage bezieht sich auf eine Relation, die in der FROM-Klausel der Hauptabfrage angesprochen wird und nicht in der FROM-Klausel der Unterabfrage.

Dies kann nur funktionieren, wenn die Attributwerte der Hauptabfrage in der Unterabfrage bekannt sind. Bei SQL gilt die Regel, dass in einer Unterabfrage sämtliche Attribute der sie

umfassenden Oberabfragen mit Namen und Wert referenzierbar sind. Da eine Unterabfrage wiederum Unterabfragen enthalten kann, umfasst die Reichweite eines Attributs eventuell mehrere Stufen. Taucht ein Attribut in der Ober- und Unterabfrage mit gleichem Namen auf, so muss es mit vorangestelltem Aliasnamen qualifiziert werden.

In einem weiteren Beispiel suchen wir alle Bestellungen, die Artikel aus Lager 2 enthalten. Hierzu muss in der Unterabfrage ein Verbund der Relationen T_Artikel und T_Position hergestellt werden.

```
SELECT *
    FROM T_Bestellungen AS B
    WHERE EXISTS (SELECT *
        FROM T_Artikel_Bestellungen AS P INNER JOIN T_Artikel AS A ON A.p_artikel_nr =
        P.p_f_artikel_nr
        WHERE P.p_f_bestell_nr = B.p_bestell_nr AND A.lagerplatz = 2);
```

Im nächsten Beispiel werden mit Hilfe des EXISTS-Operators die Kunden ermitteln, die noch keine Bestellung aufgegeben haben:

```
SELECT p_kunden_nr, vname, nname
    FROM T_Kunden AS K
    WHERE NOT EXISTS (SELECT * FROM T_Bestellungen AS B WHERE K.p_kunden_nr =
    B.f_kunden_nr);
```

C02-1

Welche Bestellungen sind nicht von Kunden aus Kayhude?

(Ausgabe: Bestellnummer)

C02-2

Welche Artikel haben einen höheren Listenpreis als alle Artikel, deren Artikelnummer mit G anfängt?

Hinweis: Verwenden Sie in der Bedingung der Unterabfrage den Operator LIKE.

(Ausgabe: Artikelnummer, Bezeichnung, Listenpreis)

C02-3

Welche Kunden haben schon irgendwann einmal vor der letzten an sie gerichteten Werbeaktion eine Bestellung aufgegeben?

Hinweis: Hier ist eine korrelierte Unterabfrage erforderlich, bei der nur jeweils die Bestellungen ausgewertet werden, die der betreffende Kunde aufgegeben hat.

(Ausgabe: Kundennummer, Vorname, Nachname)

C02-4

Welche Kunden haben kein Girokonto?

(Ausgabe: Kundennummer, Vorname, Nachname)