LS4-6: SQL-DQL (INNER JOIN)

# Situation

Die Datenbank für den Online-Shop der Höllental-Senfmühle ist nun fertig gestellt und alle Daten migriert. Bevor wir den Auftrag abschließen und die Datenbank an den Auftraggeber übergeben, stehen noch etwaige Funktionstests an.

Liebe Kolleginnen und Kollegen, die ersten Funktionstests sind durchgeführt und so weit erfolgreich. Nun steht die zweite Stufe des Funktionstests an, in der überprüft werden soll, ob alle Tabellen auch richtig miteinander verknüpft sind und damit auch die richtigen Abfrageergebnisse ausgegeben werden.

Prüfen Sie die Funktionsfähigkeit anhand folgender Abfragen.

1. Liste aller Kunden mit Vornamen, Nachnamen und E-Mail-Adresse, bei denen noch unbezahlte Rechnungen vorliegen.
2. Liste aller Artikel mit Bezeichnung und Firmenname der Lieferer „Rein & Mir KG“, SUN-Senf GmbH“ und „Groebert GmbH“, sortiert nach dem Firmennamen

Sofern die Ergebnisse korrekt sind, legen Sie bitte für die 1. Abfrage (unbezahlte Rechnungen) eine Standardabfrage an, welche bei Bedarf ausgeführt werden kann, ohne das SQL-Statement schreiben zu müssen.

# Handlungsaufträge

1. Informieren Sie sich über die SQL-Befehle der ‚DQL‘ in Bezug auf INNER JOIN (Abfrage über mehrere Tabellen).

2. Bearbeiten Sie die Datenbank der Höllental-Senfmühle und führen Sie die SQL-Befehle des Arbeitsauftrags aus. Das Datenmodell findet sich in LS04-1 im Dokument LF7\_LS4-1\_INFO\_Datenmodell.docx.

4. Informieren Sie sich im Internet, wie Abfragen (= VIEW) dauerhaft gespeichert werden können, um immer wieder darauf zurückzugreifen.

5. Erstellen Sie einen SQL-Befehl, um die Abfrage „Kundenauftrag\_offen“ dauerhaft zu speichern und notieren sie diesen auf Ihrem Lösungsblatt.

Daten abfragen – TESTPROTOKOLL II (INNER JOIN)

Formulieren Sie für die oben bzw. unten abgebildeten Aufgabenstellungen ein entsprechendes SQL-Statement. Verifizieren Sie Ihre Ergebnisse mit einer manuellen Sichtung des Datenbestands der DB.

Lösung zu Abfrage 1:

*Liste aller Kunden mit Vornamen, Nachnamen und E-Mail-Adresse,  
bei denen noch unbezahlte Rechnungen vorliegen.*

|  |  |
| --- | --- |
| Statement | Testergebnis |
|  |  Fehlerfrei   Fehler:  \_\_\_\_\_\_\_\_\_\_\_\_ |

Lösung zu Abfrage 2:

*Liste aller Artikel mit Bezeichnung und Firmenname der Lieferer „Rein & Mir KG“, SUN-Senf GmbH“ und „Groebert GmbH“, sortiert nach dem Firmennamen.*

|  |  |
| --- | --- |
| Statement | Testergebnis |
|  |  Fehlerfrei   Fehler:  \_\_\_\_\_\_\_\_\_\_\_\_ |

Formulieren Sie für die Abfrage 1 (‚unbezahlte Rechnungen‘) eine Standardabfrage (= VIEW) mit dem Namen „*Kundenauftrag\_offen*“. Dies soll dazu dienen, dass bei Abfragen, welche mehrmals wiederholt werden, nicht immer dasselbe SQL-Statement geschrieben werden muss.

Geben Sie außerdem an, mit welchem Befehl der ‚VIEW‘ aufgerufen werden kann.

Lösung zu einer Standardabfrage (‚VIEW‘ für Abfrage 1):

|  |
| --- |
|  |

Aufruf einer Standardabfrage (= VIEW ‚Kundenauftrag\_offen‘)

|  |
| --- |
|  |

SQL – INNER JOIN (Abfrage über mehrere Tabellen) - INFO

# Syntax des INNER JOIN

Mit Hilfe eines INNER JOIN lassen sich Abfragen generieren, die sich über zwei oder mehrere Tabellen erstrecken. Eine Alternative zum INNER JOIN wäre der EQUI JOIN, der aber in diesem Rahmen keine weitere Rolle spielen soll.

Der INNER JOIN lässt sich in zwei Varianten anwenden. Sie unterscheiden sich in der Zusammensetzung des „FROM-Teils“ des jeweiligen SQL-Statements. Beide Varianten führen zum gleichen Ergebnis. Entscheiden Sie deshalb selbst, mit welcher Variante Sie am besten zurechtkommen.

**Syntax INNER JOIN – Variante 1** *(bei Aufruf von* ***3*** *Tabellen)*

|  |
| --- |
| SELECT tabelle\_1.attribut, tabelle\_2.attribut, ...,  tabelle\_n.attribut  FROM (tabelle\_1 **INNER JOIN** tabelle\_2  **ON** tabelle\_1.Schlüsselattribut = tabelle\_2.Schlüsselattribut)  **INNER JOIN** tabelle\_n  **ON** tabelle\_2.Schlüsselattribut = tabelle\_n.Schlüsselattribut  [WHERE Bedingung]; |

**Syntax INNER JOIN – Variante 2** *(bei Aufruf von* ***3*** *Tabellen)*

|  |
| --- |
| SELECT tabelle\_1.attribut, tabelle\_2.attribut, ...,  tabelle\_n.attribut  FROM tabelle\_1 **INNER JOIN** (tabelle\_2, tabelle\_n)  **ON** (tabelle\_1.Schlüsselattribut = tabelle\_2.Schlüsselattribut  **AND** tabelle\_2.Schlüsselattribut = tabelle\_n.Schlüsselattribut)  [WHERE Bedingung]; |

# Attribute eindeutig kennzeichnen

Bei der Verwendung eines JOIN sind mindestens zwei Tabellen betroffen. Dabei kann es – insbesondere bei größeren Datenbanken – vorkommen, dass eine Bezeichnung eines Attributs in identischer Schreibweise in mehreren Tabellen auftritt. Das trifft vor allem auf die Schlüsselattribute zu, bei denen der Primär- wie auch der Fremdschlüssel häufig gleich benannt sind.

Liegt eine solche Konstellation vor, muss beim Ausführen des SQL-Statements darauf geachtet werden, dass das Attribut aus der richtigen Tabelle angesprochen wird. Ansonsten kann es zu falschen Ergebnissen führen bzw. das SQL-Statement kann nicht ausgeführt werden, weil die Zuordnung nicht eindeutig ist. Um das zu vermeiden, wird das **Attribut zusammen mit dem Tabellennamen angegeben** und dadurch eindeutig gekennzeichnet.

Tritt ein Attributname in der gesamten Datenbank sowieso nur einmal auf, so kann auf die Nennung der Tabelle verzichtet werden. **TIPP**: da in der Praxis – aufgrund der Größe einer Datenbank – der Gesamtüberblick über alle Tabellen häufig fehlt, ist es empfehlenswert, den Tabellennamen grundsätzlich mit anzugeben.

**Attribut eindeutig kennzeichnen (lange Schreibweise)**

|  |
| --- |
| SELECT mitarbeiter.Name, abteilung.bezeichnung  FROM (mitarbeiter **INNER JOIN** abteilung  **ON** mitarbeiter.M\_Nr = abteilung.M\_Nr); |

Bei sehr langen SQL-Statements ist es äußerst mühselig, den Tabellennamen immer in voller Länge auszuschreiben. Durch einen Alias lässt sich das deutlich verkürzen:

|  |
| --- |
| SELECT M.Name, A.bezeichnung  FROM (mitarbeiter [AS] M **INNER JOIN** abteilung [AS] A  **ON** M.M\_Nr = A.M\_Nr); |

Das ‚AS‘ bei der Benennung des Aliasnamens ist optional und kann weggelassen werden.

Zur Verdeutlichung der oben dargestellten allgemeinen Syntax des INNER JOIN, formulieren Sie bitte für folgende Datenbank die nachstehenden Abfragen.

**Beispieldatenbank**

1

1

n

n

**Mitarbeiter**

**P\_Nr**

V\_Name

N\_Name

Strasse

PLZ

Ort

**Abt\_Nr**

**GS\_Nr**

**Abteilungen**

**Abt\_Nr**

Bezeichnung

**Gehaltsstufen**

**GS\_Nr**

Bezeichnung

Bruttoverdienst

**Abfrage – Beispiel 1** *(Abfrage über 2 Tabellen)*

|  |
| --- |
| *Erstellen Sie eine Liste aller Mitarbeiter, die weniger als 2.000,00 € brutto verdienen. Geben Sie den Vor- und Nachnamen, die Gehaltsstufennummer, die Bezeichnung der Gehaltsstufen und den Bruttoverdienst an.* |

**Lösung INNER JOIN – Variante 1**

|  |
| --- |
|  |

**Abfrage – Beispiel 1** *(Abfrage über 2 Tabellen)*

|  |
| --- |
| *Erstellen Sie eine Liste aller Mitarbeiter, die weniger als 2.000,00 € brutto verdienen. Geben Sie den Vor- und Nachnamen, die Gehaltsstufennummer, die Bezeichnung der Gehaltsstufen und den Bruttoverdienst an.* |

**Lösung INNER JOIN – Variante 2**

|  |
| --- |
|  |

**Abfrage – Beispiel 2** *(Abfrage über mehr als 2 Tabellen)*

|  |
| --- |
| *Erstellen Sie eine Liste aller Mitarbeiter, die weniger als 2.000,00 € brutto verdienen. Geben Sie den Vor- und Nachnamen, den Bruttoverdienst, die Abteilungsnummer und die Abteilungsbezeichnung an.* |

**Lösung INNER JOIN – Variante 1**

|  |
| --- |
|  |

**Lösung INNER JOIN – Variante 2**

|  |
| --- |
|  |