LS4-7: SQL-DQL (OUTER 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 dritte 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

Liste, welche die Kunden der Höllental-Senfmühle anzeigt, die bereits eine Bestellung aufgegeben haben. Zudem sollen die jeweiligen Bestellungen der Kunden angezeigt und nach dem Nachnamen sortiert werden. (Ausgabe der Spalten kunden\_id, nachname, bestellung\_id, status)

1. Liste, welche alle Kunden der Höllental-Senfmühle anzeigt. Zudem sollen, soweit vorhanden, die jeweiligen Bestellungen der Kunden angezeigt und nach dem Nachnamen sortiert werden. (Ausgabe der Spalten kunden\_id, nachname, bestellung\_id, status).

# Handlungsaufträge

1. Informieren Sie sich über die SQL-Befehle der ‚DQL‘ in Bezug auf OUTER JOIN und bearbeiten Sie die dort hinterlegten Verständnisaufgaben.

2. Bearbeiten Sie die Datenbank der Höllental-Senfmühle und führen Sie die SQL-Befehle des Arbeitsauftrags aus.

Daten abfragen – TESTPROTOKOLL III (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, welche die Kunden der Höllental-Senfmühle anzeigt, die bereits eine Bestellung aufgegeben haben. Zudem sollen die jeweiligen Bestellungen der Kunden angezeigt und nach dem Nachnamen sortiert werden. (Ausgabe der Spalten kunden\_id, nachname, bestellung\_id, status)*

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

Lösung zu Abfrage 2:

*Liste, welche alle Kunden der Höllental-Senfmühle anzeigt. Zudem sollen, soweit vorhanden, die jeweiligen Bestellungen der Kunden angezeigt und nach dem Nachnamen sortiert werden. (Ausgabe der Spalten kunden\_id, nachname, bestellung\_id, status).*

|  |  |
| --- | --- |
| Statement | Testergebnis |
|  |  Fehlerfrei Fehler:\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Welcher grundlegende Unterschied besteht zwischen der Abfrage 1 und Abfrage 2?** |

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

# Syntax des OUTER JOIN

Mit Hilfe eines OUTER JOIN lassen sich Abfragen generieren, welche alle Zeilen einer Tabelle auflistet, unabhängig davon, ob es in der verknüpften Tabelle übereinstimmende Schlüsselattribute gibt.

**OUTER JOIN**

Der OUTER JOIN zweier Tabellen ist der INNER JOIN dieser beiden Tabellen, der um folgende Zeilen erweitert wird: Zeilen der rechten (RIGHT OUTER JOIN) oder linken (LEFT OUTER JOIN) Tabelle, für welche keine passenden Primär-Fremdschlüssel-Paarungen gefunden wurden.

Nehmen Sie beispielsweise eine Liste von Vertretern und die von den Vertretern abgeschlossenen Verträge. Wollte man nun die Anzahl der abgeschlossenen Verträge pro Vertreter wissen, würde ein INNER JOIN alle Vertreter unterdrücken, die noch keinen Vertrag abgeschlossen haben.

In der Anwendung verhält sich der OUTER JOIN genau wie der INNER JOIN. Es wird lediglich das Wort INNER durch LEFT oder RIGHT (je nach Situation) ersetzt.

Zur Verdeutlichung formulieren Sie bitte für folgende Datenbank die nachstehenden Abfragen.

**Beispieldatenbank**

1

n

**Vertreter**

**Vertreter\_Nr (PK)**

Name

Vorname

Geburtsdatum

**Verträge**

**Vertrags\_Nr (PK)**

Bezeichnung

Abschlussdatum

**Vertreter\_ID (FK)**

|  |  |  |  |
| --- | --- | --- | --- |
| Vertreter\_Nr | Name | Vorname | Geburtsdatum |
| 1 | Beck | Moritz | 28.01.1995 |
| 2 | Aston | Martin |  |
| 3 | Vedes | Julia | 16.04.1999 |

|  |  |  |  |
| --- | --- | --- | --- |
| Vertrags\_Nr | Bezeichnung | Abschlussdatum | Vertreter\_ID |
| 1001 | Hausratversich. | 05.07.2022 | 2 |
| 1002 | Kfz-Versich. | 07.07.2022 | 3 |
| 1003 | Unfallversich. | 08.07.2022 |  |
| 1004 | Haftpflichtvers. | 10.07.2022 | 2 |

**Abfrage – Beispiel 1 (INNER JOIN)**

|  |
| --- |
| *Erstellen Sie eine Liste, in der ersichtlich wird, welcher Vertreter, welche Versicherungen verkauft haben. (Spalten: Vertretername, Vertrags\_Nr, Bezeichnung, Vertreter\_ID)* |
| **Statement:** |  |
| **Ergebnistabelle:** |
|  |  |  |  |  |  |
|  | Name | Vertrags\_Nr | Bezeichnung | Vertreter\_ID |  |
|  | Aston | 1001 | Hausratversich. | 2 |  |
|  | Aston | 1004 | Haftpflichtvers. | 2 |  |
|  | Vedes | 1002 | Kfz-Versich. | 3 |  |
|  |  |  |  |  |  |
| **Ergebnisinterpretation:** |
| 3 Datensätze: Vertreter Aston hat 2 Verträge verkauft, Vedes nur einen. Vertreter Beck wird nicht angezeigt, weil dieser keinen Vertrag verkauft hat. |

**Abfrage – Beispiel 2 (LEFT JOIN)**

|  |
| --- |
| *Erstellen Sie eine Liste, aller Vertreter. Zudem sollen – soweit vorhanden – die Verträge angezeigt werden. (Spalten: Vertretername, Vertrags\_Nr, Bezeichnung, Vertreter\_ID)* |
| **Statement:** |  |
| **Ergebnistabelle:** |
|  |  |  |  |  |  |
|  | Name | Vertrags\_Nr | Bezeichnung | Vertreter\_ID |  |
|  | Beck | NULL | NULL | NULL |  |
|  | Aston | 1001 | Hausratversich. | 2 |  |
|  | Aston | 1004 | Haftpflichversch.. | 2 |  |
|  | Vedes | 1002 | Kfz-Versich. | 3 |  |
|  |  |  |  |  |  |

**Abfrage – Beispiel 2 (LEFT JOIN) - Fortsetzung**

|  |
| --- |
| **Ergebnisinterpretation:** |
| * 4 Datensätze: Zusätzlich zu Beispiel 1 wird nun auch Vertreter Beck angezeigt. Diesem werden NULL-Werte zugeordnet, da er noch keine Verträge abgeschlossen hat.
* Der LEFT JOIN führt dazu, dass alle Datensätze der linken Tabelle (in diesem Fall die Tabelle Vertreter, weil diese links vom Schlüsselwort LEFT JOIN steht) und nur die dazugehörigen Datensätze der rechten Tabelle angezeigt werden.
 |

**Abfrage – Beispiel 3 (RIGHT JOIN)**

|  |
| --- |
| *Erstellen Sie eine Liste mit allen Verträgen. Zudem sollen die Vertreter angezeigt werden, die einen Vertrag verkauft haben. (Spalten: Vertretername, Vertrags\_Nr, Bezeichnung, Vertreter\_ID)* |
| **Statement:** |  |
| **Ergebnistabelle:** |
|  |  |  |  |  |  |
|  | Name | Vertrags\_Nr | Bezeichnung | Vertreter\_ID |  |
|  | Aston | 1001 | Hausratversich. | 2 |  |
|  | Vedes | 1002 | Kfz-Versich. | 3 |  |
|  | NULL | 1003 | Unfallversich. | NULL |  |
|  | Aston | 1004 | Haftpflichtvers. | 2 |  |
|  |  |  |  |  |  |
| **Ergebnisinterpretation:** |
| * 4 Datensätze: Die Versicherung hat insgesamt 4 Verträge verkauft. Nur bei der Unfallversicherung wurde nicht dokumentiert, welcher Vertreter diese verkauft hat.
* Der RIGHT JOIN führt dazu, dass alle Datensätze der rechten Tabelle (in diesem Fall die Tabelle Verträge, weil diese rechts vom Schlüsselwort RIGHT JOIN steht) und nur die dazugehörigen Datensätze der linken Tabelle angezeigt werden.
 |

Die Unterscheidung zwischen **LEFT** und **RIGHT** JOIN hat damit nur Auswirkung darauf, von welcher Tabelle alle Datensätze angezeigt werden sollen. Ein LEFT JOIN lässt sich deshalb problemlos in einen RIGHT JOIN umwandeln (und umgekehrt), indem die Reihenfolge der Tabelle getauscht wird.

FROM Tabelle A LEFT JOIN Tabelle B **=** FROM Tabelle B RIGHT-JOIN Tabelle A

**Abfrage – Beispiel 4 (wie Beispiel 3, nur als LEFT JOIN)**

|  |
| --- |
| *Erstellen Sie eine Liste mit allen Verträgen. Zudem sollen die Vertreter angezeigt werden, die einen Vertrag verkauft haben. (Spalten: Vertretername, Vertrags\_Nr, Bezeichnung, Vertreter\_ID)* |
| **Statement:** |  |
| **Ergebnistabelle:** |
|  |  |  |  |  |  |
|  | Name | Vertrags\_Nr | Bezeichnung | Vertreter\_ID |  |
|  | Aston | 1001 | Hausratversich. | 2 |  |
|  | Vedes | 1002 | Kfz-Versich. | 3 |  |
|  | NULL | 1003 | Unfallversich. | NULL |  |
|  | Aston | 1004 | Haftpflichtvers. | 2 |  |

Grundsätzlich lassen sich LEFT und RIGHT JOIN alternativ verwenden. Die Literatur empfiehlt aber, auf die Verwendung von RIGHT JOIN zu verzichten. Dieser JOIN wird nicht von allen Datenbankmanagementsystemen (DBMS) unterstützt und lässt sich durch einfaches Umsortieren durch einen LEFT JOIN ersetzen.