



Hardware and Software
Engineered to Work Together

Oracle Database 12c: PL/SQL Fundamentals

Übungen

D80182DE11

Production 1.1 | November 2014 | D88612

Learn more from Oracle University at oracle.com/education/

Copyright © 2014, Oracle und/oder verbundene Unternehmen. All rights reserved. Alle Rechte vorbehalten.

Diese Software und zugehörige Dokumentation werden im Rahmen eines Lizenzvertrages zur Verfügung gestellt, der Einschränkungen hinsichtlich Nutzung und Offenlegung enthält und durch Gesetze zum Schutz geistigen Eigentums geschützt ist. Sofern nicht ausdrücklich in Ihrem Lizenzvertrag vereinbart oder gesetzlich geregelt, darf diese Software weder ganz noch teilweise in irgendeiner Form oder durch irgendein Mittel zu irgendeinem Zweck kopiert, reproduziert, übersetzt, gesendet, verändert, lizenziert, übertragen, verteilt, ausgestellt, ausgeführt, veröffentlicht oder angezeigt werden. Reverse Engineering, Disassemblierung oder Dekompilierung der Software ist verboten, es sei denn, dies ist erforderlich, um die gesetzlich vorgesehene Interoperabilität mit anderer Software zu ermöglichen.

Die hier angegebenen Informationen können jederzeit und ohne vorherige Ankündigung geändert werden. Wir übernehmen keine Gewähr für deren Richtigkeit. Sollten Sie Fehler oder Unstimmigkeiten finden, bitten wir Sie, uns diese schriftlich mitzuteilen.

Wird diese Software oder zugehörige Dokumentation an die Regierung der Vereinigten Staaten von Amerika bzw. einen Lizenznehmer im Auftrag der Regierung der Vereinigten Staaten von Amerika geliefert, gilt Folgendes:

U.S. GOVERNMENT END USERS:

Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Oracle und Java sind eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen. Andere Namen und Bezeichnungen können Marken ihrer jeweiligen Inhaber sein.

Autor

Dimpi Rani Sarmah

Technischer Inhalt und Überarbeitung

Nancy Greenberg, Swarnapriya Shridhar, KimSeong Loh, Miyuki Osato, Laszlo Czinkoczki, Madhavi Siddireddy, Jim Spiller, Christopher Wensley

Dieses Buch wurde erstellt mit: Oracle Tutor

Inhaltsverzeichnis

| | |
|--|-------------|
| Übungen zu Lektion 1: Einführung..... | 1-1 |
| Übungen zu Lektion 1 | 1-2 |
| Übungen zu Lektion 1: Erste Schritte..... | 1-3 |
| Übungen zu Lektion 1 – Lösung: Erste Schritte | 1-5 |
| Übungen zu Lektion 2: PL/SQL – Einführung..... | 2-1 |
| Übungen zu Lektion 2 | 2-2 |
| Übungen zu Lektion 2: PL/SQL – Einführung | 2-3 |
| Übungen zu Lektion 2 – Lösung: PL/SQL – Einführung..... | 2-4 |
| Übungen zu Lektion 3: PL/SQL-Variablen deklarieren | 3-1 |
| Übungen zu Lektion 3: PL/SQL-Variablen deklarieren..... | 3-2 |
| Übungen zu Lektion 3 – Lösung: PL/SQL-Variablen deklarieren | 3-5 |
| Übungen zu Lektion 4: Ausführbare Anweisungen erstellen..... | 4-1 |
| Übungen zu Lektion 4: Ausführbare Anweisungen erstellen | 4-2 |
| Übungen zu Lektion 4 – Lösung: Ausführbare Anweisungen erstellen | 4-4 |
| Übungen zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken..... | 5-1 |
| Übungen zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken | 5-2 |
| Übungen zu Lektion 5 – Lösung: SQL-Anweisungen in PL/SQL-Blöcken..... | 5-4 |
| Übungen zu Lektion 6: Kontrollstrukturen erstellen..... | 6-1 |
| Übungen zu Lektion 6: Kontrollstrukturen erstellen..... | 6-2 |
| Übungen zu Lektion 6 – Lösung: Kontrollstrukturen erstellen | 6-4 |
| Übungen zu Lektion 7: Mit zusammengesetzten Datentypen arbeiten | 7-1 |
| Übungen zu Lektion 7: Mit zusammengesetzten Datentypen arbeiten | 7-2 |
| Übungen zu Lektion 7 – Lösung: Mit zusammengesetzten Datentypen arbeiten..... | 7-5 |
| Übungen zu Lektion 8: Explizite Cursor | 8-1 |
| Übung 1 zu Lektion 8: Explizite Cursor | 8-2 |
| Übung 1 zu Lektion 8 – Lösung: Explizite Cursor..... | 8-5 |
| Übung 2 zu Lektion 8: Optionale Übung zu expliziten Cursors | 8-10 |
| Übung 2 zu Lektion 8 – Lösung: Optionale Übung zu expliziten Cursors | 8-11 |
| Übungen zu Lektion 9: Exceptions behandeln..... | 9-1 |
| Übung 1 zu Lektion 9: Vordefinierte Exceptions behandeln..... | 9-2 |
| Übung 1 zu Lektion 9 – Lösung: Vordefinierte Exceptions behandeln | 9-3 |
| Übung 2 zu Lektion 9: Standardmäßige Oracle-Server-Exceptions behandeln | 9-5 |
| Übung 2 zu Lektion 9 – Lösung: Standardmäßige Oracle-Server-Exceptions behandeln..... | 9-6 |
| Übungen zu Lektion 10: Stored Procedures und Stored Functions – Einführung..... | 10-1 |
| Übungen zu Lektion 10: Stored Procedures erstellen und verwenden..... | 10-2 |
| Übungen zu Lektion 10 – Lösung: Stored Procedures erstellen und verwenden | 10-4 |
| Zusätzliche Übungen und Lösungen zu Lektion 1..... | 11-1 |
| Übungen zu Lektion 1 | 11-2 |
| Zusätzliche Übungen und Lösungen zu Lektion 2..... | 12-1 |
| Zusätzliche Übungen zu Lektion 2 | 12-2 |
| Übungen zu Lektion 2: Deklarationen beurteilen..... | 12-3 |
| Übungen zu Lektion 2 – Lösung: Deklarationen beurteilen..... | 12-4 |

| | |
|--|-------------|
| Zusätzliche Übungen und Lösungen zu Lektion 3 | 13-1 |
| Übungen zu Lektion 3: Ausdrücke beurteilen..... | 13-2 |
| Übungen zu Lektion 3 – Lösung: Ausdrücke beurteilen | 13-3 |
| Zusätzliche Übungen und Lösungen zu Lektion 4 | 14-1 |
| Übungen zu Lektion 4: Ausführbare Anweisungen beurteilen..... | 14-2 |
| Übungen zu Lektion 4 – Lösung: Ausführbare Anweisungen beurteilen | 14-3 |
| Zusätzliche Übungen und Lösungen zu Lektion 5 | 15-1 |
| Übung 1 zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken | 15-2 |
| Übung 1 zu Lektion 5 – Lösung: SQL-Anweisungen in PL/SQL-Blöcken..... | 15-3 |
| Übung 2 zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken | 15-4 |
| Übung 2 zu Lektion 5 – Lösung: SQL-Anweisungen in PL/SQL-Blöcken..... | 15-5 |
| Zusätzliche Übungen und Lösungen zu Lektion 6 | 16-1 |
| Übung 1 zu Lektion 6: Kontrollstrukturen erstellen..... | 16-2 |
| Übung 1 zu Lektion 6 – Lösung: Kontrollstrukturen erstellen | 16-3 |
| Übung 2 zu Lektion 6: Kontrollstrukturen erstellen..... | 16-4 |
| Übung 2 zu Lektion 6 – Lösung: Kontrollstrukturen erstellen | 16-5 |
| Zusätzliche Übungen und Lösungen zu Lektion 7: Mit zusammengesetzten Datentypen arbeiten | 17-1 |
| Zusätzliche Übungen zu den Lektionen "Mit zusammengesetzten Datentypen arbeiten" und "Explizite Cursor" | 17-2 |
| Übung 1 zu Lektionen 7/8: Daten mit einem expliziten Cursor abrufen..... | 17-3 |
| Übung 1 zu Lektionen 7/8 – Lösung: Daten mit einem expliziten Cursor abrufen..... | 17-4 |
| Übung 2 zu Lektionen 7/8: Assoziative Arrays und explizite Cursor | 17-5 |
| Übung 2 zu Lektionen 7/8 – Lösung: Assoziative Arrays und explizite Cursor..... | 17-6 |
| Zusätzliche Übungen und Lösungen zu Lektion 8: Explizite Cursor | 18-1 |
| Übungen zu Lektion 8 | 18-2 |
| Zusätzliche Übungen und Lösungen zu Lektion 9: Exceptions behandeln | 19-1 |
| Übungen zu Lektion 9: Exceptions behandeln | 19-2 |
| Übungen zu Lektion 9 – Lösung: Exceptions behandeln | 19-3 |

Übungen zu Lektion 1: Einführung

Kapitel 1

Übungen zu Lektion 1

Lektionsüberblick

In diesen Übungen führen Sie die folgenden Aufgaben aus:

- SQL Developer starten
- Neue Datenbankverbindung erstellen
- Schematabellen durchsuchen
- SQL Developer-Voreinstellung festlegen

Hinweis: Für alle schriftlich vorliegenden Übungen wird SQL Developer als Entwicklungsumgebung verwendet. Den Kursteilnehmern wird ebenfalls die Verwendung von SQL Developer empfohlen, allerdings steht ihnen in diesem Kurs als Alternative auch SQL*Plus zur Verfügung.

Übungen zu Lektion 1: Erste Schritte

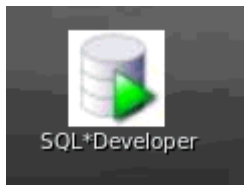
1. Starten Sie SQL Developer.
2. Erstellen Sie mit folgenden Informationen eine Datenbankverbindung. (**Hinweis:** Aktivieren Sie das Kontrollkästchen **Save Password**:
 - a. **Connection Name:** MyConnection
 - b. **Username:** ora41
 - c. **Password:** ora41
 - d. **Hostname:** localhost
 - e. **Port:** 1521
 - f. **SID:** orcl
3. Testen Sie die neue Verbindung. Wenn als Status **Success** angezeigt wird, melden Sie sich über diese neue Verbindung bei der Datenbank an.
 - a. Klicken Sie im Fenster **Database Connection** auf die Schaltfläche **Test**.
Hinweis: Der Verbindungsstatus wird in der unteren linken Ecke des Fensters angezeigt.
 - b. Wenn als Status **Success** angezeigt wird, klicken Sie auf die Schaltfläche **Connect**.
4. Durchsuchen Sie die Struktur der Tabelle `EMPLOYEES`, und zeigen Sie ihre Daten an.
 - a. Blenden Sie die Verbindung **MyConnection** ein, indem Sie daneben auf das Pluszeichen klicken.
 - b. Blenden Sie die Einträge für das Symbol **Tables** ein, indem Sie daneben auf das Pluszeichen klicken.
 - c. Zeigen Sie die Struktur der Tabelle `EMPLOYEES` an.
5. Zeigen Sie in der Registerkarte **Data** die Daten in der Tabelle `EMPLOYEES` an.
6. Wählen Sie mit dem SQL Worksheet die Nachnamen und Gehälter aller Mitarbeiter, deren Jahresgehalt über 10.000 US-Dollar liegt. Klicken Sie zur Ausführung der `SELECT`-Anweisung sowohl auf Symbol **Execute Statement** (oder drücken F9) als auch auf das Symbol **Run Script** (oder drücken F5). Prüfen Sie die Ergebnisse beider Ausführungsmethoden für die `SELECT`-Anweisungen in den entsprechenden Registerkarten.
Hinweis: Machen Sie sich einige Minuten mit den Daten vertraut, oder konsultieren Sie Anhang A mit den Beschreibungen und Daten aller Tabellen im Schema `HR`, die in diesem Kurs verwendet werden.
7. Wählen Sie im SQL Developer-Menü **Tools** die Option **Preferences**. Das Fenster **Preferences** wird angezeigt.
8. Wählen Sie **Database > Worksheet Parameters**. Klicken Sie im Textfeld **Select default path to look for scripts** auf die Schaltfläche **Browse**, um zum Verzeichnis `/home/oracle/labs/plsf` zu navigieren. Dieses Verzeichnis enthält die Codebeispielskripte, Übungsskripte und Lösungsskripte zu den Übungen, die in diesem Kurs verwendet werden. Um die Einstellung für **Worksheet Parameter** zu speichern, klicken Sie im Fenster **Preferences** auf **OK**.

9. Machen Sie sich mit der Struktur des Verzeichnisses `/home/oracle/labs/plsf` vertraut.
- a. Wählen Sie im Menü **File** die Option **Open**. Im Fenster **Open** wird automatisch das Verzeichnis `.../plsf` als Ausgangsspeicherort gewählt. Dieses Verzeichnis enthält drei Unterverzeichnisse:
- Das Verzeichnis `/code_ex` enthält die Codebeispiele aus dem Kursmaterial. Jedes `SQL`-Skript gehört zu einer bestimmten Seite in der Lektion.
 - Das Verzeichnis `/labs` enthält den Code, der in bestimmten Lektionsübungen verwendet wird. Sie werden angewiesen, das bei einer bestimmten Übung erforderliche Skript auszuführen.
 - Das Verzeichnis `/soln` enthält die Lösungen zu den einzelnen Übungen. Jedes `SQL`-Skript ist mit der zugehörigen Referenz **Übung_Übungsaufgabe** nummeriert.
- b. Sie können auch in der Registerkarte **Files** durch die Verzeichnisse navigieren, um die Skriptdateien zu öffnen.
- c. Navigieren Sie in der Registerkarte **Files** mit dem Fenster **Open** durch die Verzeichnisse, und öffnen Sie eine Skriptdatei, ohne den Code auszuführen.
- d. Schließen Sie das `SQL Worksheet`.

Übungen zu Lektion 1 – Lösung: Erste Schritte

1. Starten Sie SQL Developer.

Klicken Sie auf dem Desktop auf das Symbol **SQL Developer**.

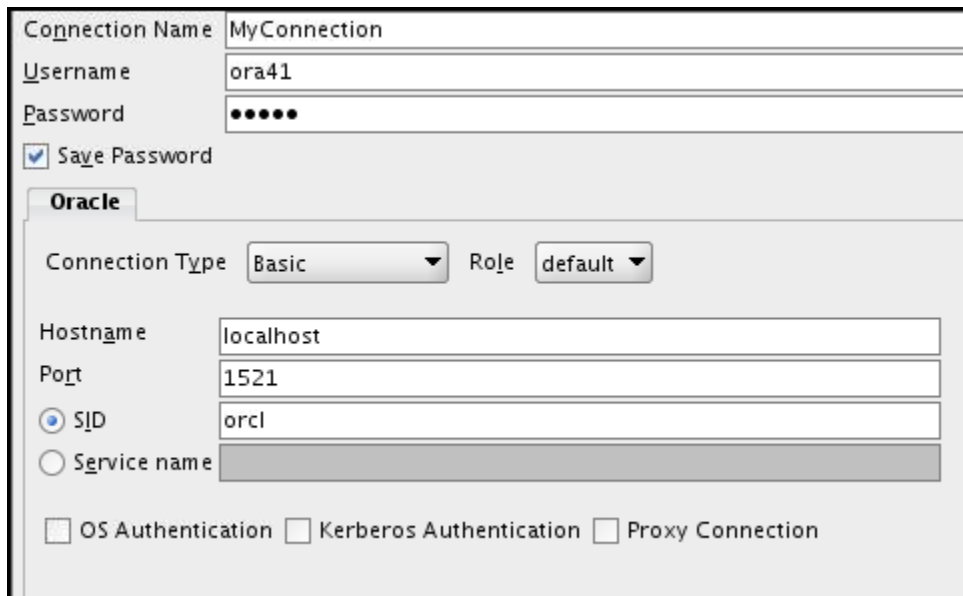


2. Erstellen Sie mit folgenden Informationen eine Datenbankverbindung. (**Hinweis:** Aktivieren Sie das Kontrollkästchen **Save Password**.):
 - a. **Connection Name:** MyConnection
 - b. **Username:** ora41
 - c. **Password:** ora41
 - d. **Hostname:** localhost
 - e. **Port:** 1521
 - f. **SID:** orcl

Klicken Sie in der Registerkarte **Connections** mit der rechten Maustaste auf den Knoten **Connections**, und wählen Sie **New Connection...**

Ergebnis: Das Fenster **New/Select Database Connection** wird angezeigt.

Erstellen Sie mithilfe der oben aufgeführten Informationen die neue Datenbankverbindung. Aktivieren Sie außerdem das Kontrollkästchen **Save Password**. Beispiel:

The image shows the 'New/Select Database Connection' dialog box in SQL Developer. The 'Connection Name' is 'MyConnection'. The 'Username' is 'ora41' and the 'Password' is masked with dots. The 'Save Password' checkbox is checked. The 'Oracle' tab is selected, showing 'Connection Type' as 'Basic' and 'Role' as 'default'. The 'Hostname' is 'localhost', 'Port' is '1521', and 'SID' is 'orcl'. The 'Service name' radio button is unselected. At the bottom, there are checkboxes for 'OS Authentication', 'Kerberos Authentication', and 'Proxy Connection', all of which are unselected.

3. Testen Sie die neue Verbindung. Wenn als Status **Success** angezeigt wird, melden Sie sich über diese neue Verbindung bei der Datenbank an.
- a. Klicken Sie im Fenster **Database Connection** auf die Schaltfläche **Test**.
Hinweis: Der Verbindungsstatus wird in der unteren linken Ecke des Fensters angezeigt.

The screenshot shows the 'New / Select Database Connection' dialog box. The 'Connection Name' is 'MyConnection', 'Username' is 'ora41', and 'Password' is masked with dots. The 'Save Password' checkbox is checked. Under the 'Oracle' tab, 'Connection Type' is 'Basic', 'Role' is 'default', 'Hostname' is 'localhost', 'Port' is '1521', and 'SID' is 'orcl'. The 'Service name' option is unselected. At the bottom, there are buttons for 'Help', 'Save', 'Clear', 'Test' (highlighted in yellow), 'Connect', and 'Cancel'. The status 'Status : Success' is displayed in the bottom left corner, highlighted with a red rectangle.

- b. Wenn als Status **Success** angezeigt wird, klicken Sie auf die Schaltfläche **Connect**.

This screenshot is identical to the previous one, showing the same dialog box configuration. However, the 'Connect' button at the bottom right is now highlighted with a red rectangle, indicating the next step in the process. The 'Status : Success' label remains in the bottom left.

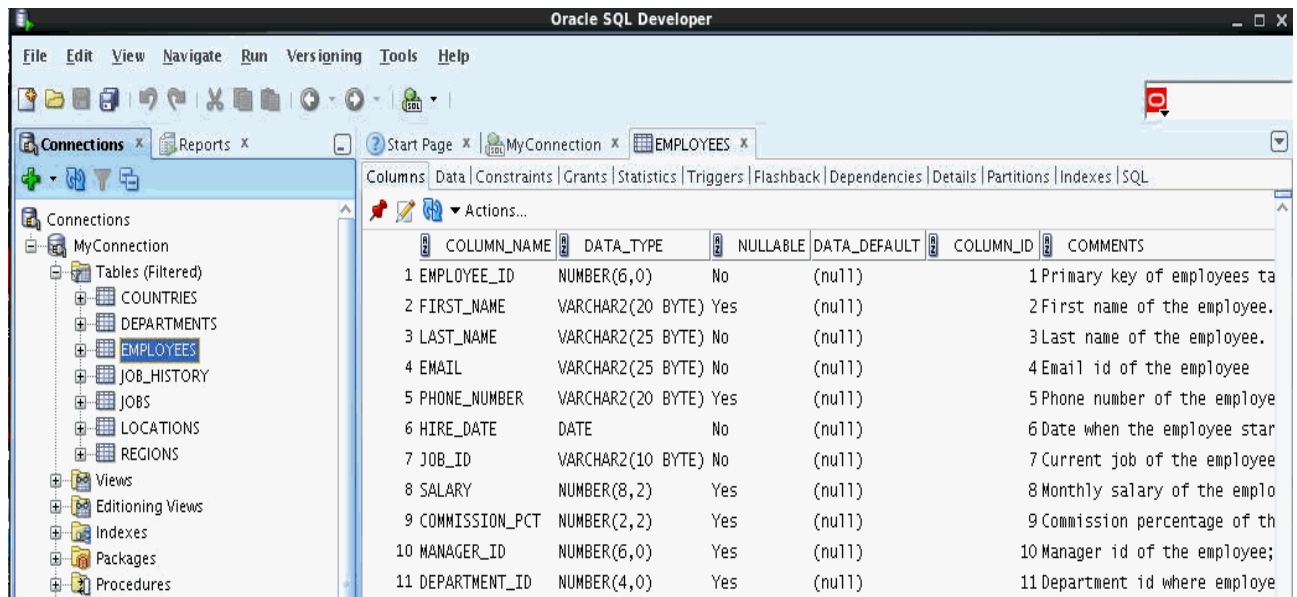
Hinweis: Um die Eigenschaften einer bestehenden Verbindung anzuzeigen, klicken Sie in der Registerkarte **Connections** mit der rechten Maustaste auf den Namen der Verbindung und wählen im Kontextmenü die Option **Properties**.

4. Durchsuchen Sie die Struktur der Tabelle `EMPLOYEES`, und zeigen Sie ihre Daten an.
 - a. Blenden Sie die Verbindung **MyConnection** ein, indem Sie daneben auf das Pluszeichen klicken.
 - b. Blenden Sie die Einträge für das Symbol **Tables** ein, indem Sie daneben auf das Pluszeichen klicken.
 - c. Zeigen Sie die Struktur der Tabelle `EMPLOYEES` an.

Führen Sie einen Drilldown in die Tabelle `EMPLOYEES` durch, indem Sie daneben auf das Pluszeichen klicken.

Klicken Sie auf die Tabelle `EMPLOYEES`.

Ergebnis: In der Registerkarte **Columns** werden die Spalten der Tabelle `EMPLOYEES` wie folgt angezeigt:

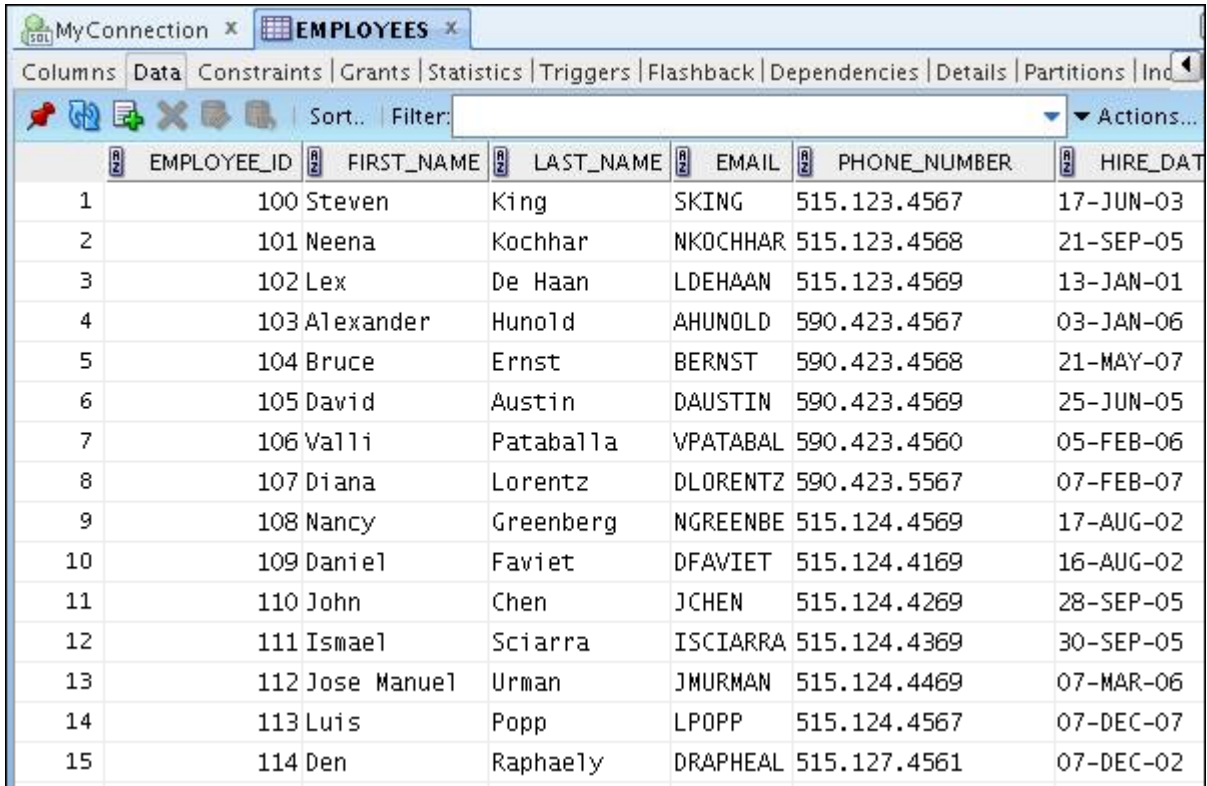


The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane shows 'MyConnection' expanded, with 'Tables (Filtered)' containing 'EMPLOYEES'. The main pane shows the 'Columns' tab for the 'EMPLOYEES' table, displaying a list of columns with their data types, nullability, and comments.

| | COLUMN_NAME | DATA_TYPE | NULLABLE | DATA_DEFAULT | COLUMN_ID | COMMENTS |
|----|----------------|-------------------|----------|--------------|-----------|--------------------------------|
| 1 | EMPLOYEE_ID | NUMBER(6,0) | No | (null) | | 1Primary key of employees ta |
| 2 | FIRST_NAME | VARCHAR2(20 BYTE) | Yes | (null) | | 2First name of the employee. |
| 3 | LAST_NAME | VARCHAR2(25 BYTE) | No | (null) | | 3Last name of the employee. |
| 4 | EMAIL | VARCHAR2(25 BYTE) | No | (null) | | 4Email id of the employee |
| 5 | PHONE_NUMBER | VARCHAR2(20 BYTE) | Yes | (null) | | 5Phone number of the employe |
| 6 | HIRE_DATE | DATE | No | (null) | | 6Date when the employee star |
| 7 | JOB_ID | VARCHAR2(10 BYTE) | No | (null) | | 7Current job of the employee |
| 8 | SALARY | NUMBER(8,2) | Yes | (null) | | 8Monthly salary of the emplo |
| 9 | COMMISSION_PCT | NUMBER(2,2) | Yes | (null) | | 9Commission percentage of th |
| 10 | MANAGER_ID | NUMBER(6,0) | Yes | (null) | | 10 Manager id of the employee; |
| 11 | DEPARTMENT_ID | NUMBER(4,0) | Yes | (null) | | 11Department id where employe |

5. Zeigen Sie in der Registerkarte **Data** die Daten in der Tabelle `EMPLOYEES` an.

Ergebnis: Die Daten der Tabelle `EMPLOYEES` werden wie folgt angezeigt:



| | EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE |
|----|-------------|-------------|-----------|----------|--------------|-----------|
| 1 | 100 | Steven | King | SKING | 515.123.4567 | 17-JUN-03 |
| 2 | 101 | Neena | Kochhar | NKOCHHAR | 515.123.4568 | 21-SEP-05 |
| 3 | 102 | Lex | De Haan | LDEHAAN | 515.123.4569 | 13-JAN-01 |
| 4 | 103 | Alexander | Hunold | AHUNOLD | 590.423.4567 | 03-JAN-06 |
| 5 | 104 | Bruce | Ernst | BERNST | 590.423.4568 | 21-MAY-07 |
| 6 | 105 | David | Austin | DAUSTIN | 590.423.4569 | 25-JUN-05 |
| 7 | 106 | Valli | Pataballa | VPATABAL | 590.423.4560 | 05-FEB-06 |
| 8 | 107 | Diana | Lorentz | DLORENTZ | 590.423.5567 | 07-FEB-07 |
| 9 | 108 | Nancy | Greenberg | NGREENBE | 515.124.4569 | 17-AUG-02 |
| 10 | 109 | Daniel | Faviet | DFAVIET | 515.124.4169 | 16-AUG-02 |
| 11 | 110 | John | Chen | JCHEN | 515.124.4269 | 28-SEP-05 |
| 12 | 111 | Ismael | Sciarra | ISCIARRA | 515.124.4369 | 30-SEP-05 |
| 13 | 112 | Jose Manuel | Urman | JMURMAN | 515.124.4469 | 07-MAR-06 |
| 14 | 113 | Luis | Popp | LPOPP | 515.124.4567 | 07-DEC-07 |
| 15 | 114 | Den | Raphaely | DRAPHEAL | 515.127.4561 | 07-DEC-02 |

6. Wählen Sie mit dem SQL Worksheet die Nachnamen und Gehälter aller Mitarbeiter, deren Jahresgehalt über 10.000 US-Dollar liegt. Klicken Sie zur Ausführung der `SELECT`-Anweisung sowohl auf Symbol **Execute Statement** (oder drücken F9) als auch auf das Symbol **Run Script** (oder drücken F5). Prüfen Sie die Ergebnisse beider Ausführungsmethoden für die `SELECT`-Anweisungen in den entsprechenden Registerkarten.

Hinweis: Machen Sie sich einige Minuten mit den Daten vertraut, oder konsultieren Sie Anhang A mit den Beschreibungen und Daten aller Tabellen im Schema `HR`, die in diesem Kurs verwendet werden.

Um das SQL Worksheet anzuzeigen, wechseln Sie zur Registerkarte **MyConnection**.

Hinweis: Sie haben diese Registerkarte bereits geöffnet, als Sie den Drilldown für die Datenbankverbindung durchgeführt haben.

Geben Sie die richtige `SELECT`-Anweisung ein. Drücken Sie F9, um die Abfrage auszuführen, und F5, um die Abfrage mit der Methode **Run Script** auszuführen.

Wenn Sie beispielsweise F9 drücken, ähneln die Ergebnisse dem folgenden Screenshot:

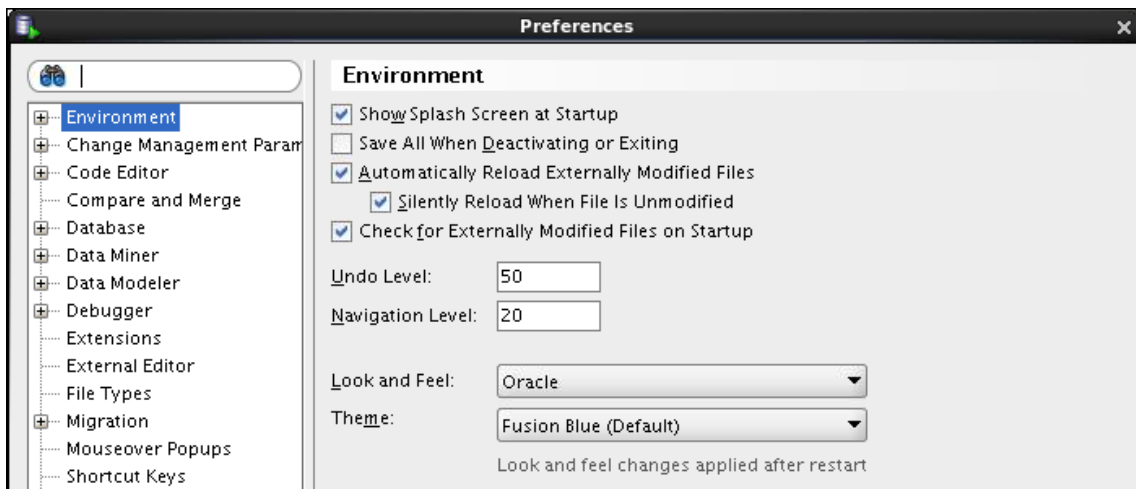
The screenshot shows the MyConnection SQL tool interface. The top toolbar includes icons for running queries, saving, and other database functions. The 'Query Builder' tab is active, displaying the following SQL query:

```
1 select last_name, salary
2 from employees
3 where salary > 10000;
```

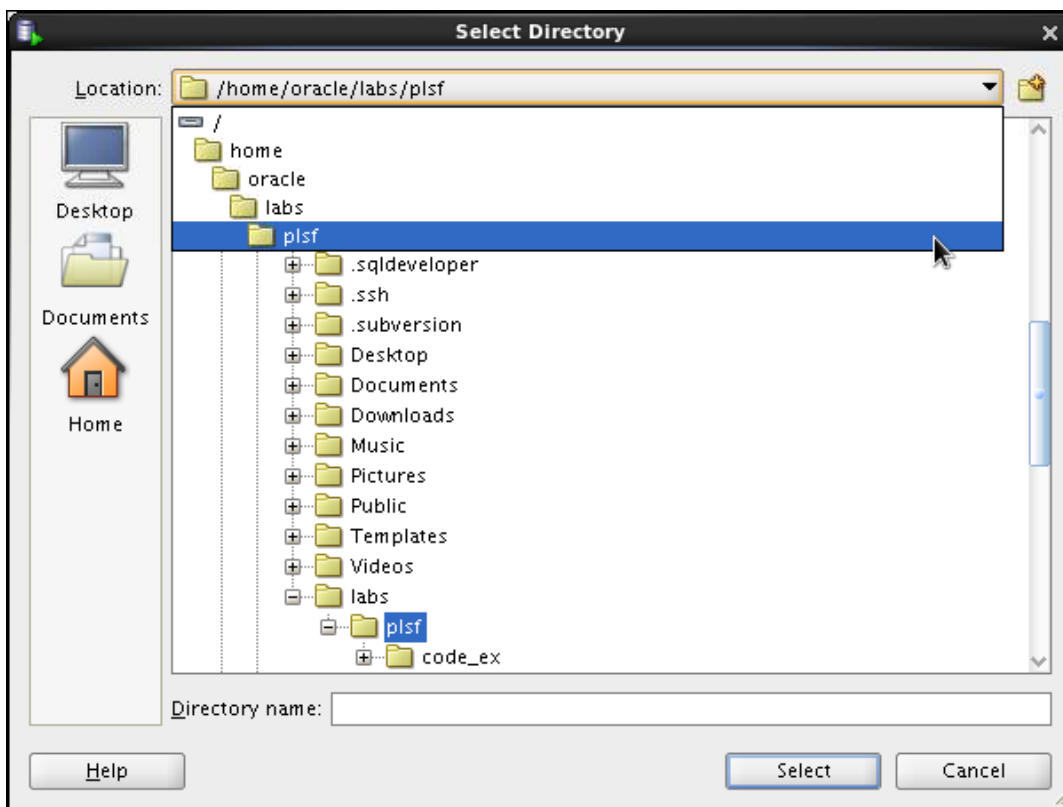
Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 15 in 0.004 seconds'. The results are displayed in a table with two columns: LAST_NAME and SALARY.

| | LAST_NAME | SALARY |
|----|-----------|--------|
| 1 | King | 24000 |
| 2 | Kochhar | 17000 |
| 3 | De Haan | 17000 |
| 4 | Greenberg | 12008 |
| 5 | Raphaely | 11000 |
| 6 | Russell | 14000 |
| 7 | Partners | 13500 |
| 8 | Errazuriz | 12000 |
| 9 | Cambrault | 11000 |
| 10 | Zlotkey | 10500 |
| 11 | Vishney | 10500 |
| 12 | Ozer | 11500 |
| 13 | Abel | 11000 |
| 14 | Hartstein | 13000 |
| 15 | Higgins | 12008 |

7. Wählen Sie im SQL Developer-Menü **Tools** die Option **Preferences**. Das Fenster **Preferences** wird angezeigt.



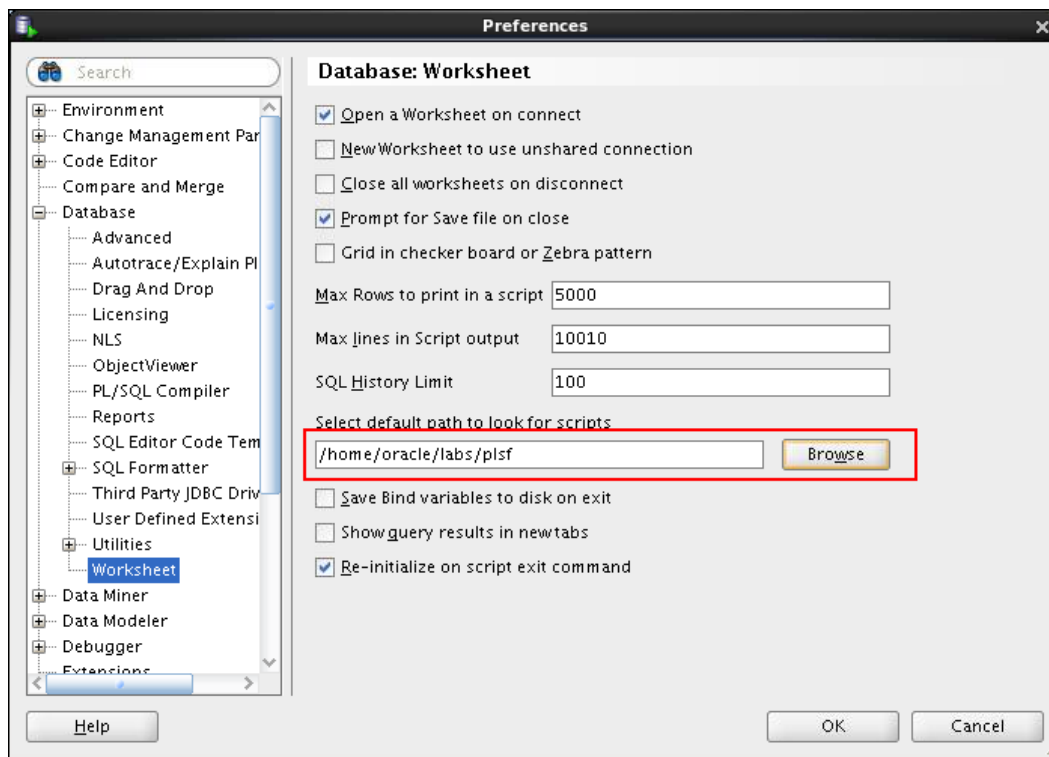
8. Wählen Sie **Database > Worksheet Parameters**. Klicken Sie im Textfeld **Select default path to look for scripts** auf die Schaltfläche **Browse**, um zum Verzeichnis `/home/oracle/labs/plsf` zu navigieren.



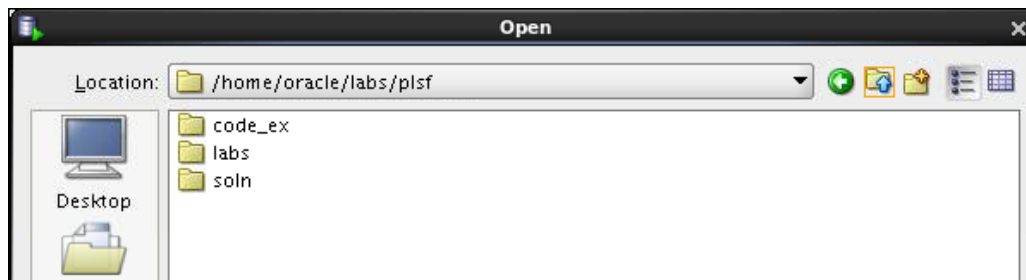
Dieses Verzeichnis enthält die Codebeispielskripte, Übungsskripte und Lösungsskripte zu den Übungen, die in diesem Kurs verwendet werden.

Um das Verzeichnis zu wählen, klicken Sie auf **Select**.

Um die Einstellung für **Worksheet Parameter** zu speichern, klicken Sie im Fenster **Preferences** auf **OK**.

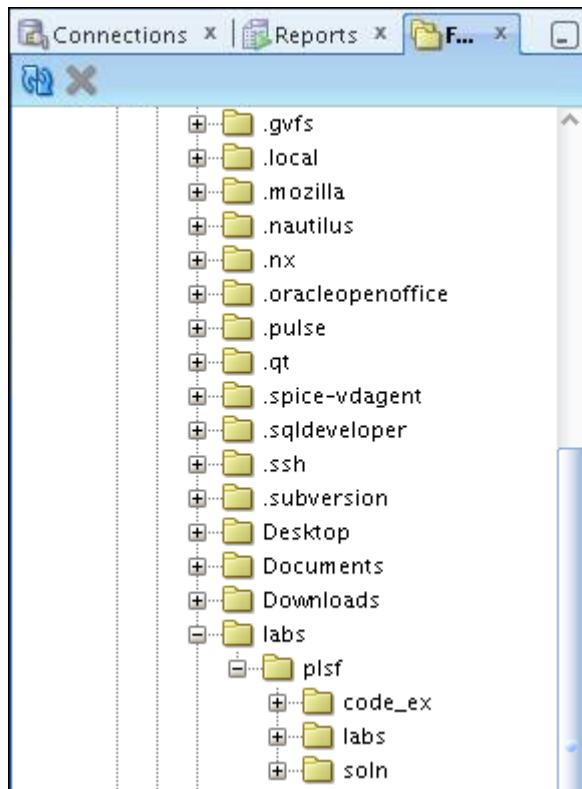


9. Machen Sie sich mit der Struktur des Verzeichnisses `/home/oracle/labs/plsf` vertraut.
- a. Wählen Sie im Menü **File** die Option **Open**. Navigieren Sie zum Verzeichnis `/home/oracle/labs/plsf`. Dieses Verzeichnis enthält drei Unterverzeichnisse:



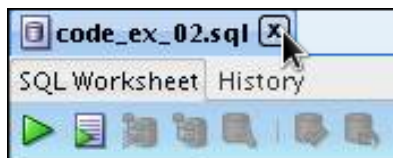
- Das Verzeichnis `/code_ex` enthält die Codebeispiele aus dem Kursmaterial. Jedes SQL-Skript gehört zu einer bestimmten Seite in der Lektion.
- Das Verzeichnis `/labs` enthält den Code, der in bestimmten Lektionsübungen verwendet wird. Sie werden angewiesen, das bei einer bestimmten Übung erforderliche Skript auszuführen.
- Das Verzeichnis `/soln` enthält die Lösungen zu den einzelnen Übungen. Jedes SQL-Skript ist mit der zugehörigen Referenz **Übung_Übungsaufgabe** nummeriert.

- b. Sie können auch in der Registerkarte **Files** durch die Verzeichnisse navigieren, um die Skriptdateien zu öffnen.



- c. Navigieren Sie in der Registerkarte **Files** mit dem Fenster **Open** durch die Verzeichnisse, und öffnen Sie eine Skriptdatei, ohne den Code auszuführen.
- d. Schließen Sie das SQL Worksheet.

Um eine SQL Worksheet-Registerkarte zu schließen, klicken Sie in der Registerkarte auf das "X" (siehe Screenshot):



Übungen zu Lektion 2: PL/SQL – Einführung

Kapitel 2

Übungen zu Lektion 2

Lektionsüberblick

Der Ordner `/home/oracle/labs/plsf/labs` ist das Arbeitsverzeichnis, in dem Sie die erstellten Skripte speichern.

Die Lösungen für alle Übungen befinden sich im Ordner `/home/oracle/labs/plsf/soln`.

Übungen zu Lektion 2: PL/SQL – Einführung

1. Welche(r) der folgenden PL/SQL-Blöcke wird/werden erfolgreich ausgeführt?
 - a.

```
BEGIN  
END;
```
 - b.

```
DECLARE  
v_amount INTEGER(10);  
END;
```
 - c.

```
DECLARE  
BEGIN  
END;
```
 - d.

```
DECLARE  
v_amount INTEGER(10);  
BEGIN  
DBMS_OUTPUT.PUT_LINE(v_amount);  
END;
```
2. Erstellen Sie einen einfachen anonymen Block, der "Hello World" ausgibt, und führen Sie ihn aus. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_02_02_soln.sql`.

Übungen zu Lektion 2 – Lösung: PL/SQL – Einführung

1. Welche(r) der folgenden PL/SQL-Blöcke wird/werden erfolgreich ausgeführt?

- a. BEGIN
END;
- b. DECLARE
v_amount INTEGER(10);
END;
- c. DECLARE
BEGIN
END;
- d. DECLARE
v_amount INTEGER(10);
BEGIN
DBMS_OUTPUT.PUT_LINE(v_amount);
END;

Der Block in a wird nicht ausgeführt, da er keine ausführbaren Anweisungen besitzt.
Im Block in b fehlt der obligatorische ausführbare Bereich, der mit dem Schlüsselwort `BEGIN` beginnt.

Der Block in c verfügt über alle erforderlichen Teile, enthält aber keine ausführbaren Anweisungen.

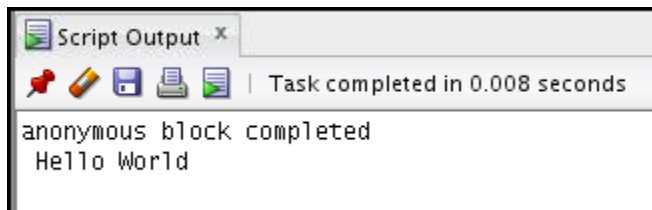
Der Block in d wird erfolgreich ausgeführt.

2. Erstellen Sie einen einfachen anonymen Block, der "Hello World" ausgibt, und führen Sie ihn aus. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_02_02_soln.sql`.

Geben Sie im Workspace folgenden Code ein, und drücken Sie F5.

```
SET SERVEROUTPUT ON
BEGIN
DBMS_OUTPUT.PUT_LINE(' Hello World ');
END;
```

In der Registerkarte **Script Output** sollte die folgende Ausgabe angezeigt werden:



Klicken Sie auf die Schaltfläche **Save**. Wählen Sie den Ordner, in dem die Datei gespeichert werden soll. Geben Sie den Dateinamen `lab_02_02_soln.sql` ein, und klicken Sie auf **Save**.

Übungen zu Lektion 3: PL/SQL-Variablen deklarieren

Kapitel 3

Übungen zu Lektion 3: PL/SQL-Variablen deklarieren

In dieser Übung deklarieren Sie PL/SQL-Variablen.

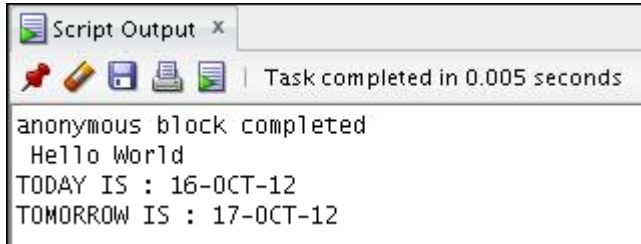
1. Identifizieren Sie die gültigen und ungültigen IDs:
 - a. today
 - b. last_name
 - c. today's_date
 - d. Number_of_days_in_February_this_year
 - e. Isleap\$year
 - f. #number
 - g. NUMBER#
 - h. number1to7
2. Identifizieren Sie die gültigen und ungültigen Variablendeklarationen und -initialisierungen:
 - a. number_of_copies PLS_INTEGER;
 - b. PRINTER_NAME constant VARCHAR2(10);
 - c. deliver_to VARCHAR2(10):=Johnson;
 - d. by_when DATE:= CURRENT_DATE+1;
3. Untersuchen Sie den folgenden anonymen Block, und wählen Sie dann unter den nachfolgenden Feststellungen eine korrekte Aussage.

```
DECLARE
  v_fname VARCHAR2(20);
  v_lname VARCHAR2(15) DEFAULT 'fernandez';
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_fname || ' ' || v_lname);
END;
```

- a. Der Block wird erfolgreich ausgeführt. Er gibt "fernandez" aus.
- b. Der Block erzeugt einen Fehler, weil die Variable `fname` ohne Initialisierung verwendet wird.
- c. Der Block wird erfolgreich ausgeführt. Er gibt "null fernandez" aus.
- d. Der Block erzeugt einen Fehler, da Sie mit dem Schlüsselwort `DEFAULT` keine Variablen vom Typ `VARCHAR2` initialisieren können.
- e. Der Block erzeugt einen Fehler, da die Variable `v_fname` nicht deklariert ist.

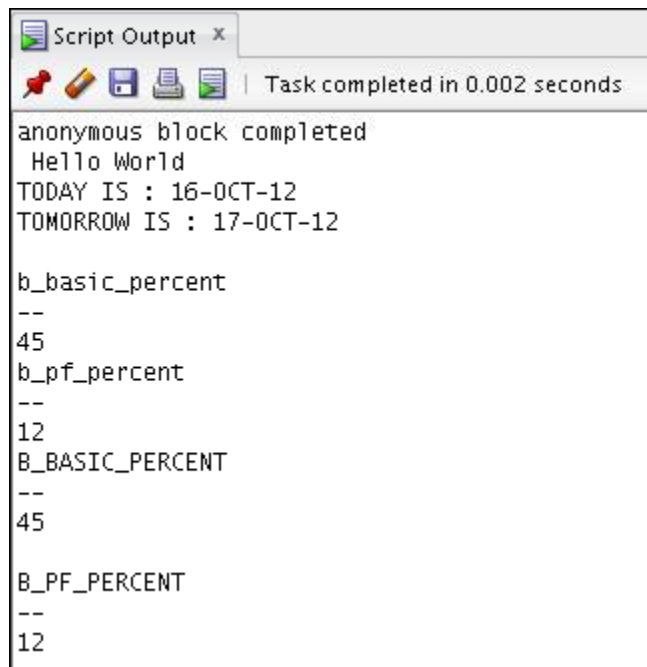
4. Ändern Sie einen bestehenden anonymen Block, und speichern Sie ihn als neues Skript.
 - a. Öffnen Sie das Skript `lab_02_02_soln.sql`, das Sie in der Übung zu Lektion 2 erstellt haben.
 - b. Deklarieren Sie in diesem PL/SQL-Block die folgenden Variablen:
 - 1) `v_today` vom Typ `DATE`. Initialisieren Sie `today` mit `SYSDATE`.
 - 2) `v_tomorrow` vom Typ `today`. Deklarieren Sie diese Variable mit dem Attribut `%TYPE`.
 - c. Im ausführbaren Bereich:
 - 1) Initialisieren Sie die Variable `v_tomorrow` mit einem Ausdruck, der das morgige Datum berechnet. (Erhöhen Sie den Wert in `today` um 1.)
 - 2) Geben Sie den Wert von `v_today` und `tomorrow` nach "Hello World" aus.
 - d. Speichern Sie das Skript unter dem Namen `lab_03_04_soln.sql`, und führen Sie es dann aus.

Ausgabebeispiel (mit unterschiedlichen Werten für `v_today` und `v_tomorrow`, da sie das Datum von heute und von morgen darstellen):



5. Bearbeiten Sie das Skript `lab_03_04_soln.sql`.
 - a. Fügen Sie Code hinzu, um die beiden Bind-Variablen `b_basic_percent` und `b_pf_percent` zu erstellen. Beide Bind-Variablen weisen den Typ `NUMBER` auf.
 - b. Weisen Sie den Bind-Variablen `b_basic_percent` und `b_pf_percent` im ausführbaren Bereich des PL/SQL-Blockes die Werte 45 und 12 zu.
 - c. Beenden Sie den PL/SQL-Block mit einem Schrägstrich (/), und zeigen Sie den Wert der Bind-Variablen mit dem Befehl `PRINT` an.

- d. Führen Sie das Skript aus, und speichern Sie es unter dem Namen lab_03_05_soln.sql. **Ausgabebeispiel:**



```
Script Output x
Task completed in 0.002 seconds

anonymous block completed
Hello World
TODAY IS : 16-OCT-12
TOMORROW IS : 17-OCT-12

b_basic_percent
--
45
b_pf_percent
--
12
B_BASIC_PERCENT
--
45

B_PF_PERCENT
--
12
```


Übungen zu Lektion 3 – Lösung: PL/SQL-Variablen deklarieren

1. Identifizieren Sie die gültigen und ungültigen IDs:

- | | |
|---|--|
| a. today | Gültig |
| b. last_name | Gültig |
| c. today's_date | Ungültig – Unzulässiges Zeichen "'" |
| d. Number_of_days_in_February_this_year | Ungültig – Zu lang |
| e. Isleap\$year | Gültig |
| f. #number | Ungültig – Darf nicht mit "#" beginnen |
| g. NUMBER# | Gültig |
| h. number1to7 | Gültig |

2. Identifizieren Sie die gültigen und ungültigen Variablendeklarationen und -initialisierungen:

- | | | |
|---------------------|-------------------------|----------|
| a. number_of_copies | PLS_INTEGER; | Gültig |
| b. PRINTER_NAME | constant VARCHAR2(10); | Ungültig |
| c. deliver_to | VARCHAR2(10) :=Johnson; | Ungültig |
| d. by_when | DATE:= CURRENT_DATE+1; | Gültig |

*Die Deklaration unter **b** ist ungültig, weil bei der Deklaration Konstantenvariablen initialisiert werden müssen.*

*Die Deklaration unter **c** ist ungültig, weil Zeichenfolgenlitterale in einfache Anführungszeichen eingeschlossen werden müssen.*

3. Untersuchen Sie den folgenden anonymen Block, und wählen Sie dann unter den nachfolgenden Feststellungen eine korrekte Aussage.

```
DECLARE
  v_fname VARCHAR2(20);
  v_lname VARCHAR2(15) DEFAULT 'fernandez';
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_fname || ' ' || v_lname);
END;
```

- a. Der Block wird erfolgreich ausgeführt. Er gibt "fernandez" aus.
- b. Der Block erzeugt einen Fehler, weil die Variable `f_name` ohne Initialisierung verwendet wird.
- c. Der Block wird erfolgreich ausgeführt. Er gibt "null fernandez" aus.
- d. Der Block erzeugt einen Fehler, da Sie mit dem Schlüsselwort `DEFAULT` keine Variablen vom Typ `VARCHAR2` initialisieren können.
- e. Der Block erzeugt einen Fehler, da die Variable `v_fname` nicht deklariert ist.

a. Der Block wird erfolgreich ausgeführt. Er gibt "fernandez" aus.

4. Ändern Sie einen bestehenden anonymen Block, und speichern Sie ihn als neues Skript.
- Öffnen Sie das Skript `lab_02_02_soln.sql`, das Sie in der Übung zu Lektion 2 erstellt haben.
 - Deklariieren Sie im PL/SQL-Block die folgenden Variablen:
 - Variable `v_today` vom Typ `DATE`. Initialisieren Sie `today` mit `SYSDATE`.

```
DECLARE
    v_today DATE:=SYSDATE;
```

- Variable `v_tomorrow` vom Typ `today`. Deklarieren Sie diese Variable mit dem Attribut `%TYPE`.

```
v_tomorrow v_today%TYPE;
```

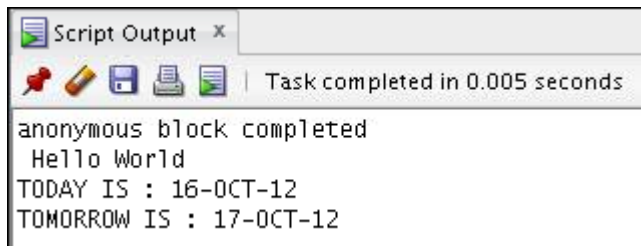
Im ausführbaren Bereich:

- Initialisieren Sie die Variable `v_tomorrow` mit einem Ausdruck, der das morgige Datum berechnet. (Erhöhen Sie den Wert in `v_today` um 1.)
- Geben Sie den Wert von `v_today` und `v_tomorrow` nach "Hello World" aus.

```
BEGIN
    v_tomorrow:=v_today +1;
    DBMS_OUTPUT.PUT_LINE(' Hello World ');
    DBMS_OUTPUT.PUT_LINE('TODAY IS : ' || v_today);
    DBMS_OUTPUT.PUT_LINE('TOMORROW IS : ' || v_tomorrow);
END;
```

- Speichern Sie das Skript unter dem Namen `lab_03_04_soln.sql`, und führen Sie es dann aus.

Ausgabebeispiel (mit unterschiedlichen Werten für `v_today` und `v_tomorrow`, da sie das Datum von heute und von morgen darstellen):



5. Bearbeiten Sie das Skript lab_03_04_soln.sql.

- a. Fügen Sie Code hinzu, um die beiden Bind-Variablen `b_basic_percent` und `b_pf_percent` zu erstellen. Beide Bind-Variablen weisen den Typ `NUMBER` auf.

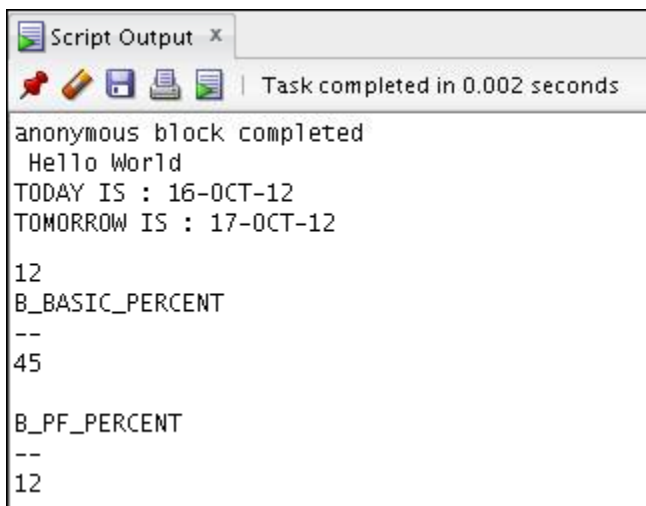
```
VARIABLE b_basic_percent NUMBER  
VARIABLE b_pf_percent NUMBER
```

- b. Weisen Sie den Bind-Variablen `b_basic_percent` und `b_pf_percent` im ausführbaren Bereich des PL/SQL-Blockes die Werte 45 bzw. 12 zu.

```
:b_basic_percent:=45;  
:b_pf_percent:=12;
```

- c. Beenden Sie den PL/SQL-Block mit einem Schrägstrich (/), und zeigen Sie den Wert der Bind-Variablen mit dem Befehl `PRINT` an.

```
/  
PRINT b_basic_percent  
PRINT b_pf_percent
```



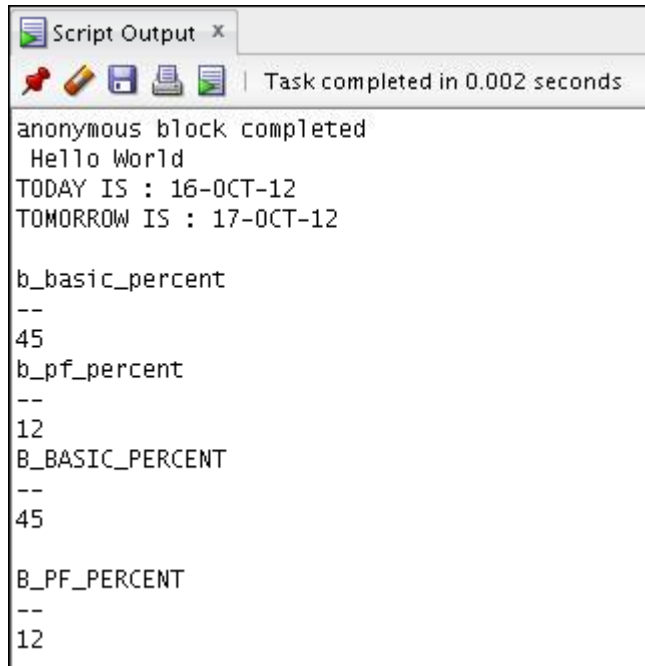
The screenshot shows a window titled "Script Output x" with a status bar indicating "Task completed in 0.002 seconds". The output text is as follows:

```
anonymous block completed  
Hello World  
TODAY IS : 16-OCT-12  
TOMORROW IS : 17-OCT-12  
  
12  
B_BASIC_PERCENT  
--  
45  
  
B_PF_PERCENT  
--  
12
```

ODER

```
PRINT
```

- d. Führen Sie das Skript aus, und speichern Sie es unter dem Namen lab_03_05_soln.sql. Ausgabebeispiel:



```
Script Output x
Task completed in 0.002 seconds

anonymous block completed
Hello World
TODAY IS : 16-OCT-12
TOMORROW IS : 17-OCT-12

b_basic_percent
--
45
b_pf_percent
--
12
B_BASIC_PERCENT
--
45
B_PF_PERCENT
--
12
```

Übungen zu Lektion 4: Ausführbare Anweisungen erstellen

Kapitel 4

Übungen zu Lektion 4: Ausführbare Anweisungen erstellen

Hinweis: Wenn Sie die Codebeispiele für diese Lektion ausgeführt haben, müssen Sie den folgenden Code ausführen, bevor Sie mit dieser Übung beginnen:

```
DROP sequence my_seq;
```

In dieser Übung prüfen und erstellen Sie ausführbare Anweisungen.

```
DECLARE
  v_weight      NUMBER(3) := 600;
  v_message     VARCHAR2(255) := 'Product 10012';
BEGIN
  DECLARE
    v_weight      NUMBER(3) := 1;
    v_message     VARCHAR2(255) := 'Product 11001';
    v_new_locn    VARCHAR2(50) := 'Europe';
  BEGIN
    v_weight := v_weight + 1;
    v_new_locn := 'Western ' || v_new_locn;
1 →
  END;
  v_weight := v_weight + 1;
  v_message := v_message || ' is in stock';
  v_new_locn := 'Western ' || v_new_locn;
2 →
END;
/
```

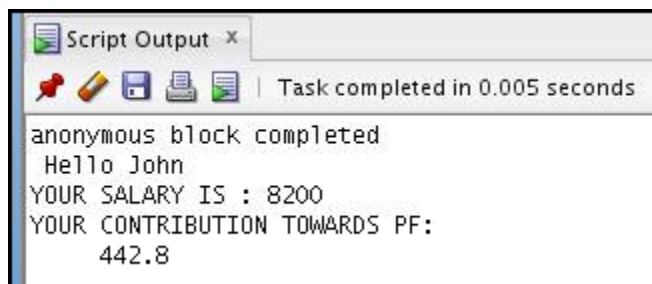
1. Werten Sie den vorstehenden PL/SQL-Block aus, und bestimmen Sie den Datentyp und Wert jeder der folgenden Variablen entsprechend den Regeln für Gültigkeitsbereiche.
 - a. Wert von `v_weight` an 1. Position:
 - b. Wert von `v_new_locn` an 1. Position:
 - c. Wert von `v_weight` an 2. Position:
 - d. Wert von `v_message` an 2. Position:
 - e. Wert von `v_new_locn` an 2. Position:

```
DECLARE
  v_customer     VARCHAR2(50) := 'Womansport';
  v_credit_rating VARCHAR2(50) := 'EXCELLENT';
BEGIN
  DECLARE
    v_customer     NUMBER(7) := 201;
    v_name         VARCHAR2(25) := 'Unisports';
  BEGIN
    v_credit_rating := 'GOOD';
    ...
  END;
  ...
END;
```

2. Bestimmen Sie im vorstehenden PL/SQL-Block den Wert und Datentyp für jeden der folgenden Fälle:
 - a. Wert von `v_customer` im verschachtelten Block:
 - b. Wert von `v_name` im verschachtelten Block:
 - c. Wert von `v_credit_rating` im verschachtelten Block:
 - d. Wert von `v_customer` im Hauptblock:
 - e. Wert von `v_name` im Hauptblock:
 - f. Wert von `v_credit_rating` im Hauptblock:
3. Verwenden Sie die Session, mit der Sie schon die Übungen in der Lektion "PL/SQL-Variablen deklarieren" ausgeführt haben. Wenn Sie eine neue Session geöffnet haben, führen Sie `lab_03_05_soln.sql` aus. Bearbeiten Sie `lab_03_05_soln.sql` anschließend wie folgt:
 - a. Kommentieren Sie die Zeilen, die die Bind-Variablen erstellen, mit einer einzeiligen Kommentarsyntax, und aktivieren Sie `SERVEROUTPUT`.
 - b. Kommentieren Sie im ausführbaren Bereich die Zeilen zum Zuweisen von Werten zu Bind-Variablen mit mehrzeiligen Kommentaren.
 - c. Im deklarativen Bereich:
 - 1) Deklarieren und initialisieren Sie zwei temporäre Variablen, um die auskommentierten Bind-Variablen zu ersetzen.
 - 2) Deklarieren Sie zwei weitere Variablen: `v_fname` vom Typ `VARCHAR2` und der Größe sowie `v_emp_sal` vom Typ `NUMBER` und der Größe 10.
 - d. Fügen Sie in den ausführbaren Bereich die folgende SQL-Anweisung ein:

```
SELECT first_name, salary INTO v_fname, v_emp_sal  
FROM employees WHERE employee_id=110;
```

- e. Ändern Sie die Zeile für die Anzeige von "Hello World" so, dass "Hello" und der Vorname angezeigt wird. Kommentieren Sie dann die Zeilen, in denen das Datum angezeigt wird, und zeigen Sie die Bind-Variablen an.
- f. Berechnen Sie den Beitrag der Mitarbeiter zur Unterstützungskasse. Der Anteil für die Unterstützungskasse liegt bei 12 % des Grundgehalts, und das Grundgehalt beträgt 45 % des Gehalts. Verwenden Sie für die Berechnung lokale Variablen. Verwenden Sie möglichst nur einen Ausdruck, um den Beitrag für die Unterstützungskasse zu berechnen. Geben Sie das Gehalt des Mitarbeiters und seinen Beitrag zur Unterstützungskasse aus.
- g. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_04_03_soln.sql`. Ausgabebeispiel:



```
Script Output x  
Task completed in 0.005 seconds  
anonymous block completed  
Hello John  
YOUR SALARY IS : 8200  
YOUR CONTRIBUTION TOWARDS PF:  
442.8
```

Übungen zu Lektion 4 – Lösung: Ausführbare Anweisungen erstellen

In dieser Übung prüfen und erstellen Sie ausführbare Anweisungen.

```
DECLARE
  v_weight      NUMBER(3) := 600;
  v_message     VARCHAR2(255) := 'Product 10012';
BEGIN
  DECLARE
    v_weight      NUMBER(3) := 1;
    v_message     VARCHAR2(255) := 'Product 11001';
    v_new_locn    VARCHAR2(50) := 'Europe';
  BEGIN
    v_weight := v_weight + 1;
    v_new_locn := 'Western ' || v_new_locn;
1 → END;
    v_weight := v_weight + 1;
    v_message := v_message || ' is in stock';
    v_new_locn := 'Western ' || v_new_locn;
2 → END;
/
```

1. Werten Sie den vorstehenden PL/SQL-Block aus, und bestimmen Sie den Datentyp und Wert jeder der folgenden Variablen entsprechend den Regeln für Gültigkeitsbereiche.
 - a. Wert von `v_weight` an 1. Position:
2
Datentyp: NUMBER
 - b. Wert von `v_new_locn` an 1. Position:
Western Europe
Datentyp: VARCHAR2
 - c. Wert von `v_weight` an 2. Position:
601
Datentyp: NUMBER
 - d. Wert von `v_message` an 2. Position:
Product 10012 is in stock
Datentyp: VARCHAR2
 - e. Wert von `v_new_locn` an 2. Position:
Unzulässig, da v_new_locn außerhalb des Unterblocks nicht sichtbar ist


```

DECLARE
    v_customer    VARCHAR2(50) := 'Womansport';
    v_credit_rating VARCHAR2(50) := 'EXCELLENT';
BEGIN
    DECLARE
        v_customer    NUMBER(7) := 201;
        v_name VARCHAR2(25) := 'Unisports';
    BEGIN
        v_credit_rating := 'GOOD';
        ...
    END;
    ...
END;

```

2. Bestimmen Sie im vorstehenden PL/SQL-Block den Wert und Datentyp für jeden der folgenden Fälle:
 - a. Wert von `v_customer` im verschachtelten Block:
201
Datentyp: NUMBER
 - b. Wert von `v_name` im verschachtelten Block:
Unisports
Datentyp: VARCHAR2
 - c. Wert von `v_credit_rating` im verschachtelten Block:
GOOD
Datentyp: VARCHAR2
 - d. Wert von `v_customer` im verschachtelten Block:
Womansport
Datentyp: VARCHAR2
 - e. Wert von `v_name` im verschachtelten Block:
NULL. name ist im Hauptblock nicht sichtbar, weswegen ein Fehler angezeigt würde.
 - f. Wert von `v_credit_rating` im Hauptblock:
EXCELLENT
Datentyp: VARCHAR2
3. Verwenden Sie die Session, mit der Sie schon die Übungen in der Lektion "PL/SQL-Variablen deklarieren" ausgeführt haben. Wenn Sie eine neue Session geöffnet haben, führen Sie `lab_03_05_soln.sql` aus. Bearbeiten Sie `lab_03_05_soln.sql` anschließend wie folgt:
 - a. Kommentieren Sie die Zeilen, die die Bind-Variablen erstellen, mit einer einzeiligen Kommentarsyntax, und aktivieren Sie `SERVEROUTPUT`.

```

-- VARIABLE b_basic_percent NUMBER
-- VARIABLE b_pf_percent NUMBER
SET SERVEROUTPUT ON

```

- b. Kommentieren Sie im ausführbaren Bereich die Zeilen zum Zuweisen von Werten zu Bind-Variablen mit mehrzeiligen Kommentaren.

```
/*:b_basic_percent:=45;
:b_pf_percent:=12;*/
```

- c. Im deklarativen Bereich:

- 1) Deklarieren und initialisieren Sie zwei temporäre Variablen, um die auskommentierten Bind-Variablen zu ersetzen.
- 2) Deklarieren Sie zwei weitere Variablen: v_fname vom Typ VARCHAR2 und der Größe sowie v_emp_sal vom Typ NUMBER und der Größe 10.

```
DECLARE
  v_basic_percent NUMBER:=45;
  v_pf_percent NUMBER:=12;
  v_fname VARCHAR2(15);
  v_emp_sal NUMBER(10);
```

- d. Fügen Sie die folgende SQL-Anweisung in den ausführbaren Bereich ein:

```
SELECT first_name, salary INTO v_fname, v_emp_sal
FROM employees WHERE employee_id=110;
```

- e. Ändern Sie die Zeile für die Anzeige von "Hello World" so, dass "Hello" und der Vorname angezeigt wird. Kommentieren Sie dann die Zeilen, in denen das Datum angezeigt wird, und zeigen Sie die Bind-Variablen an.

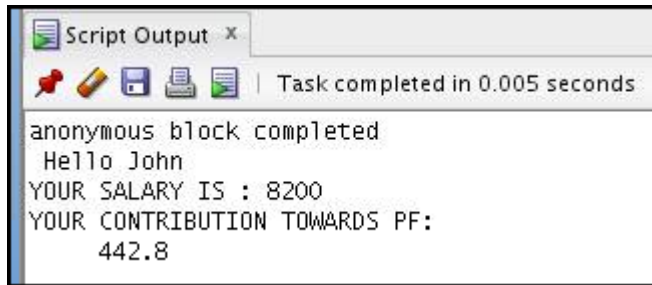
```
DBMS_OUTPUT.PUT_LINE(' Hello '|| v_fname);
/*  DBMS_OUTPUT.PUT_LINE('TODAY IS : '|| v_today);
DBMS_OUTPUT.PUT_LINE('TOMORROW IS : ' || v_tomorrow);*/
...
...

/
--PRINT b_basic_percent
--PRINT b_basic_percent
```

- f. Berechnen Sie den Beitrag der Mitarbeiter zur Unterstützungskasse. Der Anteil für die Unterstützungskasse liegt bei 12 % des Grundgehalts, und das Grundgehalt beträgt 45 % des Gehalts. Verwenden Sie für die Berechnung lokale Variablen. Verwenden Sie möglichst nur einen Ausdruck, um den Beitrag für die Unterstützungskasse zu berechnen. Geben Sie das Gehalt des Mitarbeiters und seinen Beitrag zur Unterstützungskasse aus.

```
DBMS_OUTPUT.PUT_LINE('YOUR SALARY IS : '||v_emp_sal);
DBMS_OUTPUT.PUT_LINE('YOUR CONTRIBUTION TOWARDS PF:
' ||v_emp_sal*v_basic_percent/100*v_pf_percent/100);
END;
```

- g. Führen Sie das Skript aus, und speichern Sie es unter dem Namen lab_04_03_soln.sql. Ausgabebeispiel:



The screenshot shows a 'Script Output' window with a title bar containing a close button. Below the title bar is a toolbar with icons for a pushpin, a pencil, a save icon, a print icon, and a document icon. To the right of the toolbar, it says 'Task completed in 0.005 seconds'. The main area of the window displays the following text:

```
anonymous block completed  
Hello John  
YOUR SALARY IS : 8200  
YOUR CONTRIBUTION TOWARDS PF:  
442.8
```


Übungen zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken

Kapitel 5

Übungen zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken

Hinweis: Wenn Sie die Codebeispiele für diese Lektion ausgeführt haben, müssen Sie den folgenden Code ausführen, bevor Sie mit dieser Übung beginnen:

```
DROP table employees2;  
DROP table copy_emp;
```

In dieser Übung verwenden Sie PL/SQL-Code, um mit dem Oracle-Server zu interagieren.

1. Erstellen Sie einen PL/SQL-Block, der in der Tabelle `departments` die höchste Abteilungsnummer wählt und in der Variable `v_max_deptno` speichert. Zeigen Sie die höchste Abteilungsnummer an.
 - a. Deklarieren Sie im deklarativen Bereich die Variable `v_max_deptno` vom Typ `NUMBER`.
 - b. Starten Sie den ausführbaren Bereich mit dem Schlüsselwort `BEGIN`. Nehmen Sie eine `SELECT`-Anweisung auf, mit der Sie die höchste Abteilungsnummer (`department_id`) aus der Tabelle `departments` abrufen können.
 - c. Zeigen Sie `v_max_deptno` an, und beenden Sie den ausführbaren Block.
 - d. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_05_01_soln.sql`. Ausgabebeispiel:

```
anonymous block completed  
The maximum department_id is : 270
```

2. Bearbeiten Sie den im 1. Schritt erstellten PL/SQL-Block, und fügen Sie eine neue Abteilung in die Tabelle `departments` ein.
 - a. Laden Sie das Skript `lab_05_01_soln.sql`. Deklarieren Sie zwei Variablen:
`v_dept_name` vom Typ `departments.department_name` und
`v_dept_id` vom Typ `NUMBER`
Weisen Sie `v_dept_name` im deklarativen Bereich `'Education'` zu.
 - b. Sie haben die derzeit höchste Abteilungsnummer bereits aus der Tabelle `departments` abgerufen. Erhöhen Sie die Zahl um 10, und weisen Sie das Ergebnis der Variablen `v_dept_id` zu.
 - c. Nehmen Sie eine `INSERT`-Anweisung auf, um Daten in die Spalten `department_name`, `department_id` und `location_id` der Tabelle `departments` einzufügen.
Verwenden Sie Werte in `v_dept_name` und `v_dept_id` für `department_name` und `department_id`, und verwenden Sie `NULL` für `location_id`.
 - d. Zeigen Sie mit dem SQL-Attribut `SQL%ROWCOUNT` die Anzahl der betroffenen Zeilen an.
 - e. Führen Sie eine `SELECT`-Anweisung aus, um zu prüfen, ob die neue Abteilung eingefügt wurde. Sie können den PL/SQL-Block mit `"/` abschließen und die `SELECT`-Anweisung in Ihr Skript aufnehmen.

- f. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_05_02_soln.sql`. Ausgabebeispiel:

```
anonymous block completed
The maximum department_id is : 270
SQL%ROWCOUNT gives 1

DEPARTMENT_ID DEPARTMENT_NAME          MANAGER_ID LOCATION_ID
-----
          280 Education
```

3. Im 2. Schritt haben Sie `location_id` auf `NULL` gesetzt. Erstellen Sie einen PL/SQL-Block, der die Standortnummer (`location_id`) für die neue Abteilung auf 3000 aktualisiert. **Hinweis:** Wenn Sie den 2. Schritt erfolgreich beendet haben, fahren Sie mit Schritt 3a fort. Andernfalls führen Sie erst das Lösungsskript `/soln/sol_05_02.sql` aus.
- Starten Sie den ausführbaren Block mit dem Schlüsselwort `BEGIN`. Nehmen Sie die `UPDATE`-Anweisung auf, um die Standortnummer (`location_id`) für die neue Abteilung (`v_dept_id = 280`) auf 3000 einzustellen.
 - Beenden Sie den ausführbaren Block mit dem Schlüsselwort `END`. Schließen Sie den PL/SQL-Block mit `"/` ab. Nehmen Sie eine `SELECT`-Anweisung auf, um die aktualisierte Abteilung anzuzeigen.
 - Nehmen Sie eine `DELETE`-Anweisung auf, um die hinzugefügte Abteilung zu löschen.
 - Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_05_03_soln.sql`. Ausgabebeispiel:

```
anonymous block completed
DEPARTMENT_ID DEPARTMENT_NAME          MANAGER_ID LOCATION_ID
-----
          280 Education                      3000

1 rows deleted.
```

Übungen zu Lektion 5 – Lösung: SQL-Anweisungen in PL/SQL-Blöcken

In dieser Übung verwenden Sie PL/SQL-Code, um mit dem Oracle-Server zu interagieren.

1. Erstellen Sie einen PL/SQL-Block, der in der Tabelle `departments` die höchste Abteilungsnummer wählt und in der Variable `v_max_deptno` speichert. Zeigen Sie die höchste Abteilungsnummer an.

- a. Deklarieren Sie im deklarativen Bereich die Variable `v_max_deptno` vom Typ `NUMBER`.

```
DECLARE
    v_max_deptno NUMBER;
```

- b. Starten Sie den ausführbaren Bereich mit dem Schlüsselwort `BEGIN`. Nehmen Sie eine `SELECT`-Anweisung auf, mit der Sie die höchste Abteilungsnummer (`department_id`) aus der Tabelle `departments` abrufen können.

```
BEGIN
    SELECT MAX(department_id) INTO v_max_deptno FROM
        departments;
```

- c. Zeigen Sie `v_max_deptno` an, und beenden Sie den ausführbaren Block.

```
DBMS_OUTPUT.PUT_LINE('The maximum department_id is : ' ||
    v_max_deptno);
END;
```

- d. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_05_01_soln.sql`. Ausgabebeispiel:

```
anonymous block completed
The maximum department_id is : 270
```

2. Bearbeiten Sie den im 1. Schritt erstellten PL/SQL-Block, und fügen Sie eine neue Abteilung in die Tabelle `departments` ein.

- a. Laden Sie das Skript `lab_05_01_soln.sql`. Deklarieren Sie zwei Variablen: `v_dept_name` vom Typ `departments.department_name` und `v_dept_id` vom Typ `NUMBER`. Weisen Sie `v_dept_name` im deklarativen Bereich `'Education'` zu.

```
v_dept_name departments.department_name%TYPE:= 'Education';
v_dept_id NUMBER;
```

- b. Sie haben die derzeit höchste Abteilungsnummer bereits aus der Tabelle `departments` abgerufen. Erhöhen Sie die Zahl um 10, und weisen Sie das Ergebnis der Variablen `v_dept_id` zu.

```
v_dept_id := 10 + v_max_deptno;
```


- c. Nehmen Sie eine INSERT-Anweisung auf, um Daten in die Spalten department_name, department_id und location_id der Tabelle departments einzufügen. Verwenden Sie Werte in v_dept_name und v_dept_id für department_name und department_id, und verwenden Sie NULL für location_id.

```
...
INSERT INTO departments (department_id, department_name,
location_id)
VALUES (v_dept_id, v_dept_name, NULL);
```

- d. Zeigen Sie mit dem SQL-Attribut SQL%ROWCOUNT die Anzahl der betroffenen Zeilen an.

```
DBMS_OUTPUT.PUT_LINE (' SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);
...
```

- e. Führen Sie eine SELECT-Anweisung aus, um zu prüfen, ob die neue Abteilung eingefügt wurde. Sie können den PL/SQL-Block mit "/" abschließen und die SELECT-Anweisung in Ihr Skript aufnehmen.

```
...
/
SELECT * FROM departments WHERE department_id= 280;
```

- f. Führen Sie das Skript aus, und speichern Sie es unter dem Namen lab_05_02_soln.sql. Ausgabebeispiel:

```
anonymous block completed
The maximum department_id is : 270
SQL%ROWCOUNT gives 1

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
-----
          280 Education
```

3. Im 2. Schritt haben Sie location_id auf NULL gesetzt. Erstellen Sie einen PL/SQL-Block, der die Standortnummer (location_id) für die neue Abteilung auf 3000 aktualisiert. **Hinweis:** Wenn Sie den 2. Schritt erfolgreich beendet haben, fahren Sie mit Schritt 3a fort. Andernfalls führen Sie erst das Lösungsskript /soln/sol_05_02.sql aus.

- a. Starten Sie den ausführbaren Block mit dem Schlüsselwort BEGIN. Nehmen Sie die UPDATE-Anweisung auf, um die Standortnummer (location_id) für die neue Abteilung (v_dept_id =280) auf 3000 einzustellen.

```
BEGIN
UPDATE departments SET location_id=3000 WHERE
department_id=280;
```

- b. Beenden Sie den ausführbaren Block mit dem Schlüsselwort END. Schließen Sie den PL/SQL-Block mit "/" ab. Nehmen Sie eine SELECT-Anweisung auf, um die aktualisierte Abteilung anzuzeigen.

```
END;
/
SELECT * FROM departments WHERE department_id=280;
```

- c. Nehmen Sie eine DELETE-Anweisung auf, um die hinzugefügte Abteilung zu löschen.

```
DELETE FROM departments WHERE department_id=280;
```

- d. Führen Sie das Skript aus, und speichern Sie es unter dem Namen lab_05_03_soln.sql. Ausgabebeispiel:

```
anonymous block completed
DEPARTMENT_ID DEPARTMENT_NAME          MANAGER_ID LOCATION_ID
-----
              280 Education                      3000
1 rows deleted.
```

Übungen zu Lektion 6: Kontrollstrukturen erstellen

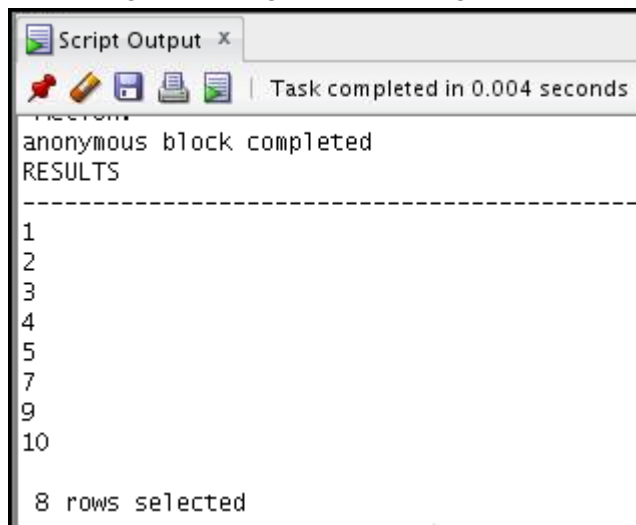
Kapitel 6

Übungen zu Lektion 6: Kontrollstrukturen erstellen

In diesen Übungen erstellen Sie PL/SQL-Blöcke, die Schleifen und bedingte Kontrollstrukturen enthalten. Sie erstellen dabei anhand des Gelernten verschiedene `IF`-Anweisungen und `LOOP`-Konstrukte.

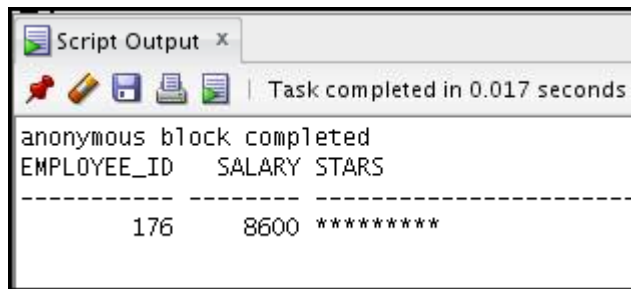
1. Führen Sie den Befehl in der Datei `lab_06_01.sql` aus, um die Tabelle `messages` zu erstellen. Erstellen Sie einen PL/SQL-Block, um Zahlen in die Tabelle `messages` einzufügen.
 - a. Fügen Sie die Zahlen 1 bis 10 außer 6 und 8 ein.
 - b. Führen Sie vor dem Ende des Blockes einen Commit-Befehl aus.
 - c. Führen Sie eine `SELECT`-Anweisung aus, um zu prüfen, ob der PL/SQL-Block funktioniert.

Ergebnis: Folgende Ausgabe sollte eingeblendet werden:



2. Führen Sie das Skript `lab_06_02.sql` aus. Dieses Skript erstellt die Tabelle `emp`, die ein Replikat der Tabelle `employees` ist. Es fügt in die Tabelle `emp` eine neue Spalte ein (`stars` vom Datentyp `VARCHAR2` und mit der Größe 50). Erstellen Sie einen PL/SQL-Block, der pro \$1000 des Mitarbeitergehalts ein Sternchen in die Spalte `stars` einfügt. Speichern Sie das Skript unter dem Namen `lab_06_02_soln.sql`.
 - a. Deklarieren Sie im deklarativen Bereich des Blockes die Variable `v_empno` vom Typ `emp.employee_id`, und initialisieren Sie sie auf 176. Deklarieren Sie die Variable `v_asterisk` vom Typ `emp.stars`, und initialisieren Sie sie auf `NULL`. Erstellen Sie die Variable `v_sal` vom Typ `emp.salary`.
 - b. Erstellen Sie im ausführbaren Bereich die Logik, um der Zeichenfolge pro 1.000 US-Dollar Gehalt ein Sternchen (*) hinzuzufügen. Wenn der Mitarbeiter beispielsweise 8.000 US-Dollar verdient, sollte die Zeichenfolge acht Sternchen enthalten. Verdient der Mitarbeiter 12.500 US-Dollar, sollte die Zeichenfolge 13 Sternchen enthalten (Rundung auf die nächste ganze Zahl).
 - c. Aktualisieren Sie die Spalte `stars` für den Mitarbeiter mit der Sternzeichenfolge. Führen Sie vor dem Ende des Blockes einen Commit-Befehl aus.
 - d. Zeigen Sie die Zeilen aus der Tabelle `emp` an, um zu prüfen, ob der PL/SQL-Block erfolgreich ausgeführt wurde.

- e. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_06_02_soln.sql`. Die Ausgabe lautet:



```
Script Output x
Task completed in 0.017 seconds
anonymous block completed
EMPLOYEE_ID  SALARY STARS
-----
          176      8600 *****
```

Übungen zu Lektion 6 – Lösung: Kontrollstrukturen erstellen

1. Führen Sie den Befehl in der Datei `lab_06_01.sql` aus, um die Tabelle `messages` zu erstellen. Erstellen Sie einen PL/SQL-Block, um Zahlen in die Tabelle `messages` einzufügen.

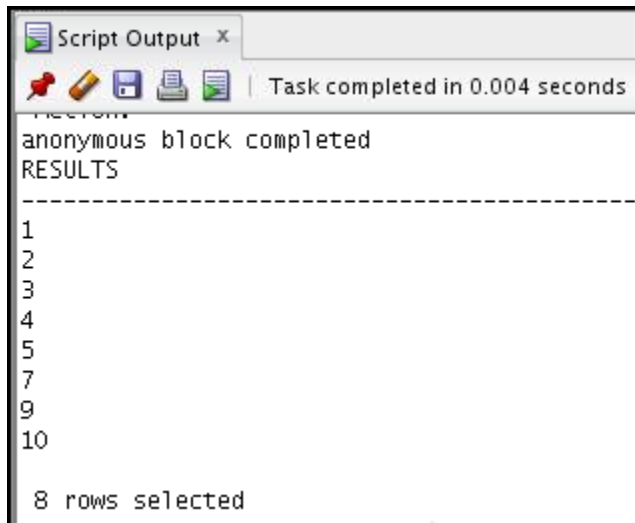
- a. Fügen Sie die Zahlen 1 bis 10 außer 6 und 8 ein.
- b. Führen Sie vor dem Ende des Blockes einen Commit-Befehl aus.

```
BEGIN
FOR i in 1..10 LOOP
  IF i = 6 or i = 8 THEN
    null;
  ELSE
    INSERT INTO messages(results)
    VALUES (i);
  END IF;
END LOOP;
COMMIT;
END;
/
```

- c. Führen Sie eine `SELECT`-Anweisung aus, um zu prüfen, ob der PL/SQL-Block funktioniert.

```
SELECT * FROM messages;
```

Ergebnis: Folgende Ausgabe sollte eingeblendet werden:



2. Führen Sie das Skript `lab_06_02.sql` aus. Dieses Skript erstellt die Tabelle `emp`, die ein Replikat der Tabelle `employees` ist. Es fügt in die Tabelle `emp` eine neue Spalte ein (`stars` vom Datentyp `VARCHAR2` und mit der Größe 50). Erstellen Sie einen PL/SQL-Block, der pro \$1000 des Mitarbeitergehalts ein Sternchen in die Spalte `stars` einfügt. Speichern Sie das Skript unter dem Namen `lab_06_02_soln.sql`.

- a. Deklarieren Sie im deklarativen Bereich des Blockes die Variable `v_empno` vom Typ `emp.employee_id`, und initialisieren Sie sie auf 176. Deklarieren Sie die Variable `v_asterisk` vom Typ `emp.stars`, und initialisieren Sie sie auf `NULL`. Erstellen Sie die Variable `v_sal` vom Typ `emp.salary`.

```
DECLARE
    v_empno      emp.employee_id%TYPE := 176;
    v_asterisk    emp.stars%TYPE := NULL;
    v_sal         emp.salary%TYPE;
```

- b. Erstellen Sie im ausführbaren Bereich die Logik, um der Zeichenfolge pro 1.000 US-Dollar Gehalt ein Sternchen (*) hinzuzufügen. Wenn der Mitarbeiter beispielsweise 8.000 US-Dollar verdient, sollte die Zeichenfolge acht Sternchen enthalten. Verdient der Mitarbeiter 12.500 US-Dollar, sollte die Zeichenfolge 13 Sternchen enthalten.

```
BEGIN
    SELECT NVL(ROUND(salary/1000), 0) INTO v_sal
    FROM emp WHERE employee_id = v_empno;

    FOR i IN 1..v_sal
        LOOP
            v_asterisk := v_asterisk || '*';
        END LOOP;
```

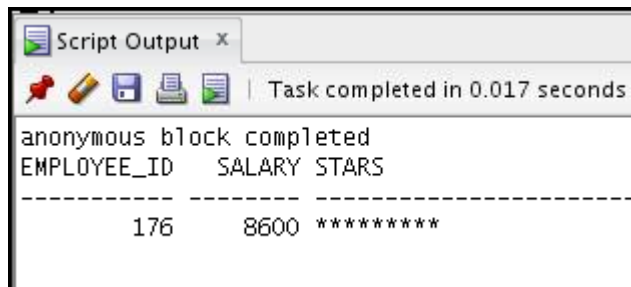
- c. Aktualisieren Sie die Spalte `stars` für den Mitarbeiter mit der Sternzeichenfolge. Führen Sie vor dem Ende des Blockes einen Commit-Befehl aus.

```
UPDATE emp SET stars = v_asterisk
WHERE employee_id = v_empno;
COMMIT;
END;
/
```

- d. Zeigen Sie die Zeilen aus der Tabelle `emp` an, um zu prüfen, ob der PL/SQL-Block erfolgreich ausgeführt wurde.

```
SELECT employee_id, salary, stars
FROM emp WHERE employee_id = 176;
```

- e. Führen Sie das Skript aus, und speichern Sie es unter dem Namen lab_06_02_soln.sql. Die Ausgabe lautet:



```
Script Output x
Task completed in 0.017 seconds
anonymous block completed
EMPLOYEE_ID  SALARY STARS
-----
          176      8600 *****
```


Übungen zu Lektion 7: Mit zusammengesetzten Datentypen arbeiten

Kapitel 7

Übungen zu Lektion 7: Mit zusammengesetzten Datentypen arbeiten

Hinweis: Wenn Sie die Codebeispiele für diese Lektion ausgeführt haben, müssen Sie den folgenden Code ausführen, bevor Sie mit dieser Übung beginnen:

```
DROP table retired_emps;
```

```
DROP table empl;
```

1. Erstellen Sie einen PL/SQL-Block, der Informationen über ein bestimmtes Land ausgibt.
 - a. Deklarieren Sie einen PL/SQL-Record basierend auf der Struktur der Tabelle COUNTRIES.
 - b. Deklarieren Sie die Variable `v_countryid`. Ordnen Sie CA zu `v_countryid` zu.
 - c. Deklarieren Sie im deklarativen Bereich mit dem Attribut `%ROWTYPE` die Variable `v_country_record` vom Typ `countries`.
 - d. Rufen Sie im ausführbaren Bereich mit `v_countryid` alle Informationen aus der Tabelle COUNTRIES ab. Zeigen Sie die gewählten Informationen zum Land an. Ausgabebeispiel:

```
anonymous block completed  
Country Id: CA Country Name: Canada Region: 2
```
 - e. Führen Sie den PL/SQL-Block für die Länder mit den IDs DE, UK und US aus, und testen Sie ihn.
2. Erstellen Sie einen PL/SQL-Block, um die Namen einiger Abteilungen aus der Tabelle DEPARTMENTS abzurufen, und geben Sie die einzelnen Abteilungsamen auf dem Bildschirm aus, indem Sie ein assoziatives Array aufnehmen. Speichern Sie das Skript unter dem Namen `lab_07_02_soln.sql`.
 - a. Deklarieren Sie eine INDEX BY-Tabelle namens `dept_table_type` vom Typ `departments.department_name`. Um die Namen der Abteilungen temporär zu speichern, deklarieren Sie eine `my_dept_table`-Variable vom Typ `dept_table_type`.
 - b. Deklarieren Sie zwei Variablen: `f_loop_count` und `v_deptno` vom Typ `NUMBER`. Ordnen Sie `f_loop_count` die Zahl 10 und `v_deptno` die Zahl 0 zu.

- c. Rufen Sie mithilfe einer Schleife die Namen von 10 Abteilungen ab, und speichern Sie die Namen im assoziativen Array. Beginnen Sie mit der `department_id` 10. Erhöhen Sie `v_deptno` bei jeder Iteration der Schleife um 10. Die folgende Tabelle zeigt die `department_id`, für die Sie `department_name` abrufen sollen.

| DEPARTMENT_ID | DEPARTMENT_NAME |
|---------------|------------------|
| 10 | Administration |
| 20 | Marketing |
| 30 | Purchasing |
| 40 | Human Resources |
| 50 | Shipping |
| 60 | IT |
| 70 | Public Relations |
| 80 | Sales |
| 90 | Executive |
| 100 | Finance |

- d. Rufen Sie mit einer weiteren Schleife die Abteilungsnamen aus dem assoziativen Array ab, und zeigen Sie sie an.
- e. Führen Sie das Skript aus, und speichern Sie es unter dem Namen `lab_07_02_soln.sql`. Die Ausgabe lautet:

```
anonymous block completed
Administration
Marketing
Purchasing
Human Resources
Shipping
IT
Public Relations
Sales
Executive
Finance
```

3. Bearbeiten Sie den in der 2. Aufgabe erstellten Block so, dass alle Informationen über die einzelnen Abteilungen aus der Tabelle `DEPARTMENTS` abgerufen und angezeigt werden. Verwenden Sie ein assoziatives Array mit der Methode der `INDEX BY-Record-Tabelle`.
- a. Laden Sie das Skript `lab_07_02_soln.sql`.
- b. Sie haben das assoziative Array als Typ `departments.department_name` deklariert. Ändern Sie die Deklaration des assoziativen Arrays, sodass die Nummer, der Name und der Standort aller Abteilungen temporär gespeichert werden. Verwenden Sie das Attribut `%ROWTYPE`.
- c. Ändern Sie die `SELECT`-Anweisung, um alle derzeit in der Tabelle `DEPARTMENTS` enthaltenen Abteilungsinformationen abzurufen und im assoziativen Array zu speichern.

- d. Rufen Sie mit einer weiteren Schleife die Abteilungsinformationen aus dem assoziativen Array ab, und zeigen Sie die Informationen an.

Ausgabebeispiel:

```
anonymous block completed
Department Number: 10 Department Name: Administration Manager Id: 200 Location Id: 1700
Department Number: 20 Department Name: Marketing Manager Id: 201 Location Id: 1800
Department Number: 30 Department Name: Purchasing Manager Id: 114 Location Id: 1700
Department Number: 40 Department Name: Human Resources Manager Id: 203 Location Id: 2400
Department Number: 50 Department Name: Shipping Manager Id: 121 Location Id: 1500
Department Number: 60 Department Name: IT Manager Id: 103 Location Id: 1400
Department Number: 70 Department Name: Public Relations Manager Id: 204 Location Id: 2700
Department Number: 80 Department Name: Sales Manager Id: 145 Location Id: 2500
Department Number: 90 Department Name: Executive Manager Id: 100 Location Id: 1700
Department Number: 100 Department Name: Finance Manager Id: 108 Location Id: 1700
```

Übungen zu Lektion 7 – Lösung: Mit zusammengesetzten Datentypen arbeiten

1. Erstellen Sie einen PL/SQL-Block, der Informationen über ein bestimmtes Land ausgibt.

- Deklarieren Sie einen PL/SQL-Record basierend auf der Struktur der Tabelle COUNTRIES.
- Deklarieren Sie die Variable `v_countryid`. Ordnen Sie CA zu `v_countryid` zu.

```
SET SERVEROUTPUT ON

SET VERIFY OFF
DECLARE
    v_countryid varchar2(20) := 'CA';
```

- Deklarieren Sie im deklarativen Bereich mit dem Attribut `%ROWTYPE` die Variable `v_country_record` vom Typ `countries`.

```
v_country_record countries%ROWTYPE;
```

- Rufen Sie im ausführbaren Bereich mit `v_countryid` alle Informationen aus der Tabelle COUNTRIES ab. Zeigen Sie die gewählten Informationen zum Land an.

```
BEGIN
    SELECT *
    INTO    v_country_record
    FROM    countries
    WHERE   country_id = UPPER(v_countryid);

    DBMS_OUTPUT.PUT_LINE ('Country Id: ' ||
        v_country_record.country_id ||
        ' Country Name: ' || v_country_record.country_name
        || ' Region: ' || v_country_record.region_id);

END;
```

Nach Ausführung aller obigen Schritte sieht das Ausgabebeispiel wie folgt aus:

```
anonymous block completed
Country Id: CA Country Name: Canada Region: 2
```

- Führen Sie den PL/SQL-Block für die Länder mit den IDs DE, UK und US aus, und testen Sie ihn.

2. Erstellen Sie einen PL/SQL-Block, um die Namen einiger Abteilungen aus der Tabelle `DEPARTMENTS` abzurufen, und geben Sie die einzelnen Abteilungsnamen auf dem Bildschirm aus, indem Sie ein assoziatives Array aufnehmen. Speichern Sie das Skript unter dem Namen `lab_07_02_soln.sql`.

- a. Deklarieren Sie eine INDEX BY-Tabelle namens `dept_table_type` vom Typ `departments.department_name`. Um die Namen der Abteilungen temporär zu speichern, deklarieren Sie eine `my_dept_table`-Variable vom Typ `dept_table_type`.

```
SET SERVEROUTPUT ON

DECLARE
    TYPE dept_table_type is table of
        departments.department_name%TYPE
    INDEX BY PLS_INTEGER;
    my_dept_table    dept_table_type;
```

- b. Deklarieren Sie zwei Variablen: `f_loop_count` und `v_deptno` vom Typ `NUMBER`. Ordnen Sie `f_loop_count` die Zahl 10 und `v_deptno` die Zahl 0 zu.

```
f_loop_count    NUMBER (2) :=10;
v_deptno        NUMBER (4) :=0;
```

- c. Rufen Sie mithilfe einer Schleife die Namen von 10 Abteilungen ab, und speichern Sie die Namen im assoziativen Array. Beginnen Sie mit der `department_id` 10. Erhöhen Sie `v_deptno` bei jeder Iteration der Schleife um 10. Die folgende Tabelle zeigt die `department_id`, für die Sie `department_name` abrufen und im assoziativen Array speichern sollen.

| DEPARTMENT_ID | DEPARTMENT_NAME |
|---------------|------------------|
| 10 | Administration |
| 20 | Marketing |
| 30 | Purchasing |
| 40 | Human Resources |
| 50 | Shipping |
| 60 | IT |
| 70 | Public Relations |
| 80 | Sales |
| 90 | Executive |
| 100 | Finance |

```

BEGIN
  FOR i IN 1..f_loop_count
  LOOP
    v_deptno:=v_deptno+10;
    SELECT department_name
    INTO my_dept_table(i)
    FROM departments
    WHERE department_id = v_deptno;
  END LOOP;

```

- d. Rufen Sie mit einer weiteren Schleife die Abteilungsnamen aus dem assoziativen Array ab, und zeigen Sie sie an.

```

FOR i IN 1..f_loop_count
  LOOP
    DBMS_OUTPUT.PUT_LINE (my_dept_table(i));
  END LOOP;
END;

```

- e. Führen Sie das Skript aus, und speichern Sie es unter dem Namen lab_07_02_soln.sql. Die Ausgabe lautet:

```

anonymous block completed
Administration
Marketing
Purchasing
Human Resources
Shipping
IT
Public Relations
Sales
Executive
Finance

```

3. Bearbeiten Sie den in der 2. Aufgabe erstellten Block so, dass alle Informationen über die einzelnen Abteilungen aus der Tabelle `DEPARTMENTS` abgerufen und angezeigt werden. Verwenden Sie ein assoziatives Array mit der Methode der `INDEX BY-Record-Tabelle`.
- Laden Sie das Skript lab_07_02_soln.sql.
 - Sie haben das assoziative Array als Typ `departments.department_name` deklariert. Ändern Sie die Deklaration des assoziativen Arrays, sodass die Nummer, der Name und der Standort aller Abteilungen temporär gespeichert werden. Verwenden Sie das Attribut `%ROWTYPE`.

```

SET SERVEROUTPUT ON

DECLARE
  TYPE dept_table_type is table of departments%ROWTYPE
  INDEX BY PLS_INTEGER;
  my_dept_table  dept_table_type;
  f_loop_count   NUMBER (2) :=10;
  v_deptno       NUMBER (4) :=0;

```

- c. Ändern Sie die SELECT-Anweisung, um alle derzeit in der Tabelle DEPARTMENTS enthaltenen Abteilungsinformationen abzurufen und im assoziativen Array zu speichern.

```
BEGIN
  FOR i IN 1..f_loop_count
  LOOP
    v_deptno := v_deptno + 10;
    SELECT *
    INTO my_dept_table(i)
    FROM departments
    WHERE department_id = v_deptno;
  END LOOP;
```

- d. Rufen Sie mit einer weiteren Schleife die Abteilungsinformationen aus dem assoziativen Array ab, und zeigen Sie die Informationen an.

```
FOR i IN 1..f_loop_count
LOOP
  DBMS_OUTPUT.PUT_LINE ('Department Number: ' ||
my_dept_table(i).department_id
  || ' Department Name: ' || my_dept_table(i).department_name
  || ' Manager Id: ' || my_dept_table(i).manager_id
  || ' Location Id: ' || my_dept_table(i).location_id);
END LOOP;
END;
```

Ausgabebeispiel:

```
anonymous block completed
Department Number: 10 Department Name: Administration Manager Id: 200 Location Id: 1700
Department Number: 20 Department Name: Marketing Manager Id: 201 Location Id: 1800
Department Number: 30 Department Name: Purchasing Manager Id: 114 Location Id: 1700
Department Number: 40 Department Name: Human Resources Manager Id: 203 Location Id: 2400
Department Number: 50 Department Name: Shipping Manager Id: 121 Location Id: 1500
Department Number: 60 Department Name: IT Manager Id: 103 Location Id: 1400
Department Number: 70 Department Name: Public Relations Manager Id: 204 Location Id: 2700
Department Number: 80 Department Name: Sales Manager Id: 145 Location Id: 2500
Department Number: 90 Department Name: Executive Manager Id: 100 Location Id: 1700
Department Number: 100 Department Name: Finance Manager Id: 108 Location Id: 1700
```


Übungen zu Lektion 8: Explizite Cursor

Kapitel 8

Übung 1 zu Lektion 8: Explizite Cursor

In dieser Übung führen Sie zwei Aufgaben aus:

- Zunächst verarbeiten Sie eine Reihe von Tabellenzeilen mit einem expliziten Cursor und füllen eine andere Tabelle mithilfe einer Cursor `FOR`-Schleife mit den Ergebnissen.
- Anschließend erstellen Sie einen PL/SQL-Block, der Informationen mit zwei Cursors verarbeitet. Einer der Cursor verwendet einen Parameter.

1. Erstellen Sie einen PL/SQL-Block, und führen Sie die folgenden Schritte aus:

- a. Deklarieren und initialisieren Sie im deklarativen Bereich die Variable `v_deptno` vom Typ `NUMBER`. Ordnen Sie einen gültigen Wert für die Abteilungs-ID zu. (Die Werte entnehmen Sie der Tabelle in Schritt d.)
- b. Deklarieren Sie den Cursor `c_emp_cursor`, der Nachnamen (`last_name`), Gehalt (`salary`) und Manager-ID (`manager_id`) der Mitarbeiter abrufen, die in der in `v_deptno` angegebenen Abteilung arbeiten.
- c. Verwenden Sie die abgerufenen Daten für einen Vorgang im ausführbaren Bereich mit einer Cursor `FOR`-Schleife. Wenn das Gehalt des Mitarbeiters unter 5000 liegt und die Manager-ID entweder 101 oder 124 ist, zeigen Sie die Meldung "`<<last_name>> Due for a raise`" an. Zeigen Sie andernfalls die Meldung "`<<last_name>> Not Due for a raise`" an.
- d. Testen Sie den PL/SQL-Block mit folgenden Werten:

| Department ID | Message |
|---------------|---|
| 10 | Whalen Due for a raise |
| 20 | Hartstein Not Due for a raise Fay Not Due for a raise |
| 50 | Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise. OConnell Due for a raise Grant Due for a raise |
| 80 | Russell Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise . . . Livingston Not Due for a raise Johnson Not Due for a raise |

2. Erstellen Sie dann einen PL/SQL-Block, in dem zwei Cursor (einer mit und einer ohne Parameter) deklariert und verwendet werden. Der erste Cursor ruft die Abteilungsnummer und den Abteilungsnamen aus der Tabelle `DEPARTMENTS` für alle Abteilungen ab, deren ID-Nummer kleiner als 100 ist. Der zweite Cursor ruft die Abteilungsnummer als einen Parameter und Details zu allen Mitarbeitern ab, die in dieser Abteilung arbeiten und deren Personalnummer (`employee_id`) kleiner als 120 ist.
- Deklariieren Sie den Cursor `c_dept_cursor`, mit dem Abteilungsnummer (`department_id`) und Abteilungsname (`department_name`) für Abteilungen abgerufen werden, deren Abteilungsnummer (`department_id`) kleiner als 100 ist. Sortieren Sie nach der Abteilungsnummer (`department_id`).
 - Deklariieren Sie einen weiteren Cursor `c_emp_cursor`, der die Abteilungsnummer als Parameter annimmt und für Mitarbeiter dieser Abteilung, deren Personalnummer (`employee_id`) kleiner als 120 ist, folgende Daten aus der Tabelle `EMPLOYEES` abrufen: Nachname (`last_name`), Tätigkeits-ID (`job_id`), Einstellungsdatum (`hire_date`) und Gehalt (`salary`).
 - Deklariieren Sie Variablen, die die von den einzelnen Cursors abgerufenen Werte aufnehmen. Deklarieren Sie die Variablen mit dem Attribut `%TYPE`.
 - Öffnen Sie `c_dept_cursor`, und verwenden Sie eine einfache Schleife, um Werte in die deklarierten Variablen zu lesen. Zeigen Sie die Abteilungsnummer und den Abteilungsnamen an. Verwenden Sie das entsprechende Cursorattribut, um die Schleife zu beenden.
 - Öffnen Sie `c_emp_cursor`, wobei Sie die aktuelle Abteilungsnummer als Parameter weitergeben. Starten Sie eine weitere Schleife, und lesen Sie die Werte von `emp_cursor` in Variablen ein. Geben Sie alle aus der Tabelle `EMPLOYEES` abgerufenen Details aus.

Hinweise

- Prüfen Sie vor dem Öffnen des Cursors, ob `c_emp_cursor` bereits geöffnet ist.
 - Verwenden Sie das entsprechende Cursorattribut als `EXIT`-Bedingung.
 - Wenn die Schleife beendet wird, geben Sie nach Anzeige der Details der einzelnen Abteilungen eine Zeile aus, und schließen Sie `c_emp_cursor`.
- f. Beenden Sie die erste Schleife, und schließen Sie `c_dept_cursor`. Beenden Sie dann den ausführbaren Bereich.

g. Führen Sie das Skript aus. Ausgabebeispiel:

| | | | |
|---|----------|-----------|-------|
| anonymous block completed | | | |
| Department Number : 10 Department Name : Administration | | | |
| ----- | | | |
| Department Number : 20 Department Name : Marketing | | | |
| ----- | | | |
| Department Number : 30 Department Name : Purchasing | | | |
| Raphaely | PU_MAN | 07-DEC-02 | 11000 |
| Khoo | PU_CLERK | 18-MAY-03 | 3100 |
| Baida | PU_CLERK | 24-DEC-05 | 2900 |
| Tobias | PU_CLERK | 24-JUL-05 | 2800 |
| Himuro | PU_CLERK | 15-NOV-06 | 2600 |
| Colmenares | PU_CLERK | 10-AUG-07 | 2500 |
| ----- | | | |
| Department Number : 40 Department Name : Human Resources | | | |
| ----- | | | |
| Department Number : 50 Department Name : Shipping | | | |
| ----- | | | |
| Department Number : 60 Department Name : IT | | | |
| Hunold | IT_PROG | 03-JAN-06 | 9000 |
| Ernst | IT_PROG | 21-MAY-07 | 6000 |
| Austin | IT_PROG | 25-JUN-05 | 4800 |
| Pataballa | IT_PROG | 05-FEB-06 | 4800 |
| Lorentz | IT_PROG | 07-FEB-07 | 4200 |
| ----- | | | |
| Department Number : 70 Department Name : Public Relations | | | |
| ----- | | | |
| Department Number : 80 Department Name : Sales | | | |
| ----- | | | |
| Department Number : 90 Department Name : Executive | | | |
| King | AD_PRES | 17-JUN-03 | 24000 |
| Kochhar | AD_VP | 21-SEP-05 | 17000 |
| De Haan | AD_VP | 13-JAN-01 | 17000 |

Übung 1 zu Lektion 8 – Lösung: Explizite Cursor

In dieser Übung führen Sie zwei Aufgaben aus:

- Zunächst verarbeiten Sie eine Reihe von Tabellenzeilen mit einem expliziten Cursor und füllen eine andere Tabelle mithilfe einer Cursor FOR-Schleife mit den Ergebnissen.
- Anschließend erstellen Sie einen PL/SQL-Block, der Informationen mit zwei Cursors verarbeitet. Einer der Cursor verwendet einen Parameter.

1. Erstellen Sie einen PL/SQL-Block, und führen Sie die folgenden Schritte aus:

- a. Deklarieren und initialisieren Sie im deklarativen Bereich die Variable `v_deptno` vom Typ `NUMBER`. Ordnen Sie einen gültigen Wert für die Abteilungs-ID zu. (Die Werte entnehmen Sie der Tabelle in Schritt d.)

```
DECLARE
v_deptno NUMBER := 10;
```

- b. Deklarieren Sie den Cursor `c_emp_cursor`, der Nachnamen (`last_name`), Gehalt (`salary`) und Manager-ID (`manager_id`) der Mitarbeiter abrufen, die in der in `v_deptno` angegebenen Abteilung arbeiten.

```
CURSOR c_emp_cursor IS
SELECT      last_name, salary, manager_id
FROM        employees
WHERE       department_id = v_deptno;
```

- c. Verwenden Sie die abgerufenen Daten für einen Vorgang im ausführbaren Bereich mit einer Cursor FOR-Schleife. Wenn das Gehalt des Mitarbeiters unter 5000 liegt und die Manager-ID entweder 101 oder 124 ist, zeigen Sie die Meldung "<<*last_name*>> Due for a raise" an. Zeigen Sie andernfalls die Meldung "<<*last_name*>> Not Due for a raise" an.

```
BEGIN
FOR emp_record IN c_emp_cursor
LOOP
IF emp_record.salary < 5000 AND (emp_record.manager_id=101 OR
emp_record.manager_id=124) THEN
DBMS_OUTPUT.PUT_LINE (emp_record.last_name || ' Due for a
raise');
ELSE
DBMS_OUTPUT.PUT_LINE (emp_record.last_name || ' Not Due for a
raise');
END IF;
END LOOP;
END;
```

d. Testen Sie den PL/SQL-Block mit folgenden Werten:

| Department ID | Message |
|---------------|---|
| 10 | Whalen Due for a raise |
| 20 | Hartstein Not Due for a raise Fay Not Due for a raise |
| 50 | Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise. OConnell Due for a raise Grant Due for a raise |
| 80 | Russell Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise . . . Livingston Not Due for a raise Johnson Not Due for a raise |

2. Erstellen Sie dann einen PL/SQL-Block, in dem zwei Cursor (einer mit und einer ohne Parameter) deklariert und verwendet werden. Der erste Cursor ruft die Abteilungsnummer und den Abteilungsnamen aus der Tabelle `DEPARTMENTS` für alle Abteilungen ab, deren ID-Nummer kleiner als 100 ist. Der zweite Cursor ruft die Abteilungsnummer als einen Parameter und Details zu allen Mitarbeitern ab, die in dieser Abteilung arbeiten und deren Personalnummer (`employee_id`) kleiner als 120 ist.

a. Deklarieren Sie den Cursor `c_dept_cursor`, mit dem Abteilungsnummer (`department_id`) und Abteilungsname (`department_name`) für Abteilungen abgerufen werden, deren Abteilungsnummer (`department_id`) kleiner als 100 ist. Sortieren Sie nach der Abteilungsnummer (`department_id`).

```
DECLARE
  CURSOR c_dept_cursor IS
    SELECT department_id, department_name
    FROM departments
    WHERE department_id < 100
    ORDER BY department_id;
```

- b. Deklarieren Sie einen weiteren Cursor `c_emp_cursor`, der die Abteilungsnummer als Parameter annimmt und für Mitarbeiter dieser Abteilung, deren Personalnummer (`employee_id`) kleiner als 120 ist, folgende Daten aus der Tabelle `EMPLOYEES` abrufen: Nachname (`last_name`), Tätigkeits-ID (`job_id`), Einstellungsdatum (`hire_date`) und Gehalt (`salary`) `employee_id`.

```
CURSOR c_emp_cursor(v_deptno NUMBER) IS
    SELECT last_name, job_id, hire_date, salary
    FROM employees
    WHERE department_id = v_deptno
    AND employee_id < 120;
```

- c. Deklarieren Sie Variablen, die die von den einzelnen Cursors abgerufenen Werte aufnehmen. Deklarieren Sie die Variablen mit dem Attribut `%TYPE`.

```
v_current_deptno departments.department_id%TYPE;
v_current_dname departments.department_name%TYPE;
v_ename employees.last_name%TYPE;
v_job employees.job_id%TYPE;
v_hiredate employees.hire_date%TYPE;
v_sal employees.salary%TYPE;
```

- d. Öffnen Sie `c_dept_cursor`, und verwenden Sie eine einfache Schleife, um Werte in die deklarierten Variablen zu lesen. Zeigen Sie die Abteilungsnummer und den Abteilungs-namen an. Verwenden Sie das entsprechende Cursorattribut, um die Schleife zu beenden.

```
BEGIN
    OPEN c_dept_cursor;
    LOOP
        FETCH c_dept_cursor INTO v_current_deptno,
            v_current_dname;
        EXIT WHEN c_dept_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE ('Department Number : ' ||
            v_current_deptno || ' Department Name : ' ||
            v_current_dname);
    END LOOP;
```

- e. Öffnen Sie `c_emp_cursor`, wobei Sie die aktuelle Abteilungsnummer als Parameter weitergeben. Starten Sie eine weitere Schleife, und lesen Sie die Werte von `emp_cursor` in Variablen ein. Geben Sie alle aus der Tabelle `EMPLOYEES` abgerufenen Details aus.

Hinweise

- Prüfen Sie vor dem Öffnen des Cursors, ob `c_emp_cursor` bereits geöffnet ist.
- Verwenden Sie das entsprechende Cursorattribut als `EXIT`-Bedingung.
- Wenn die Schleife beendet wird, geben Sie nach Anzeige der Details der einzelnen Abteilungen eine Zeile aus, und schließen Sie `c_emp_cursor`.

```
IF c_emp_cursor%ISOPEN THEN
    CLOSE c_emp_cursor;
END IF;
OPEN c_emp_cursor (v_current_deptno);
LOOP
    FETCH c_emp_cursor INTO v_ename,v_job,v_hiredate,v_sal;
    EXIT WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (v_ename || ' ' || v_job
                          || ' ' || v_hiredate || ' ' || v_sal);
END LOOP;
DBMS_OUTPUT.PUT_LINE('-----');
--
CLOSE c_emp_cursor;
```

- f. Beenden Sie die erste Schleife, und schließen Sie `c_dept_cursor`. Beenden Sie dann den ausführbaren Bereich.

```
END LOOP;
CLOSE c_dept_cursor;
END;
```


g. Führen Sie das Skript aus. Ausgabebeispiel:

| | | | |
|---|----------|-----------|-------|
| anonymous block completed | | | |
| Department Number : 10 Department Name : Administration | | | |
| ----- | | | |
| Department Number : 20 Department Name : Marketing | | | |
| ----- | | | |
| Department Number : 30 Department Name : Purchasing | | | |
| Raphaely | PU_MAN | 07-DEC-02 | 11000 |
| Khoo | PU_CLERK | 18-MAY-03 | 3100 |
| Baida | PU_CLERK | 24-DEC-05 | 2900 |
| Tobias | PU_CLERK | 24-JUL-05 | 2800 |
| Himuro | PU_CLERK | 15-NOV-06 | 2600 |
| Colmenares | PU_CLERK | 10-AUG-07 | 2500 |
| ----- | | | |
| Department Number : 40 Department Name : Human Resources | | | |
| ----- | | | |
| Department Number : 50 Department Name : Shipping | | | |
| ----- | | | |
| Department Number : 60 Department Name : IT | | | |
| Hunold | IT_PROG | 03-JAN-06 | 9000 |
| Ernst | IT_PROG | 21-MAY-07 | 6000 |
| Austin | IT_PROG | 25-JUN-05 | 4800 |
| Pataballa | IT_PROG | 05-FEB-06 | 4800 |
| Lorentz | IT_PROG | 07-FEB-07 | 4200 |
| ----- | | | |
| Department Number : 70 Department Name : Public Relations | | | |
| ----- | | | |
| Department Number : 80 Department Name : Sales | | | |
| ----- | | | |
| Department Number : 90 Department Name : Executive | | | |
| King | AD_PRES | 17-JUN-03 | 24000 |
| Kochhar | AD_VP | 21-SEP-05 | 17000 |
| De Haan | AD_VP | 13-JAN-01 | 17000 |

Übung 2 zu Lektion 8: Optionale Übung zu expliziten Cursors

Führen Sie die folgende optionale Übung durch, falls Sie noch Zeit haben. Erstellen Sie hier einen PL/SQL-Block, der die höchsten n Mitarbeitergehälter mithilfe eines expliziten Cursors bestimmt.

1. Führen Sie das Skript `lab_08-02.sql` aus, um die Tabelle `TOP_SALARIES` zu erstellen, in der die Mitarbeitergehälter gespeichert werden.
2. Deklarieren Sie im deklarativen Bereich die Variable `v_num` vom Typ `NUMBER`, in der ein Wert n enthalten ist, der für die Anzahl der n Spitzenverdiener aus der Tabelle `employees` steht. Beispiel: Um die fünf höchsten Gehälter anzuzeigen, geben Sie 5 ein. Deklarieren Sie eine weitere Variable `sal` vom Typ `employees.salary`. Deklarieren Sie den Cursor `c_emp_cursor`, der die Gehälter der Mitarbeiter in absteigender Reihenfolge abruft. Die Gehälter dürfen nicht dupliziert werden.
3. Öffnen Sie die Schleife im ausführbaren Bereich. Rufen Sie die höchsten n Gehälter ab, und fügen Sie sie in die Tabelle `TOP_SALARIES` ein. Sie können die Daten mit einer einfachen Schleife bearbeiten. Probieren Sie außerdem die Attribute `%ROWCOUNT` und `%FOUND` für die `EXIT`-Bedingung aus.

Hinweis: Sie müssen eine `EXIT`-Bedingung hinzufügen, um keine unendliche Schleife zu erhalten.

4. Nachdem Sie die Daten in die Tabelle `TOP_SALARIES` eingefügt haben, zeigen Sie die Zeilen mit einer `SELECT`-Anweisung an. Die Ausgabe zeigt die fünf höchsten Gehälter in der Tabelle `EMPLOYEES` an.

| SALARY |
|--------|
| ----- |
| 24000 |
| 17000 |
| 17000 |
| 14000 |
| 13500 |

5. Testen Sie einige Sonderfälle wie `v_num = 0` oder den Fall, in dem `v_num` größer ist als die Anzahl der Mitarbeiter in der Tabelle `EMPLOYEES` ist. Leeren Sie die Tabelle `TOP_SALARIES` nach jedem Test.

Übung 2 zu Lektion 8 – Lösung: Optionale Übung zu expliziten Cursorn

Führen Sie die folgende optionale Übung durch, falls Sie noch Zeit haben. Erstellen Sie hier einen PL/SQL-Block, der die höchsten n Mitarbeitergehälter mithilfe eines expliziten Cursors bestimmt.

1. Führen Sie das Skript `lab_08-02.sql` aus, um eine neue Tabelle `TOP_SALARIES` zu erstellen, in der die Mitarbeitergehälter gespeichert werden.
2. Deklarieren Sie im deklarativen Bereich die Variable `v_num` vom Typ `NUMBER`, in der ein Wert n enthalten ist, der für die Anzahl der n Spitzenverdiener aus der Tabelle `EMPLOYEES` steht. Beispiel: Um die fünf höchsten Gehälter anzuzeigen, geben Sie 5 ein. Deklarieren Sie eine weitere Variable `sal` vom Typ `employees.salary`. Deklarieren Sie den Cursor `c_emp_cursor`, der die Gehälter der Mitarbeiter in absteigender Reihenfolge abruft. Die Gehälter dürfen nicht dupliziert werden.

```
DECLARE
  v_num          NUMBER(3) := 5;
  v_sal          employees.salary%TYPE;
  CURSOR c_emp_cursor IS
    SELECT salary
    FROM employees
    ORDER BY salary DESC;
```

3. Öffnen Sie die Schleife im ausführbaren Bereich. Rufen Sie die höchsten n Gehälter ab, und fügen Sie sie in die Tabelle `TOP_SALARIES` ein. Sie können die Daten mit einer einfachen Schleife bearbeiten. Probieren Sie außerdem die Attribute `%ROWCOUNT` und `%FOUND` für die `EXIT`-Bedingung aus.

Hinweis: Sie müssen eine `EXIT`-Bedingung hinzufügen, um keine unendliche Schleife zu erhalten.

```
BEGIN
  OPEN c_emp_cursor;
  FETCH c_emp_cursor INTO v_sal;
  WHILE c_emp_cursor%ROWCOUNT <= v_num AND c_emp_cursor%FOUND LOOP
    INSERT INTO top_salaries (salary)
      VALUES (v_sal);
    FETCH c_emp_cursor INTO v_sal;
  END LOOP;
  CLOSE c_emp_cursor;
END;
```

4. Nachdem Sie die Daten in die Tabelle `TOP_SALARIES` eingefügt haben, zeigen Sie die Zeilen mit einer `SELECT`-Anweisung an. Die Ausgabe zeigt die fünf höchsten Gehälter in der Tabelle `EMPLOYEES` an.

```
/
SELECT * FROM top_salaries;
```

Ausgabebeispiel:

| SALARY |
|--------|
| ----- |
| 24000 |
| 17000 |
| 17000 |
| 14000 |
| 13500 |

5. Testen Sie einige Sonderfälle wie $v_num = 0$ oder den Fall, in dem v_num größer ist als die Anzahl der Mitarbeiter in der Tabelle `EMPLOYEES` ist. Leeren Sie die Tabelle `TOP_SALARIES` nach jedem Test.

Übungen zu Lektion 9: Exceptions behandeln

Kapitel 9

Übung 1 zu Lektion 9: Vordefinierte Exceptions behandeln

In dieser Übung erstellen Sie einen PL/SQL-Block, der eine vordefinierte Exception anwendet und damit einen Record verarbeitet. Der PL/SQL-Block wählt den Namen des Mitarbeiters mit einem gegebenen Gehaltswert.

1. Führen Sie den Befehl in der Datei `lab_06_01.sql` aus, um die Tabelle `MESSAGES` neu zu erstellen.
2. Deklarieren Sie im deklarativen Bereich zwei Variablen: `v_ename` vom Typ `employees.last_name` und `v_emp_sal` vom Typ `employees.salary`. Initialisieren Sie die zweite Variable mit 6000.
3. Rufen Sie im ausführbaren Bereich die Nachnamen der Mitarbeiter ab, deren Gehalt dem Wert in `v_emp_sal` entspricht. Wenn das eingegebene Gehalt nur eine Zeile zurückgibt, geben Sie in die Tabelle `MESSAGES` den Namen des Mitarbeiters und die Höhe seines Gehalts ein.
Hinweis: Verwenden Sie keine expliziten Cursor.
4. Wenn das eingegebene Gehalt keine Zeilen zurückgibt, behandeln Sie die Exception mit dem entsprechenden Exception Handler und fügen in die Tabelle `MESSAGES` die Meldung "No employee with a salary of <salary>" ein.
5. Wenn das eingegebene Gehalt mehrere Zeilen zurückgibt, behandeln Sie die Exception mit einem entsprechenden Exception Handler und fügen in die Tabelle `MESSAGES` die Meldung "More than one employee with a salary of <salary>" ein.
6. Behandeln Sie beliebige weitere Exceptions mit einem entsprechenden Exception Handler, und fügen Sie in die Tabelle `MESSAGES` die Meldung "Some other error occurred" ein.
7. Um zu prüfen, ob der PL/SQL-Block erfolgreich ausgeführt wurde, zeigen Sie die Zeilen aus der Tabelle `MESSAGES` an. Die Ausgabe lautet:

| |
|--|
| RESULTS |
| ----- |
| More than one employee with a salary of 6000 |
| 1 rows selected |

8. Ändern Sie den initialisierten Wert von `v_emp_sal` in 2000, und führen Sie den Befehl erneut aus. Die Ausgabe lautet:

| |
|--|
| RESULTS |
| ----- |
| More than one employee with a salary of 6000 |
| No employee with a salary of 2000 |

Übung 1 zu Lektion 9 – Lösung: Vordefinierte Exceptions behandeln

In dieser Übung erstellen Sie einen PL/SQL-Block, der eine vordefinierte Exception anwendet und damit einen Record verarbeitet. Der PL/SQL-Block wählt den Namen des Mitarbeiters mit einem gegebenen Gehaltswert.

1. Führen Sie den Befehl in der Datei `lab_06_01.sql` aus, um die Tabelle `MESSAGES` neu zu erstellen.
2. Deklarieren Sie im deklarativen Bereich zwei Variablen: `v_ename` vom Typ `employees.last_name` und `v_emp_sal` vom Typ `employees.salary`. Initialisieren Sie die zweite Variable mit 6000.

```
DECLARE
    v_ename      employees.last_name%TYPE;
    v_emp_sal    employees.salary%TYPE := 6000;
```

3. Rufen Sie im ausführbaren Bereich die Nachnamen der Mitarbeiter ab, deren Gehalt dem Wert in `v_emp_sal` entspricht. Wenn das eingegebene Gehalt nur eine Zeile zurückgibt, geben Sie in die Tabelle `MESSAGES` den Namen des Mitarbeiters und die Höhe seines Gehalts ein.

Hinweis: Verwenden Sie keine expliziten Cursor.

```
BEGIN
    SELECT      last_name
    INTO        v_ename
    FROM        employees
    WHERE       salary = v_emp_sal;
    INSERT INTO messages (results)
    VALUES (v_ename || ' - ' || v_emp_sal);
```

4. Wenn das eingegebene Gehalt keine Zeilen zurückgibt, behandeln Sie die Exception mit dem entsprechenden Exception Handler und fügen in die Tabelle `MESSAGES` die Meldung "No employee with a salary of <salary>" ein.

```
EXCEPTION
    WHEN no_data_found THEN
        INSERT INTO messages (results)
        VALUES ('No employee with a salary of ' ||
                TO_CHAR(v_emp_sal));
```

5. Wenn das eingegebene Gehalt mehrere Zeilen zurückgibt, behandeln Sie die Exception mit einem entsprechenden Exception Handler und fügen in die Tabelle `MESSAGES` die Meldung "More than one employee with a salary of <salary>" ein.

```
WHEN too_many_rows THEN
    INSERT INTO messages (results)
    VALUES ('More than one employee with a salary of ' ||
            TO_CHAR(v_emp_sal));
```

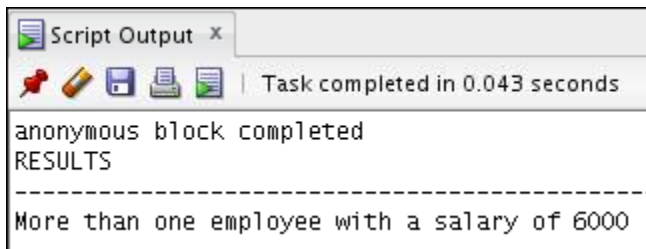
6. Behandeln Sie beliebige weitere Exceptions mit einem entsprechenden Exception Handler, und fügen Sie in die Tabelle `MESSAGES` die Meldung "Some other error occurred" ein.

```
WHEN others THEN
    INSERT INTO messages (results)
    VALUES ('Some other error occurred. ');
END;
```

7. Um zu prüfen, ob der PL/SQL-Block erfolgreich ausgeführt wurde, zeigen Sie die Zeilen aus der Tabelle `MESSAGES` an.

```
/
SELECT * FROM messages;
```

Die Ausgabe lautet:



8. Ändern Sie den initialisierten Wert von `v_emp_sal` in 2000, und führen Sie den Befehl erneut aus. Die Ausgabe lautet:

```
RESULTS
-----
More than one employee with a salary of 6000
No employee with a salary of 2000
```


Übung 2 zu Lektion 9: Standardmäßige Oracle-Server-Exceptions behandeln

In dieser Übung erstellen Sie einen PL/SQL-Block, der eine Exception für Oracle-Serverfehler ORA-02292 (`integrity constraint violated - child record found`) deklariert. Der Block prüft das Vorliegen der Exception und gibt die Fehlermeldung aus.

1. Deklarieren Sie im deklarativen Bereich die Exception `e_childrecord_exists`. Ordnen Sie die deklarierte Exception dem standardmäßigen Oracle-Serverfehler `-02292` zu.
2. Zeigen Sie im ausführbaren Bereich "Deleting department 40..." an. Um die Abteilung mit der `department_id` 40 zu löschen, nehmen Sie eine `DELETE`-Anweisung auf.
3. Um die Exception `e_childrecord_exists` zu behandeln und die entsprechende Meldung anzuzeigen, nehmen Sie einen Exception-Bereich auf.

Ausgabebeispiel:

| | |
|---|--|
| <pre>anonymous block completed Deleting department 40..... Cannot delete this department. There are employees in this department (child records exist.)</pre> | |
|---|--|

Übung 2 zu Lektion 9 – Lösung: Standardmäßige Oracle-Server-Exceptions behandeln

In dieser Übung erstellen Sie einen PL/SQL-Block, der eine Exception für Oracle-Serverfehler ORA-02292 (integrity constraint violated - child record found) deklariert. Der Block prüft das Vorliegen der Exception und gibt die Fehlermeldung aus.

1. Deklarieren Sie im deklarativen Bereich die Exception `e_childrecord_exists`. Ordnen Sie die deklarierte Exception dem standardmäßigen Oracle-Serverfehler `-02292` zu.

```
SET SERVEROUTPUT ON
DECLARE
    e_childrecord_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(e_childrecord_exists, -02292);
```

2. Zeigen Sie im ausführbaren Bereich "Deleting department 40..." an. Um die Abteilung mit der `department_id` 40 zu löschen, nehmen Sie eine `DELETE`-Anweisung auf.

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(' Deleting department 40.....');
    delete from departments where department_id=40;
```

3. Um die Exception `e_childrecord_exists` zu behandeln und die entsprechende Meldung anzuzeigen, nehmen Sie einen Exception-Bereich auf.

```
EXCEPTION
    WHEN e_childrecord_exists THEN
        DBMS_OUTPUT.PUT_LINE(' Cannot delete this department. There are
employees in this department (child records exist.) ');
END;
```

Ausgabebeispiel:

```
anonymous block completed
Deleting department 40.....
Cannot delete this department. There are employees in this department (child records exist.)
```

Übungen zu Lektion 10: Stored Procedures und Stored Functions – Einführung

Kapitel 10

Übungen zu Lektion 10: Stored Procedures erstellen und verwenden

Hinweis: Wenn Sie die Codebeispiele für diese Lektion ausgeführt haben, müssen Sie den folgenden Code ausführen, bevor Sie mit dieser Übung beginnen:

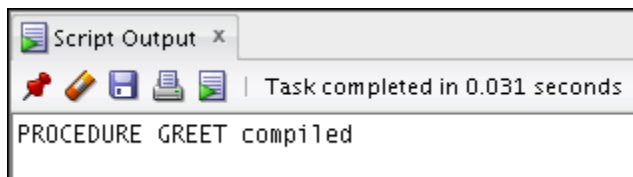
```
DROP table dept;  
DROP procedure add_dept;  
DROP function check_sal;
```

In dieser Übung erstellen und verwenden Sie Stored Procedures in vorhandenen Skripten.

1. Öffnen Sie das Skript `sol_03.sql` aus dem Ordner `/home/oracle/labs/plsf/soln/`. Kopieren Sie den Code für die 4. Aufgabe in ein neues Arbeitsblatt.

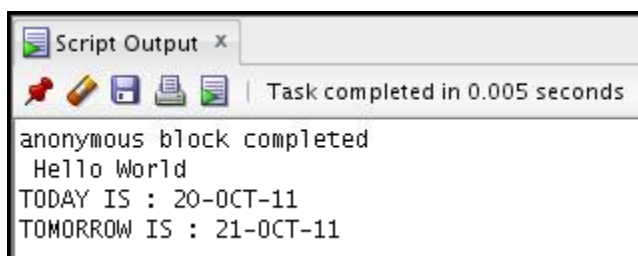
```
SET SERVEROUTPUT ON  
  
DECLARE  
    v_today DATE:=SYSDATE;  
    v_tomorrow v_today%TYPE;  
BEGIN  
    v_tomorrow:=v_today +1;  
    DBMS_OUTPUT.PUT_LINE(' Hello World ');  
    DBMS_OUTPUT.PUT_LINE('TODAY IS : ' || v_today);  
    DBMS_OUTPUT.PUT_LINE('TOMORROW IS : ' || v_tomorrow);  
END;
```

- a. Bearbeiten Sie das Skript, um den anonymen Block in eine Prozedur namens `greet` zu konvertieren.
(**Tipp:** Entfernen Sie auch den Befehl `SET SERVEROUTPUT ON`.)
- b. Führen Sie das Skript aus, um die Prozedur zu erstellen. Die Ausgabe soll in etwa wie folgt angezeigt werden:



- c. Speichern Sie das Skript unter dem Namen `lab_10_01_soln.sql`.
- d. Um den Workspace zu leeren, klicken Sie auf die Schaltfläche **Clear**.
- e. Um die Prozedur `greet` aufzurufen, erstellen Sie einen anonymen Block und führen ihn aus.
(**Tipp:** Achten Sie darauf, dass am Blockanfang `SERVEROUTPUT` aktiviert wird.)

Die Ausgabe sollte wie folgt aussehen:



2. Bearbeiten Sie wie folgt das Skript `lab_10_01_soln.sql`:

a. Löschen Sie die Prozedur `greet` mit dem folgenden Befehl:

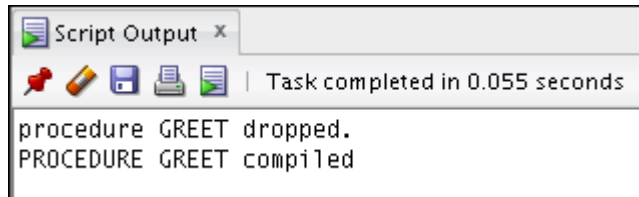
```
DROP PROCEDURE greet;
```

b. Bearbeiten Sie die Prozedur so, dass sie ein Argument vom Typ `VARCHAR2` annimmt. Nennen Sie das Argument `p_name`.

c. Geben Sie `Hello <name>` (also den Inhalt des Arguments) anstelle von `Hello World` aus.

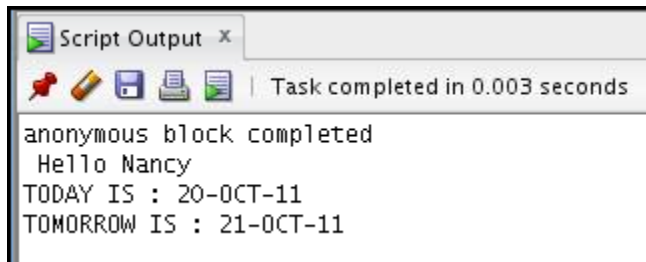
d. Speichern Sie Ihr Skript unter dem Namen `lab_10_02_soln.sql`.

e. Führen Sie das Skript aus, um die Prozedur zu erstellen. Die Ausgabe soll in etwa wie folgt angezeigt werden:



f. Um die Prozedur `greet` mit einem Parameterwert aufzurufen, erstellen Sie einen anonymen Block und führen ihn aus. Der Block sollte auch die Ausgabe erzeugen.

Die Beispielausgabe sollte wie folgt aussehen:



Übungen zu Lektion 10 – Lösung: Stored Procedures erstellen und verwenden

In dieser Übung erstellen und verwenden Sie Stored Procedures in vorhandenen Skripten.

1. Öffnen Sie das Skript `sol_03.sql` aus dem Ordner `/home/oracle/labs/plsf/soln/`. Kopieren Sie den Code für die 4. Aufgabe in ein neues Arbeitsblatt.

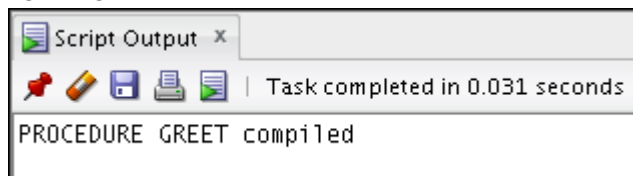
```
SET SERVEROUTPUT ON

DECLARE
    v_today DATE:=SYSDATE;
    v_tomorrow v_today%TYPE;
BEGIN
    v_tomorrow:=v_today +1;
    DBMS_OUTPUT.PUT_LINE(' Hello World ');
    DBMS_OUTPUT.PUT_LINE('TODAY IS : ' || v_today);
    DBMS_OUTPUT.PUT_LINE('TOMORROW IS : ' || v_tomorrow);
END;
```

- a. Bearbeiten Sie das Skript, um den anonymen Block in eine Prozedur namens `greet` zu konvertieren. (**Tip**: Entfernen Sie auch den Befehl `SET SERVEROUTPUT ON`.)

```
CREATE PROCEDURE greet IS
    V_today DATE:=SYSDATE;
    V_tomorrow today%TYPE;
...
```

- b. Führen Sie das Skript aus, um die Prozedur zu erstellen. Die Ausgabe soll in etwa wie folgt angezeigt werden:

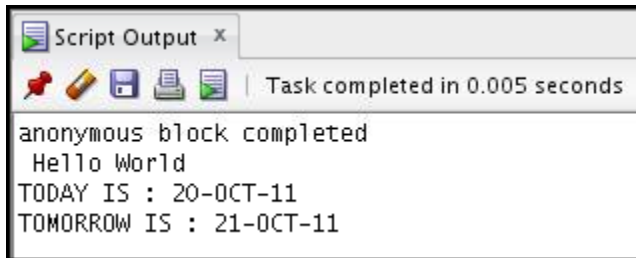


- c. Speichern Sie das Skript unter dem Namen `lab_10_01_soln.sql`.
- d. Um den Workspace zu leeren, klicken Sie auf die Schaltfläche **Clear**.
- e. Um die Prozedur `greet` aufzurufen, erstellen Sie einen anonymen Block und führen ihn aus. (**Tip**: Achten Sie darauf, dass am Blockanfang `SERVEROUTPUT` aktiviert wird.)

```
SET SERVEROUTPUT ON

BEGIN
    greet;
END;
```

Die Ausgabe sollte wie folgt aussehen:



```
Script Output x
Task completed in 0.005 seconds
anonymous block completed
Hello World
TODAY IS : 20-OCT-11
TOMORROW IS : 21-OCT-11
```

2. Bearbeiten Sie wie folgt das Skript lab_10_01_soln.sql:

a. Löschen Sie die Prozedur `greet` mit dem folgenden Befehl:

```
DROP PROCEDURE greet;
```

b. Bearbeiten Sie die Prozedur so, dass sie ein Argument vom Typ `VARCHAR2` annimmt. Nennen Sie das Argument `p_name`.

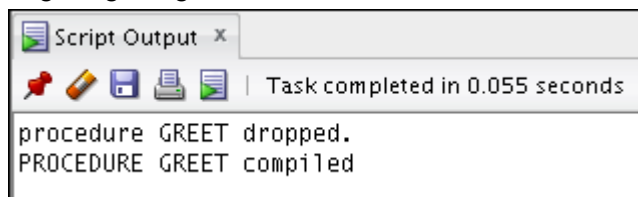
```
CREATE PROCEDURE greet(p_name VARCHAR2) IS
  V_today DATE:=SYSDATE;
  V_tomorrow today%TYPE;
```

c. Geben Sie `Hello <name>` anstelle von `Hello World` aus.

```
BEGIN
  V_tomorrow:=v_today +1;
  DBMS_OUTPUT.PUT_LINE(' Hello ' || p_name);
...
```

d. Speichern Sie Ihr Skript unter dem Namen lab_10_02_soln.sql.

e. Führen Sie das Skript aus, um die Prozedur zu erstellen. Die Ausgabe soll in etwa wie folgt angezeigt werden:

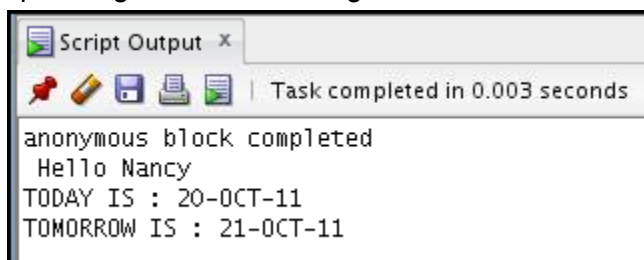


```
Script Output x
Task completed in 0.055 seconds
procedure GREET dropped.
PROCEDURE GREET compiled
```

f. Um die Prozedur `greet` mit einem Parameterwert aufzurufen, erstellen Sie einen anonymen Block und führen ihn aus. Der Block sollte auch die Ausgabe erzeugen.

```
SET SERVEROUTPUT ON;
BEGIN
  greet('Nancy');
END;
```

Die Beispielausgabe sollte wie folgt aussehen:



```
Script Output x
Task completed in 0.003 seconds
anonymous block completed
Hello Nancy
TODAY IS : 20-OCT-11
TOMORROW IS : 21-OCT-11
```


Zusätzliche Übungen und Lösungen zu Lektion 1

Kapitel 11

Übungen zu Lektion 1

Übungsüberblick

Diese Lektion enthält keine Übungen.

Zusätzliche Übungen und Lösungen zu Lektion 2

Kapitel 12

Zusätzliche Übungen zu Lektion 2

Überblick

Die zusätzlichen Übungen sind eine Ergänzung zum Kurs *Oracle Database: PL/SQL Fundamentals*. Sie wenden in den Übungen die im Kurs beschriebenen Konzepte an.

Diese zusätzlichen Übungen vermitteln Ihnen erweiterte praktische Erfahrungen bei der Deklaration von Variablen, der Erstellung von ausführbaren Anweisungen, der Interaktion mit dem Oracle-Server, der Erstellung von Kontrollstrukturen, der Arbeit mit zusammengesetzten Datentypen und Cursoren sowie der Behandlung von Exceptions. In diesem Teil der zusätzlichen Übungen werden unter anderem die Tabellen `employees`, `jobs`, `job_history` und `departments` verwendet.

Übungen zu Lektion 2: Deklarationen beurteilen

Überblick

Diese nicht am Rechner auszuführenden Übungen vermitteln zusätzliche praktische Erfahrungen bei der Deklaration von Variablen und der Erstellung von ausführbaren Anweisungen.

Beurteilen Sie die folgenden Deklarationen. Bestimmen Sie, welche Deklarationen nicht zulässig sind, und erläutern Sie warum.

1. DECLARE
name, dept VARCHAR2 (14) ;
2. DECLARE
test NUMBER (5) ;
3. DECLARE
MAXSALARY NUMBER (7,2) = 5000 ;
4. DECLARE
JOINDATE BOOLEAN := SYSDATE ;

Übungen zu Lektion 2 – Lösung: Deklarationen beurteilen

Beurteilen Sie die folgenden Deklarationen. Bestimmen Sie, welche Deklarationen nicht zulässig sind, und erläutern Sie warum.

1. DECLARE
name, dept VARCHAR2 (14) ;

Unzulässig, da nur eine ID pro Deklaration erlaubt ist

2. DECLARE
test NUMBER (5) ;

Zulässig

3. DECLARE
MAXSALARY NUMBER (7,2) = 5000 ;

Unzulässig, da der Zuweisungsoperator falsch ist. Er muss := lauten.

4. DECLARE
JOINDATE BOOLEAN := SYSDATE ;

Unzulässig, da die Datentypen nicht übereinstimmen. Booleschen Datentypen kann kein Datumswert zugewiesen werden. Als Datentyp muss DATE verwendet werden.

Zusätzliche Übungen und Lösungen zu Lektion 3

Kapitel 13

Übungen zu Lektion 3: Ausdrücke beurteilen

Bestimmen Sie in den folgenden Zuweisungen jeweils den Datentyp des resultierenden Ausdrucks.

1. `email := firstname || to_char(empno);`
2. `confirm := to_date('20-JAN-1999', 'DD-MON-YYYY');`
3. `sal := (1000*12) + 500`
4. `test := FALSE;`
5. `temp := temp1 < (temp2/ 3);`
6. `var := sysdate;`

Übungen zu Lektion 3 – Lösung: Ausdrücke beurteilen

Bestimmen Sie in den folgenden Zuweisungen jeweils den Datentyp des resultierenden Ausdrucks.

1. `email := firstname || to_char(empno);`

Zeichenfolge

2. `confirm := to_date('20-JAN-1999', 'DD-MON-YYYY');`

Date

3. `sal := (1000*12) + 500`

Numerisch

4. `test := FALSE;`

Boolesch

5. `temp := temp1 < (temp2 / 3);`

Boolesch

6. `var := sysdate;`

Datum

Zusätzliche Übungen und Lösungen zu Lektion 4

Kapitel 14

Übungen zu Lektion 4: Ausführbare Anweisungen beurteilen

In dieser nicht am Rechner auszuführenden Übung werten Sie einen PL/SQL-Block aus. Anschließend beantworten Sie die nachfolgenden Fragen, indem Sie Datentyp und Wert der einzelnen Variablen entsprechend den Regeln für Gültigkeitsbereiche bestimmen.

```
DECLARE
    v_custid      NUMBER(4) := 1600;
    v_custname    VARCHAR2(300) := 'Women Sports Club';
    v_new_custid  NUMBER(3) := 500;
BEGIN
    DECLARE
        v_custid      NUMBER(4) := 0;
        v_custname    VARCHAR2(300) := 'Shape up Sports Club';
        v_new_custid  NUMBER(3) := 300;
        v_new_custname VARCHAR2(300) := 'Jansports Club';
    BEGIN
        v_custid := v_new_custid;
        v_custname := v_custname || ' ' || v_new_custname;
1  ----->
    END;
        v_custid := (v_custid *12) / 10;
2  ----->
    END;
```

Untersuchen Sie den vorstehenden PL/SQL-Block, und bestimmen Sie *Wert* und *Datentyp* jeder der folgenden Variablen entsprechend den Regeln für Gültigkeitsbereiche:

1. v_custid an 1. Position:
2. v_custname an 1. Position:
3. v_new_custid an 1. Position:
4. v_new_custname an 1. Position:
5. v_custid an 2. Position:
6. v_custname an 2. Position:

Übungen zu Lektion 4 – Lösung: Ausführbare Anweisungen beurteilen

Untersuchen Sie den folgenden PL/SQL-Block. Beantworten Sie dann die nachstehenden Fragen, indem Sie Datentyp und Wert jeder der folgenden Variablen entsprechend den Regeln für Gültigkeitsbereiche bestimmen.

```
DECLARE
    v_custid    NUMBER(4) := 1600;
    v_custname  VARCHAR2(300) := 'Women Sports Club';
    v_new_custid NUMBER(3) := 500;
BEGIN
    DECLARE
        v_custid    NUMBER(4) := 0;
        v_custname  VARCHAR2(300) := 'Shape up Sports Club';
        v_new_custid NUMBER(3) := 300;
        v_new_custname VARCHAR2(300) := 'Jansports Club';
    BEGIN
        v_custid := v_new_custid;
        v_custname := v_custname || ' ' || v_new_custname;
1  →
        END;
        v_custid := (v_custid *12) / 10;
2  →
        END;
```

Untersuchen Sie den vorstehenden PL/SQL-Block, und bestimmen Sie *Wert* und *Datentyp* jeder der folgenden Variablen entsprechend den Regeln für Gültigkeitsbereiche:

1. v_custid an 1. Position:
500. Der Datentyp ist NUMBER.
2. v_custname an 1. Position:
Shape up Sports Club Jansports Club. Der Datentyp ist VARCHAR2.
3. v_new_custid an 1. Position:
300. Der Datentyp ist NUMBER (oder INTEGER).
4. v_new_custname an 1. Position:
Jansports Club. Der Datentyp ist VARCHAR2.
5. v_custid an 2. Position:
1920. Der Datentyp ist NUMBER.
6. v_custname an 2. Position:
Women Sports Club. Der Datentyp ist VARCHAR2.

Zusätzliche Übungen und Lösungen zu Lektion 5

Kapitel 15

Übung 1 zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken

Bei dieser Übung müssen die Ergebnisse in einer temporären Tabelle gespeichert werden.

| Column Name | NUM_STORE | CHAR_STORE | DATE_STORE |
|--------------|-----------|------------|------------|
| Key Type | | | |
| Nulls/Unique | | | |
| FK Table | | | |
| FK Column | | | |
| Data Type | Number | VARCHAR2 | Date |
| Length | 7,2 | 35 | |

1. Führen Sie das Skript `lab_ap_05.sql` aus, mit dem die folgende Tabelle erstellt wird:
2. Erstellen Sie einen PL/SQL-Block, der Folgendes ausführt:
 - a. Zwei Variablen deklarieren und diesen folgende Werte zuweisen:

| Variable | Datentyp | Inhalt |
|-----------------|---------------|---------------------------------|
| V_MESSAGE | VARCHAR2 (35) | This is my first PL/SQL program |
| V_ DATE_WRITTEN | DATE | <i>Aktuelles Datum</i> |

- b. Werte aus diesen Variablen in den entsprechenden Spalten der Tabelle `TEMP` speichern
3. Prüfen Sie die Ergebnisse, indem Sie die Tabelle `TEMP` abfragen. Die Ausgabeergebnisse sollten wie folgt angezeigt werden:

| | | |
|---------------------------------|--------------------------------|------------|
| Script Output x | | |
| Task completed in 0.037 seconds | | |
| anonymous block completed | | |
| NUM_STORE | CHAR_STORE | DATE_STORE |
| ----- | | |
| | This is my first PLSQL Program | 18-OCT-12 |

Übung 1 zu Lektion 5 – Lösung: SQL-Anweisungen in PL/SQL-Blöcken

Bei dieser Übung müssen die Ergebnisse in einer temporären Tabelle gespeichert werden.

1. Führen Sie das Skript `lab_ap_05.sql` aus, mit dem die folgende Tabelle erstellt wird:

| Column Name | NUM_STORE | CHAR_STORE | DATE_STORE |
|--------------|-----------|------------|------------|
| Key Type | | | |
| Nulls/Unique | | | |
| FK Table | | | |
| FK Column | | | |
| Data Type | Number | VARCHAR2 | Date |
| Length | 7,2 | 35 | |

2. Erstellen Sie einen PL/SQL-Block, der Folgendes ausführt:

- a. Zwei Variablen deklarieren und diesen folgende Werte zuweisen:

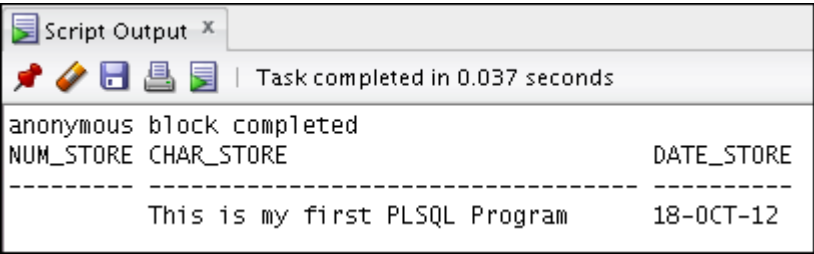
| Variable | Datentyp | Inhalt |
|----------------|--------------|---------------------------------|
| V_MESSAGE | VARCHAR2(35) | This is my first PL/SQL program |
| V_DATE_WRITTEN | DATE | Aktuelles Datum |

- b. Werte aus diesen Variablen in den entsprechenden Spalten der Tabelle `TEMP` speichern

```
DECLARE
    V_MESSAGE VARCHAR2(35);
    V_DATE_WRITTEN DATE;
BEGIN
    V_MESSAGE := 'This is my first PLSQL Program';
    V_DATE_WRITTEN := SYSDATE;
    INSERT INTO temp(CHAR_STORE,DATE_STORE)
        VALUES (V_MESSAGE,V_DATE_WRITTEN);
END;
/
```

3. Prüfen Sie die Ergebnisse, indem Sie die Tabelle `TEMP` abfragen. Die Ausgabeergebnisse sollten in etwa wie folgt aussehen:

```
SELECT * FROM TEMP;
```



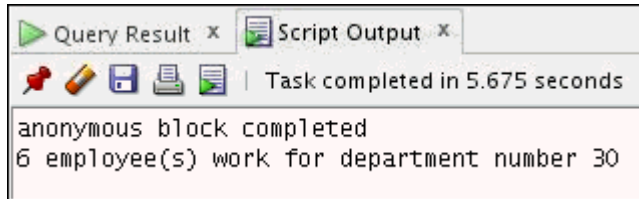
The screenshot shows a 'Script Output' window with a title bar. Below the title bar, there are icons for a red pin, a pencil, a floppy disk, a printer, and a document. To the right of these icons, it says 'Task completed in 0.037 seconds'. The main area of the window displays the output of the SQL query. It starts with 'anonymous block completed'. Below that, it shows the column names 'NUM_STORE', 'CHAR_STORE', and 'DATE_STORE' separated by dashes. The data row shows 'This is my first PLSQL Program' under CHAR_STORE and '18-OCT-12' under DATE_STORE.

| | | |
|-----------|--------------------------------|------------|
| NUM_STORE | CHAR_STORE | DATE_STORE |
| | This is my first PLSQL Program | 18-OCT-12 |

Übung 2 zu Lektion 5: SQL-Anweisungen in PL/SQL-Blöcken

In dieser Übung verwenden Sie Daten aus der Tabelle `employees`.

1. Erstellen Sie einen PL/SQL-Block, um zu bestimmen, wie viele Mitarbeiter für eine bestimmte Abteilung arbeiten. Der PL/SQL-Block soll:
 - eine Substitutionsvariable verwenden, um eine Abteilungsnummer zu speichern
 - die Anzahl der Mitarbeiter einer bestimmten Abteilung ausgeben
2. Bei Ausführung des Blockes wird ein Substitutionsvariablenfenster angezeigt. Geben Sie eine gültige Abteilungsnummer ein, und klicken Sie auf **OK**. Die Ausgabeergebnisse sollten in etwa wie folgt aussehen:



Übung 2 zu Lektion 5 – Lösung: SQL-Anweisungen in PL/SQL-Blöcken

In dieser Übung verwenden Sie Daten aus der Tabelle `employees`.

- Erstellen Sie einen PL/SQL-Block, um zu bestimmen, wie viele Mitarbeiter für eine bestimmte Abteilung arbeiten. Der PL/SQL-Block soll:
 - eine Substitutionsvariable verwenden, um eine Abteilungsnummer zu speichern
 - die Anzahl der Mitarbeiter einer bestimmten Abteilung ausgeben

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    V_HOWMANY NUMBER(3);
```

```
    V_DEPTNO DEPARTMENTS.department_id%TYPE := &P_DEPTNO;
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO V_HOWMANY FROM employees
```

```
    WHERE department_id = V_DEPTNO;
```

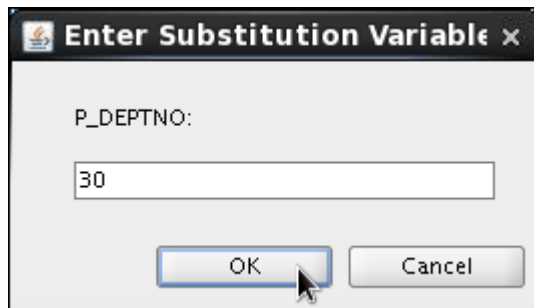
```
    DBMS_OUTPUT.PUT_LINE (V_HOWMANY || ' employee(s)
```

```
        work for department number ' || V_DEPTNO);
```

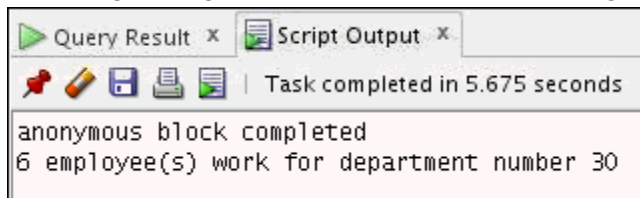
```
END;
```

```
/
```

- Bei Ausführung des Blockes wird ein Substitutionsvariablenfenster angezeigt. Geben Sie eine gültige Abteilungsnummer ein, und klicken Sie auf **OK**.



Die Ausgabeergebnisse sollten in etwa wie folgt aussehen:



Zusätzliche Übungen und Lösungen zu Lektion 6

Kapitel 16

Übung 1 zu Lektion 6: Kontrollstrukturen erstellen

In diesen Übungen steuern Sie die Logik des Programmflusses mithilfe von Kontrollstrukturen.

1. Erstellen Sie einen PL/SQL-Block, der eine Jahreseingabe annimmt, und prüfen Sie, ob es sich um ein Schaltjahr handelt.

Tipp: Schaltjahre sind stets genau durch 4 oder 400 teilbar, nicht aber durch 100.

2. Testen Sie Ihre Lösung anhand der folgenden Tabelle. Wenn Sie beispielsweise 1990 als Jahr eingeben, sollte die Ausgabe "1990 is not a leap year" lauten.

| | |
|------|-----------------|
| 1990 | Not a leap year |
| 2000 | Leap year |
| 1996 | Leap year |
| 1886 | Not a leap year |
| 1992 | Leap year |
| 1824 | Leap year |

Übung 1 zu Lektion 6 – Lösung: Kontrollstrukturen erstellen

1. Erstellen Sie einen PL/SQL-Block, der eine Jahreseingabe annimmt, und prüfen Sie, ob es sich um ein Schaltjahr handelt.

Tipp: Schaltjahre sind stets genau durch 4 oder 400 teilbar, nicht aber durch 100.

```
SET SERVEROUTPUT ON;
DECLARE
    v_YEAR NUMBER(4) := &P_YEAR;
    v_REMAINDER1 NUMBER(5,2);
    v_REMAINDER2 NUMBER(5,2);
    v_REMAINDER3 NUMBER(5,2);
BEGIN
    v_REMAINDER1 := MOD(v_YEAR,4);
    v_REMAINDER2 := MOD(v_YEAR,100);
    v_REMAINDER3 := MOD(v_YEAR,400);
    IF ((v_REMAINDER1 = 0 AND v_REMAINDER2 <> 0 ) OR
        v_REMAINDER3 = 0) THEN
        DBMS_OUTPUT.PUT_LINE(v_YEAR || ' is a leap year');
    ELSE
        DBMS_OUTPUT.PUT_LINE(v_YEAR || ' is not a leap
        year');
    END IF;
END;
/
```

2. Testen Sie Ihre Lösung anhand der folgenden Tabelle. Wenn Sie beispielsweise 1990 als Jahr eingeben, sollte die Ausgabe "1990 is not a leap year" lauten.

| | |
|------|-----------------|
| 1990 | Not a leap year |
| 2000 | Leap year |
| 1996 | Leap year |
| 1886 | Not a leap year |
| 1992 | Leap year |
| 1824 | Leap year |

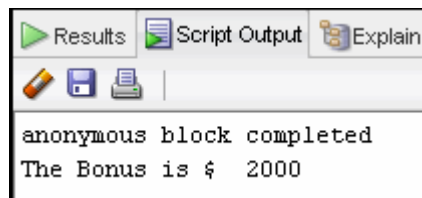
Übung 2 zu Lektion 6: Kontrollstrukturen erstellen

1. Erstellen Sie einen PL/SQL-Block, der das Monatsgehalt eines Mitarbeiters in einer Substitutionsvariablen speichert. Der PL/SQL-Block soll:

- das Jahresgehalt als "Gehalt mal 12" berechnen
- die Prämie wie in der folgenden Tabelle angegeben berechnen:

| Jahresgehalt | Prämie |
|---------------|--------|
| >= 20,000 | 2,000 |
| 19,999–10,000 | 1,000 |
| <= 9,999 | 500 |

- Zeigen Sie im Fenster **Script Output** die Höhe der Prämie im folgenden Format an:



2. Testen Sie den PL/SQL-Block mit folgenden Werten:

| Monatsgehalt | Prämie |
|--------------|--------|
| 3000 | 2000 |
| 1200 | 1000 |
| 800 | 500 |

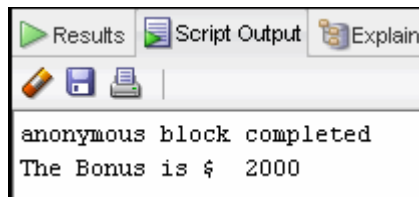
Übung 2 zu Lektion 6 – Lösung: Kontrollstrukturen erstellen

1. Erstellen Sie einen PL/SQL-Block, der das Monatsgehalt eines Mitarbeiters in einer Substitutionsvariablen speichert. Der PL/SQL-Block soll:

- das Jahresgehalt als "Gehalt mal 12" berechnen
- die Prämie wie in der folgenden Tabelle angegeben berechnen:

| Jahresgehalt | Prämie |
|---------------|--------|
| >= 20,000 | 2,000 |
| 19,999–10,000 | 1,000 |
| <= 9,999 | 500 |

- Zeigen Sie im Fenster **Script Output** die Höhe der Prämie im folgenden Format an:



```
SET SERVEROUTPUT ON;
DECLARE
    V_SAL          NUMBER(7,2) := &B_SALARY;
    V_BONUS        NUMBER(7,2);
    V_ANN_SALARY   NUMBER(15,2);
BEGIN
    V_ANN_SALARY := V_SAL * 12;
    IF V_ANN_SALARY >= 20000 THEN
        V_BONUS := 2000;
    ELSIF V_ANN_SALARY <= 19999 AND V_ANN_SALARY >= 10000 THEN
        V_BONUS := 1000;
    ELSE
        V_BONUS := 500;
    END IF;
    DBMS_OUTPUT.PUT_LINE ('The Bonus is $ ' ||
        TO_CHAR(V_BONUS));
END;
/
```

2. Testen Sie den PL/SQL-Block mit folgenden Werten:

| Monatsgehalt | Prämie |
|--------------|--------|
| 3000 | 2000 |
| 1200 | 1000 |
| 800 | 500 |

Zusätzliche Übungen und Lösungen zu Lektion 7: Mit zusammengesetzten Datentypen arbeiten

Kapitel 17

Zusätzliche Übungen zu den Lektionen "Mit zusammengesetzten Datentypen arbeiten" und "Explizite Cursor"

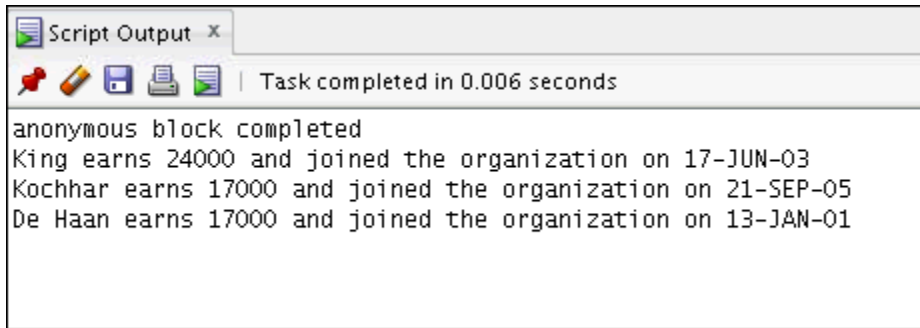
Überblick

In den folgenden Übungen machen Sie sich näher mit assoziativen Arrays und expliziten Cursors vertraut. (Diese Themen werden in den Lektionen "Mit zusammengesetzten Datentypen arbeiten" und "Explizite Cursor" behandelt.) In der ersten Übung definieren Sie einen expliziten Cursor zum Abrufen von Daten und führen ihn aus. In der zweiten Übung kombinieren Sie assoziative Arrays mit einem expliziten Cursor, um Daten auszugeben, die bestimmte Kriterien erfüllen.

Übung 1 zu Lektionen 7/8: Daten mit einem expliziten Cursor abrufen

In dieser Übung erstellen Sie einen PL/SQL-Block, um folgende Schritte auszuführen:

1. Deklarieren Sie den Cursor `EMP_CUR`, um Nachname, Gehalt und Einstellungsdatum des Mitarbeiters in der Tabelle `EMPLOYEES` zu wählen.
2. Verarbeiten Sie alle Zeilen im Cursor. Wenn das Gehalt höher als 15.000 ist und das Einstellungsdatum nach dem 01-FEB-1988 liegt, zeigen Sie Mitarbeiternamen, Gehalt und Einstellungsdatum an. Verwenden Sie dabei das in der Beispielausgabe unten gezeigte Format:



```
Script Output x
Task completed in 0.006 seconds

anonymous block completed
King earns 24000 and joined the organization on 17-JUN-03
Kochhar earns 17000 and joined the organization on 21-SEP-05
De Haan earns 17000 and joined the organization on 13-JAN-01
```

Übung 1 zu Lektionen 7/8 – Lösung: Daten mit einem expliziten Cursor abrufen

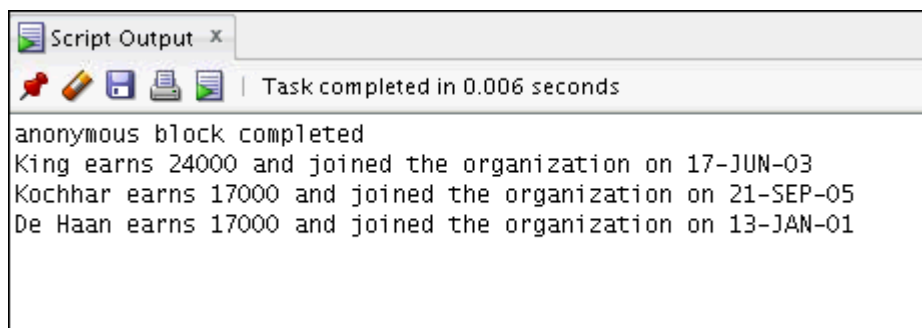
In dieser Übung erstellen Sie einen PL/SQL-Block, um folgende Schritte auszuführen:

1. Deklarieren Sie den Cursor `EMP_CUR`, um Nachname, Gehalt und Einstellungsdatum des Mitarbeiters in der Tabelle `EMPLOYEES` zu wählen.

```
SET SERVEROUTPUT ON;
DECLARE
    CURSOR C_EMP_CUR IS
        SELECT last_name,salary,hire_date FROM EMPLOYEES;
    V_ENAME VARCHAR2(25);
    v_SAL    NUMBER(7,2);
    V_HIREDATE DATE;
```

2. Verarbeiten Sie alle Zeilen im Cursor. Wenn das Gehalt höher als 15.000 ist und das Einstellungsdatum nach dem 01-FEB-1988 liegt, zeigen Sie Mitarbeitername, Gehalt und Einstellungsdatum an. Verwenden Sie dabei das in der Beispielausgabe unten gezeigte Format:

```
BEGIN
    OPEN C_EMP_CUR;
    FETCH C_EMP_CUR INTO V_ENAME,V_SAL,V_HIREDATE;
    WHILE C_EMP_CUR%FOUND
    LOOP
        IF V_SAL > 15000 AND V_HIREDATE >=
            TO_DATE('01-FEB-1988','DD-MON-YYYY') THEN
            DBMS_OUTPUT.PUT_LINE (V_ENAME || ' earns '
                || TO_CHAR(V_SAL) || ' and joined the organization on '
                || TO_DATE(V_HIREDATE,'DD-Mon-YYYY'));
        END IF;
        FETCH C_EMP_CUR INTO V_ENAME,V_SAL,V_HIREDATE;
    END LOOP;
    CLOSE C_EMP_CUR;
END;
/
```

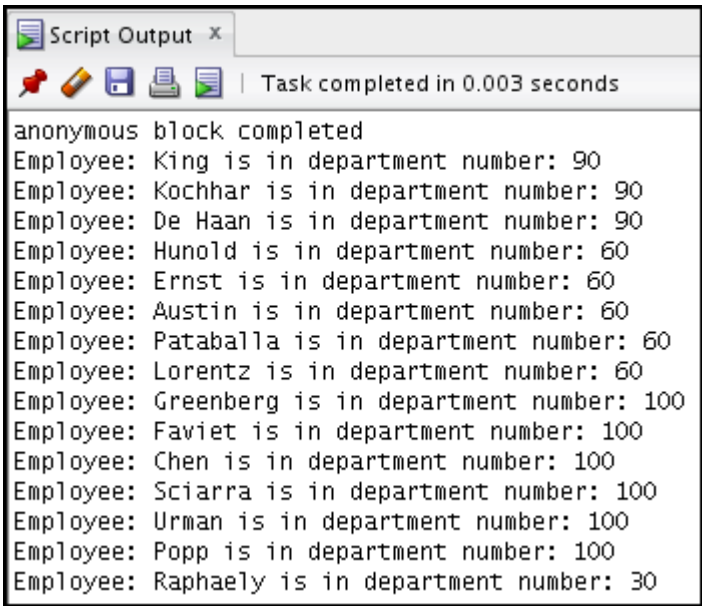


Übung 2 zu Lektionen 7/8: Assoziative Arrays und explizite Cursor

In dieser Übung erstellen Sie einen PL/SQL-Block, um Nachname und Abteilungsnummer aller Mitarbeiter aus der Tabelle `EMPLOYEES` abzurufen und auszugeben, deren `EMPLOYEE_ID` kleiner als 115 ist.

Verwenden Sie im PL/SQL-Block eine Cursor `FOR`-Schleifenstrategie anstelle der Cursormethoden `OPEN/FETCH/CLOSE` aus der vorherigen Übung.

1. Gehen Sie im deklarativen Bereich wie folgt vor:
 - Erstellen Sie zwei assoziative Arrays. Die Unique Key-Spalte für beide Arrays soll den Datentyp `BINARY_INTEGER` aufweisen. Im einen Array ist der Nachname des Mitarbeiters enthalten, im anderen die Abteilungsnummer.
 - Deklarieren Sie einen Cursor, der den Nachnamen und die Abteilungsnummer für Mitarbeiter wählt, deren ID kleiner als 115 ist.
 - Deklarieren Sie eine geeignete Zählervariable, die im ausführbaren Bereich verwendet werden soll.
2. Verwenden Sie im ausführbaren Bereich eine Cursor `FOR`-Schleife (behandelt in der Lektion "Explizite Cursor"), um auf die Cursorwerte zuzugreifen. Weisen Sie diese den entsprechenden assoziativen Arrays zu, und geben Sie die Werte von den Arrays aus. Die richtige Ausgabe sollte 15 Zeilen in folgendem Format zurückgeben:



```
Script Output x
Task completed in 0.003 seconds

anonymous block completed
Employee: King is in department number: 90
Employee: Kochhar is in department number: 90
Employee: De Haan is in department number: 90
Employee: Hunold is in department number: 60
Employee: Ernst is in department number: 60
Employee: Austin is in department number: 60
Employee: Pataballa is in department number: 60
Employee: Lorentz is in department number: 60
Employee: Greenberg is in department number: 100
Employee: Faviet is in department number: 100
Employee: Chen is in department number: 100
Employee: Sciarra is in department number: 100
Employee: Urman is in department number: 100
Employee: Popp is in department number: 100
Employee: Raphaely is in department number: 30
```

Übung 2 zu Lektionen 7/8 – Lösung: Assoziative Arrays und explizite Cursor

In dieser Übung erstellen Sie einen PL/SQL-Block, um Nachname und Abteilungsnummer aller Mitarbeiter aus der Tabelle `EMPLOYEES` abzurufen und auszugeben, deren `EMPLOYEE_ID` kleiner als 115 ist.

Verwenden Sie im PL/SQL-Block eine Cursor `FOR`-Schleifenstrategie anstelle der Cursormethoden `OPEN/FETCH/CLOSE` aus der vorherigen Übung.

1. Gehen Sie im deklarativen Bereich wie folgt vor:

- Erstellen Sie zwei assoziative Arrays. Die Unique Key-Spalte für beide Arrays soll den Datentyp `BINARY_INTEGER` aufweisen. Im einen Array ist der Nachname des Mitarbeiters enthalten, im anderen die Abteilungsnummer.
- Deklarieren Sie eine Zählervariable, die im ausführbaren Bereich verwendet werden soll.
- Deklarieren Sie einen Cursor, der den Nachnamen und die Abteilungsnummer für Mitarbeiter wählt, deren ID kleiner als 115 ist.

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    TYPE Table_Ename IS table of employees.last_name%TYPE
```

```
        INDEX BY BINARY_INTEGER;
```

```
    TYPE Table_dept IS table of employees.department_id%TYPE
```

```
        INDEX BY BINARY_INTEGER;
```

```
    Tename Table_Ename;
```

```
    Tdept Table_dept;
```

```
    i BINARY_INTEGER :=0;
```

```
    CURSOR Namedept IS SELECT last_name,department_id
```

```
    FROM employees WHERE employee_id < 115;
```

2. Verwenden Sie im ausführbaren Bereich eine Cursor `FOR`-Schleife (behandelt in der Lektion "Explizite Cursor"), um auf die Cursorwerte zuzugreifen. Weisen Sie diese den entsprechenden assoziativen Arrays zu, und geben Sie die Werte von den Arrays aus.

```
BEGIN
```

```
    FOR emprec in Namedept
```

```
    LOOP
```

```
        i          := i +1;
```

```
        Tename(i) := emprec.last_name;
```

```
        Tdept(i)  := emprec.department_id;
```

```
        DBMS_OUTPUT.PUT_LINE ('Employee: ' || Tename(i) ||
```

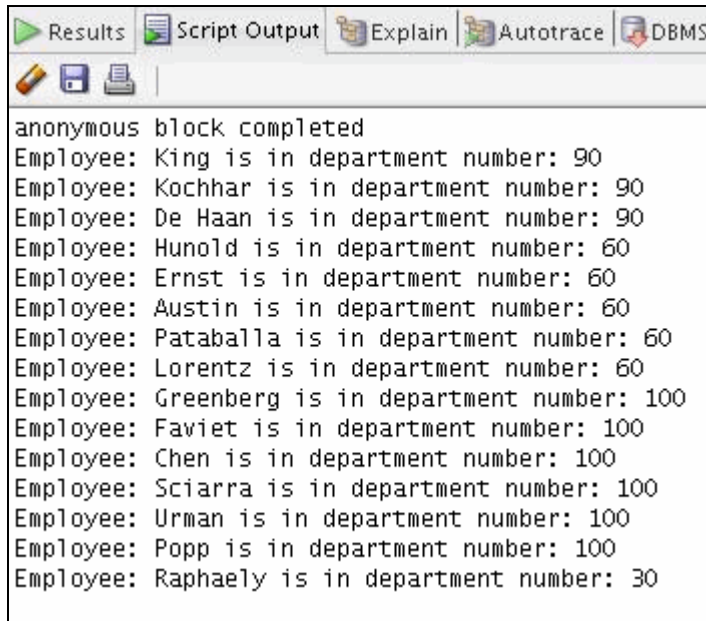
```
' is in department number: ' || Tdept(i));
```

```
    END LOOP;
```

```
END;
```

```
/
```


Die richtige Ausgabe sollte 15 Zeilen in ungefähr folgendem Format zurückgeben:



```
anonymous block completed
Employee: King is in department number: 90
Employee: Kochhar is in department number: 90
Employee: De Haan is in department number: 90
Employee: Hunold is in department number: 60
Employee: Ernst is in department number: 60
Employee: Austin is in department number: 60
Employee: Pataballa is in department number: 60
Employee: Lorentz is in department number: 60
Employee: Greenberg is in department number: 100
Employee: Faviet is in department number: 100
Employee: Chen is in department number: 100
Employee: Sciarra is in department number: 100
Employee: Urman is in department number: 100
Employee: Popp is in department number: 100
Employee: Raphaely is in department number: 30
```


Zusätzliche Übungen und Lösungen zu Lektion 8: Explizite Cursor

Kapitel 18

Übungen zu Lektion 8

Übungsüberblick

Alle Übungen zu dieser Lektion sind in den "Übungen zu Lektion 7" enthalten.

Zusätzliche Übungen und Lösungen zu Lektion 9: Exceptions behandeln

Kapitel 19

Übungen zu Lektion 9: Exceptions behandeln

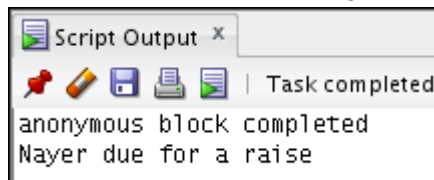
Für diese Übung müssen Sie zuerst eine Tabelle erstellen, in der einige Ergebnisse gespeichert werden. Führen Sie das Skript `lab_ap_09.sql` aus, das die Tabelle für Sie erstellt. Das Skript sieht wie folgt aus:

```
CREATE TABLE analysis
  (ename Varchar2(20), years Number(2), sal Number(8,2)
  );
```

In dieser Übung erstellen Sie einen PL/SQL-Block zur Behandlung von Exceptions. Dabei gehen Sie wie folgt vor:

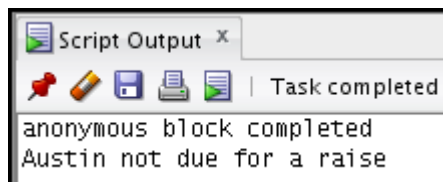
1. Deklarieren Sie Variablen für den Nachnamen, das Gehalt und das Einstellungsdatum des Mitarbeiters. Verwenden Sie eine Substitutionsvariable für den Nachnamen des Mitarbeiters. Fragen Sie dann Nachname (`last_name`), Gehalt (`salary`) und Einstellungsdatum (`hire_date`) des betreffenden Mitarbeiters in der Tabelle `EMPLOYEES` ab.
2. Wenn der Mitarbeiter bereits seit über fünf Jahren im Unternehmen beschäftigt ist und sein Gehalt unter 3.500 liegt, lösen Sie eine Exception aus. Führen Sie im Exception Handler Folgendes aus:

- Geben Sie folgende Informationen aus: Nachname des Mitarbeiters und die Meldung "due for a raise" (siehe folgender Screenshot):



- Fügen Sie den Nachnamen, die Dienstjahre und das Gehalt in die Tabelle `ANALYSIS` ein.

3. Wenn keine Exception ausgelöst wurde, geben Sie den Nachnamen des Mitarbeiters und die Meldung "not due for a raise" aus (in etwa wie im folgenden Screenshot):



4. Prüfen Sie die Ergebnisse, indem Sie die Tabelle `ANALYSIS` abfragen. Testen Sie den PL/SQL-Block mit den folgenden Werten:

| LAST_NAME | MESSAGE |
|-----------|---------------------|
| Austin | Not due for a raise |
| Nayer | Due for a raise |
| Fripp | Not due for a raise |
| Khoo | Due for a raise |

Übungen zu Lektion 9 – Lösung: Exceptions behandeln

Für diese Übung müssen Sie zuerst eine Tabelle erstellen, in der einige Ergebnisse gespeichert werden. Führen Sie das Skript `lab_ap_09.sql` aus, das die Tabelle für Sie erstellt. Das Skript sieht etwa wie folgt aus:

```
CREATE TABLE analysis
    (ename Varchar2(20), years Number(2), sal Number(8,2)
    );
```

In dieser Übung erstellen Sie einen PL/SQL-Block zur Behandlung von Exceptions. Dabei gehen Sie wie folgt vor:

1. Deklarieren Sie Variablen für den Nachnamen, das Gehalt und das Einstellungsdatum des Mitarbeiters. Verwenden Sie eine Substitutionsvariable für den Nachnamen des Mitarbeiters. Fragen Sie dann Nachname (`last_name`), Gehalt (`salary`) und Einstellungsdatum (`hire_date`) des betreffenden Mitarbeiters in der Tabelle `EMPLOYEES` ab.
2. Wenn der Mitarbeiter bereits seit über fünf Jahren im Unternehmen beschäftigt ist und sein Gehalt unter 3.500 liegt, lösen Sie eine Exception aus. Führen Sie im Exception Handler Folgendes aus:
 - Geben Sie folgende Informationen aus: Nachname des Mitarbeiters und die Meldung "due for a raise".
 - Fügen Sie den Namen des Mitarbeiters, die Dienstjahre und das Gehalt in die Tabelle `ANALYSIS` ein.
3. Wenn keine Exception ausgelöst wurde, geben Sie den Nachnamen des Mitarbeiters und die Meldung "not due for a raise" aus:

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    E_DUE_FOR_RAISE EXCEPTION;
```

```
    V_HIREDATE EMPLOYEES.HIRE_DATE%TYPE;
```

```
    V_ENAME EMPLOYEES.LAST_NAME%TYPE := INITCAP( '& B_ENAME' );
```

```
    V_SAL EMPLOYEES.SALARY%TYPE;
```

```
    V_YEARS NUMBER(2);
```

```
BEGIN
```

```
    SELECT SALARY, HIRE_DATE, MONTHS_BETWEEN(SYSDATE, hire_date) / 12
    YEARS
```

```
    INTO V_SAL, V_HIREDATE, V_YEARS
```

```
    FROM employees WHERE last_name = V_ENAME;
```

```
    IF V_SAL < 3500 AND V_YEARS > 5 THEN
```

```
        RAISE E_DUE_FOR_RAISE;
```

```
    ELSE
```

```
        DBMS_OUTPUT.PUT_LINE (V_ENAME || ' not due for a
    raise');
```

```

        END IF;
    EXCEPTION
        WHEN E_DUE_FOR_RAISE THEN
            BEGIN
                DBMS_OUTPUT.PUT_LINE (V_ENAME || ' due for a raise');
                INSERT INTO ANALYSIS(ENAME,YEARS,SAL)
                VALUES (V_ENAME,V_YEARS,V_SAL);
            END;
    END;
/

```

4. Prüfen Sie die Ergebnisse, indem Sie die Tabelle `ANALYSIS` abfragen. Testen Sie den PL/SQL-Block mit den folgenden Werten:

| LAST_NAME | MESSAGE |
|-----------|---------------------|
| Austin | Not due for a raise |
| Nayer | Due for a raise |
| Fripp | Not due for a raise |
| Khoo | Due for a raise |

```
SELECT * FROM analysis;
```

| | ENAME | YEARS | SAL |
|---|-------|-------|------|
| 1 | Nayer | 9 | 3200 |
| 2 | Khoo | 11 | 3100 |