



Hardware and Software
Engineered to Work Together

Oracle Database 12c: SQL Workshop II

Übungen

D80194DE11

Production 1.1 | Dezember 2014 | D88607

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, Bryan Roberts, Laszlo Czinkoczki, KimSeong Loh, Brent Dayley, Jim Spiller, Christopher Wensley, Maheshwari Krishnamurthy, Daniel Milne, Michael Almeida, Diganta Choudhury, Manish Pawar, Clair Bennett, Yanti Chang, Joel Goodman, Gerlinde Frenzen, Madhavi Siddireddy

Redaktion

Raj Kumar, Malavika Jinka

Herausgeber

Jobi Varghese, Pavithran Adka

Dieses Buch wurde erstellt mit: Oracle Tutor

Inhaltsverzeichnis

Übungen zu Lektion 1 – Einführung	1-1
Übungen zu Lektion 1 – Überblick	1-2
Übung 1 zu Lektion 1 – SQL Developer	1-3
Übung 1 zu Lektion 1 – Lösung: SQL Developer	1-5
Übungen zu Lektion 2 – Data Dictionary Views - Einführung	2-1
Übungen zu Lektion 2 – Überblick	2-2
Übung 1 zu Lektion 2 – Data Dictionary Views – Einführung	2-3
Übung 1 zu Lektion 2 – Lösung: Data Dictionary Views – Einführung	2-6
Übungen zu Lektion 3 – Sequences, Synonyme und Indizes erstellen	3-1
Übungen zu Lektion 3 – Überblick	3-2
Übung 1 zu Lektion 3 – Sequences, Synonyme und Indizes erstellen	3-3
Übung 1 zu Lektion 3 – Lösung: Sequences, Synonyme und Indizes erstellen	3-5
Übungen zu Lektion 4 – Views erstellen	4-1
Übungen zu Lektion 4 – Überblick	4-2
Übung 1 zu Lektion 4 – Views erstellen	4-3
Übung 1 zu Lektion 4 – Lösung: Views erstellen	4-6
Übungen zu Lektion 5 – Schemaobjekte verwalten	5-1
Übungen zu Lektion 5 – Überblick	5-2
Übung 1 zu Lektion 5 – Schemaobjekte verwalten	5-3
Übung 1 zu Lektion 5 – Lösung: Schemaobjekte verwalten	5-8
Übungen zu Lektion 6 – Daten mithilfe von Unterabfragen abrufen	6-1
Übungen zu Lektion 6 – Überblick	6-2
Übung 1 zu Lektion 6 – Daten mithilfe von Unterabfragen abrufen	6-3
Übung 1 zu Lektion 6 – Lösung: Daten mithilfe von Unterabfragen abrufen	6-8
Übungen zu Lektion 7 – Daten mit Unterabfragen bearbeiten	7-1
Übungen zu Lektion 7 – Überblick	7-2
Übung 1 zu Lektion 7 – Daten mit Unterabfragen bearbeiten	7-3
Übung 1 zu Lektion 7 – Lösung: Daten mit Unterabfragen bearbeiten	7-4
Übungen zu Lektion 8 – Benutzerzugriff steuern	8-1
Übungen zu Lektion 8 – Überblick	8-2
Übung 1 zu Lektion 8 – Benutzerzugriff steuern	8-3
Übung 1 zu Lektion 8 – Lösung: Benutzerzugriff steuern	8-7
Übungen zu Lektion 9 – Daten bearbeiten	9-1
Übungen zu Lektion 9 – Überblick	9-2
Übung 1 zu Lektion 9 – Daten bearbeiten	9-3
Übung 1 zu Lektion 9 – Lösung: Daten bearbeiten	9-8
Übungen zu Lektion 10 – Daten in verschiedenen Zeitzonen verwalten	10-1
Übungen zu Lektion 10 – Überblick	10-2
Übung 1 zu Lektion 10 – Daten in verschiedenen Zeitzonen verwalten	10-3
Übung 1 zu Lektion 10 – Lösung: Daten in verschiedenen Zeitzonen verwalten	10-6
Zusätzliche Übungen und Lösungen	11-1
Zusätzliche Übungen und Lösungen	11-2
Zusätzliche Übungen	11-3
Zusätzliche Übungen – Lösungen	11-9
Zusätzliche Übungen – Fallbeispiel	11-14
Zusätzliche Übungen – Lösungen: Fallbeispiel	11-17

Übungen zu Lektion 1 – Einführung

Kapitel 1

Übungen zu Lektion 1 – Überblick

Übungsüberblick

In dieser Übung erhalten Sie Ihren Benutzeraccount für diesen Kurs. Anschließend starten Sie SQL Developer, erstellen eine neue Datenbankverbindung und navigieren durch die HR-Tabellen. Außerdem legen Sie verschiedene Voreinstellungen für SQL Developer fest und führen SQL-Anweisungen sowie einen anonymen PL/SQL-Block mit dem SQL Worksheet aus.

Übung 1 zu Lektion 1 – SQL Developer

Aufgaben

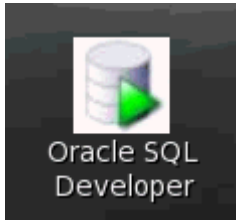
1. Starten Sie SQL Developer über das Desktopsymbol.
2. Erstellen Sie mit folgenden Informationen eine Datenbankverbindung:
 - Connection Name: `myconnection`
 - Username: `ora21`
 - Password: `ora21`
 - Hostname: `localhost`
 - Port: `1521`
 - SID: `orcl` (oder der vom Dozenten genannte Wert)
3. Testen 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 **New/Select Database Connection** auf die Schaltfläche **Test**.
 - b. Wenn als Status **Success** angezeigt wird, klicken Sie auf die Schaltfläche **Connect**.
4. Navigieren Sie durch die Struktur der Tabelle `EMPLOYEES`, und zeigen Sie die Tabellendaten an.
 - a. Blenden Sie die Verbindung `myconnection` ein, indem Sie auf das Pluszeichen klicken.
 - b. Blenden Sie das Symbol **Tables** ein, indem Sie auf das Pluszeichen klicken.
 - c. Zeigen Sie die Struktur der Tabelle `EMPLOYEES` an.
 - d. Zeigen Sie die Daten der Tabelle `DEPARTMENTS` an.
5. Führen Sie einige einfache `SELECT`-Anweisungen aus, um die Daten aus der Tabelle `EMPLOYEES` im SQL Worksheet-Bereich abzufragen. Sie können die `SELECT`-Anweisungen sowohl mit dem Symbol **Execute Statement** (oder mit F9) als auch mit dem Symbol **Run Script** (oder mit F5) ausführen. Prüfen Sie die Ergebnisse beider Ausführungsmethoden für `SELECT`-Anweisungen in den entsprechenden Registerkarten.
 - a. Erstellen Sie eine Abfrage, um Nachname und Gehalt aller Mitarbeiter anzuzeigen, deren Gehalt maximal \$ 3.000 beträgt.
 - b. Erstellen Sie eine Abfrage, um Nachname, Tätigkeits-ID und Provision aller nicht provisionsberechtigten Mitarbeiter anzuzeigen.
6. Legen Sie als Voreinstellung für den Skriptpfad `/home/oracle/labs/sql2` fest.
 - a. Navigieren Sie zu **Tools > Preferences > Database > Worksheet**.
 - b. Geben Sie den Wert im Feld **Select default path to look for scripts** ein.
7. Geben Sie im Feld **Enter SQL Statement** folgenden Code ein:

```
SELECT employee_id, first_name, last_name
      FROM employees;
```

8. Speichern Sie die SQL-Anweisung mit der Menüoption **File > Save** in einer Skriptdatei.
 - a. Klicken Sie auf **File > Save**.
 - b. Nennen Sie die Datei `intro_test.sql`.
 - c. Speichern Sie die Datei im Ordner `/home/oracle/labs/sql2/labs`.
9. Öffnen Sie die Datei `confidence.sql` im Ordner `/home/oracle/labs/sql2/labs`, führen Sie diese aus, und sehen Sie sich die Ausgabe an.

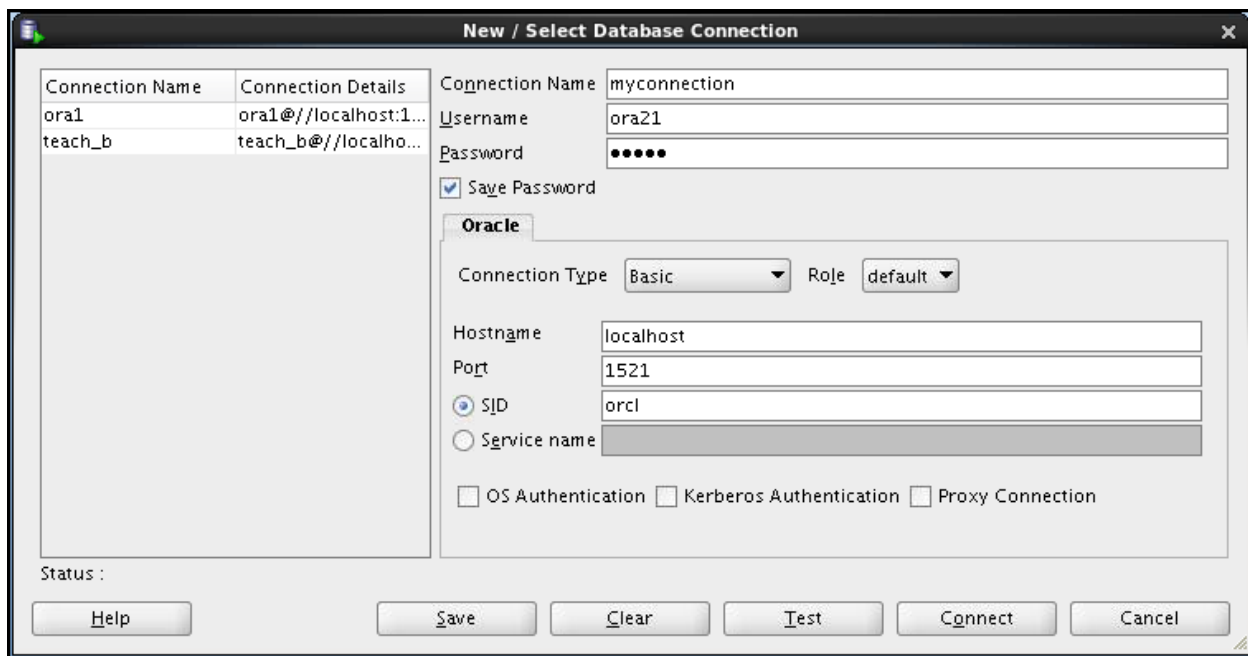
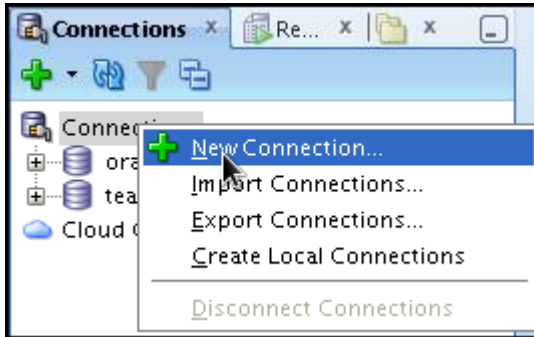
Übung 1 zu Lektion 1 – Lösung: SQL Developer

1. Starten Sie SQL Developer über das Desktopsymbol.



2. Erstellen Sie mit folgenden Informationen eine Datenbankverbindung:

- Connection Name: myconnection
- Username: ora21
- Password: ora21
- Hostname: localhost
- Port: 1521
- SID: orcl (oder der vom Dozenten genannte Wert)



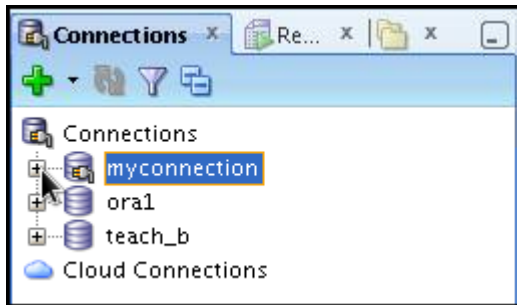
3. Testen Sie die neue Verbindung. Wenn als Status **Success** angezeigt wird, melden Sie sich über diese neue Verbindung bei der Datenbank an.
- a. Klicken Sie im Fenster **New/Select Database Connection** auf die Schaltfläche **Test**.

The screenshot shows the 'New / Select Database Connection' dialog box. On the left, there is a table with two columns: 'Connection Name' and 'Connection Details'. It contains two entries: 'ora1' with details 'ora1@//localhost:1...' and 'teach_b' with details 'teach_b@//localho...'. To the right of this table, there are input fields for 'Connection Name' (myconnection), 'Username' (ora21), and 'Password' (masked with dots). Below these is a checked 'Save Password' checkbox. The 'Oracle' tab is selected, showing 'Connection Type' as 'Basic' and 'Role' as 'default'. Further down are fields for 'Hostname' (localhost), 'Port' (1521), and a radio button selection for 'SID' (selected, value 'orcl') and 'Service name'. At the bottom of this section are three unchecked checkboxes: 'OS Authentication', 'Kerberos Authentication', and 'Proxy Connection'. At the very bottom of the dialog, the status is 'Status : Success'. A row of buttons includes 'Help', 'Save', 'Clear', 'Test' (highlighted with a mouse cursor), 'Connect', and 'Cancel'.

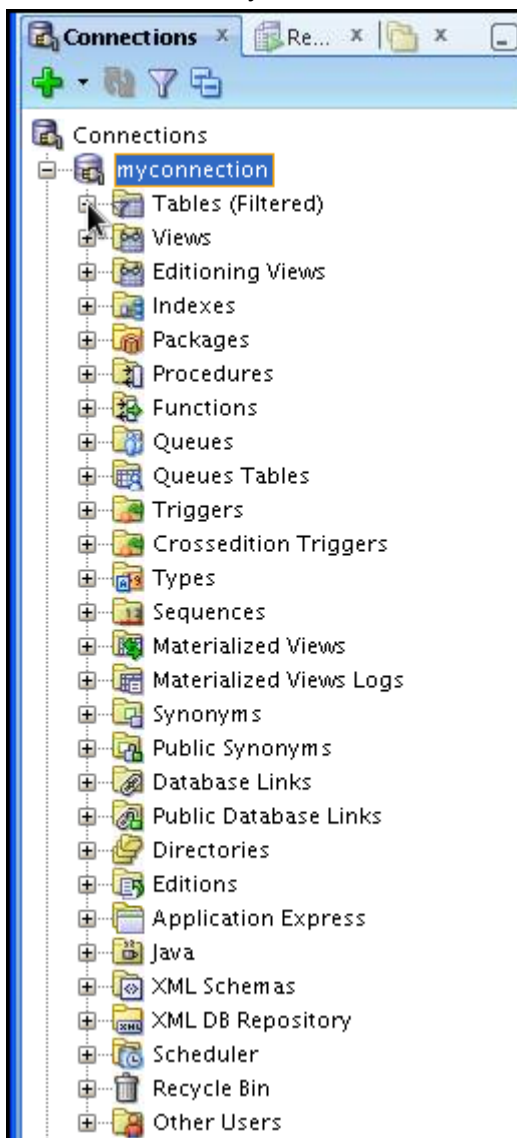
- b. Wenn als Status **Success** angezeigt wird, klicken Sie auf die Schaltfläche **Connect**.

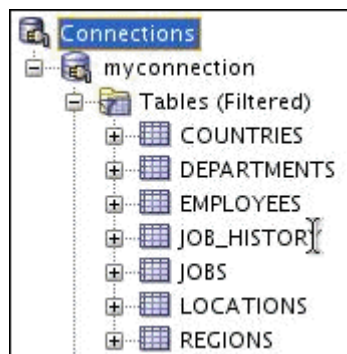
This screenshot is identical to the one above, showing the same 'New / Select Database Connection' dialog box with the same configuration and 'Status : Success' message. However, in this image, the 'Connect' button at the bottom right is highlighted with a mouse cursor instead of the 'Test' button.

4. Navigieren Sie durch die Struktur der Tabelle `EMPLOYEES`, und zeigen Sie die Tabellendaten an.
- a. Blenden Sie die Verbindung `myconnection` ein, indem Sie auf das Pluszeichen klicken.



- b. Blenden Sie das Symbol **Tables** ein, indem Sie auf das Pluszeichen klicken.





c. Zeigen Sie die Struktur der Tabelle `EMPLOYEES` an.

Klicken Sie auf die Tabelle `EMPLOYEES`. In der Registerkarte **Columns** werden die Spalten der Tabelle `EMPLOYEES` wie folgt angezeigt:

Columns							
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COM	
1	EMPLOYEE_ID	NUMBER(6,0)	No	(null)	1	Primary	
2	FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	First	
3	LAST_NAME	VARCHAR2(25 BYTE)	No	(null)	3	Last n	
4	EMAIL	VARCHAR2(25 BYTE)	No	(null)	4	Email	
5	PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)	5	Phone	
6	HIRE_DATE	TIMESTAMP(7)	No	(null)	6	Date w	
7	JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7	Current	
8	SALARY	NUMBER(8,2)	Yes	(null)	8	Monthl	
9	COMMISSION_PCT	NUMBER(2,2)	Yes	(null)	9	Commis	
10	MANAGER_ID	NUMBER(6,0)	Yes	(null)	10	Manage	
11	DEPARTMENT_ID	NUMBER(4,0)	Yes	(null)	11	Depart	

d. Zeigen Sie die Daten der Tabelle `DEPARTMENTS` an.

Klicken Sie im Connections Navigator auf die Tabelle `DEPARTMENTS`. Klicken Sie anschließend auf die Registerkarte **Data**.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700

...

5. Führen Sie einige einfache `SELECT`-Anweisungen aus, um die Daten aus der Tabelle `EMPLOYEES` im SQL Worksheet-Bereich abzufragen. Sie können die `SELECT`-Anweisungen sowohl mit dem Symbol **Execute Statement** (oder mit F9) als auch mit dem Symbol **Run Script** (oder mit F5) ausführen. Prüfen Sie die Ergebnisse beider Ausführungsmethoden für `SELECT`-Anweisungen in den entsprechenden Registerkarten.

- a. Erstellen Sie eine Abfrage, um Nachname und Gehalt aller Mitarbeiter anzuzeigen, deren Gehalt maximal \$ 3.000 beträgt.

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000;
```

	LAST_NAME	SALARY
1	Baida	2900
2	Tobias	2800
3	Himuro	2600
4	Colmenares	2500
5	Mikkilineni	2700
6	Landry	2400
7	Markle	2200
8	Atkinson	2800
9	Marlow	2500
10	Olson	2100
11	Rogers	2900
12	Gee	2400

...

- b. Erstellen Sie eine Abfrage, um Nachname, Tätigkeits-ID und Provision aller nicht provisionsberechtigten Mitarbeiter anzuzeigen.

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

	A Z	LAST_NAME	A Z	JOB_ID	A Z	COMMISSION_PCT
1		King		AD_PRES		(null)
2		Kochhar		AD_VP		(null)
3		De Haan		AD_VP		(null)
4		Hunold		IT_PROG		(null)
5		Ernst		IT_PROG		(null)
6		Austin		IT_PROG		(null)
7		Pataballa		IT_PROG		(null)
8		Lorentz		IT_PROG		(null)
9		Greenberg		FI_MGR		(null)
10		Faviet		FI_ACCOUNT		(null)
11		Chen		FI_ACCOUNT		(null)

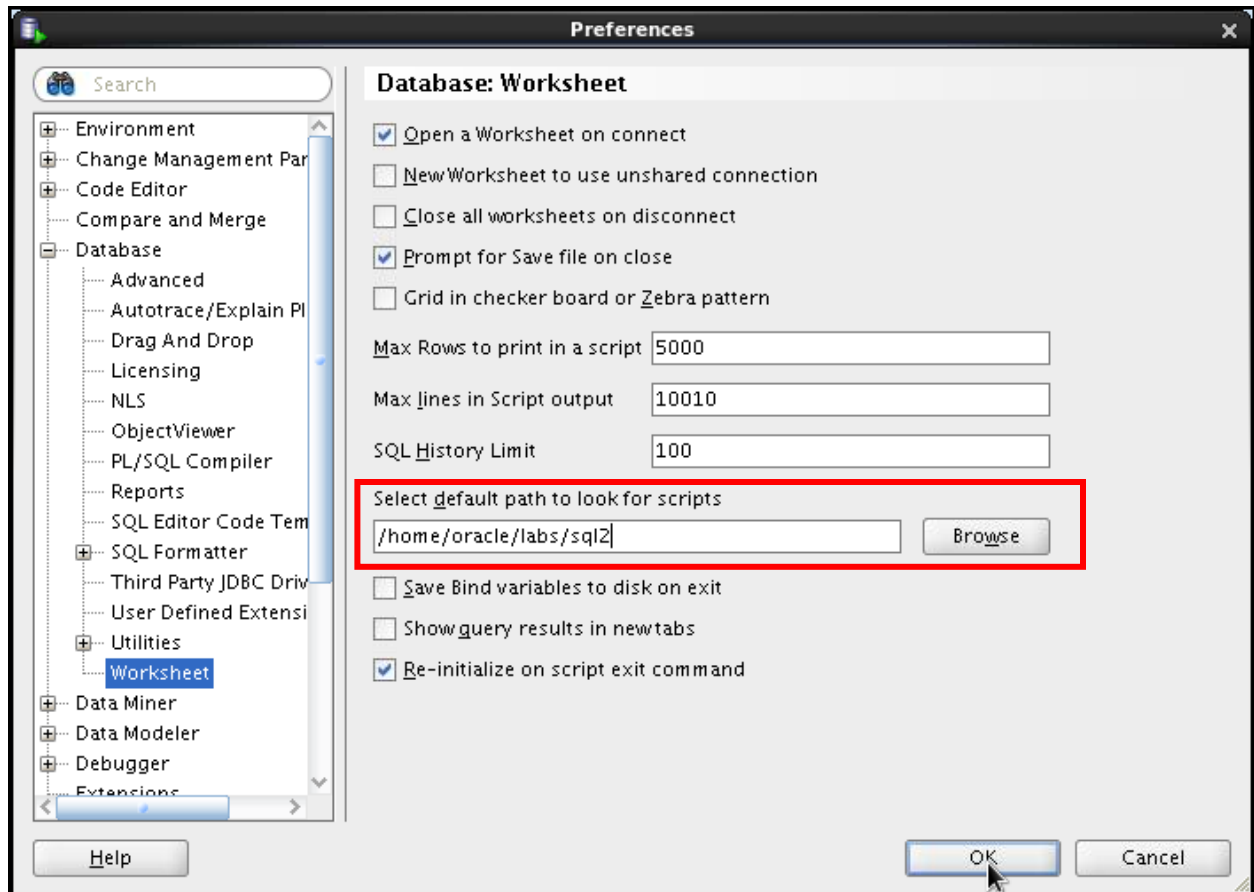
...

6. Legen Sie als Voreinstellung für den Skriptpfad `/home/oracle/labs/sql2` fest.
 - a. Navigieren Sie zu **Tools > Preferences > Database > Worksheet**.
 - b. Geben Sie den Wert im Feld **Select default path to look for scripts** ein. Klicken Sie anschließend auf **OK**.

Hinweis: Um die Anzahl der gewählten Zeilen anzuzeigen, aktivieren Sie die Feedbackoption mit dem Wert "1".

```
set feedback on;
```

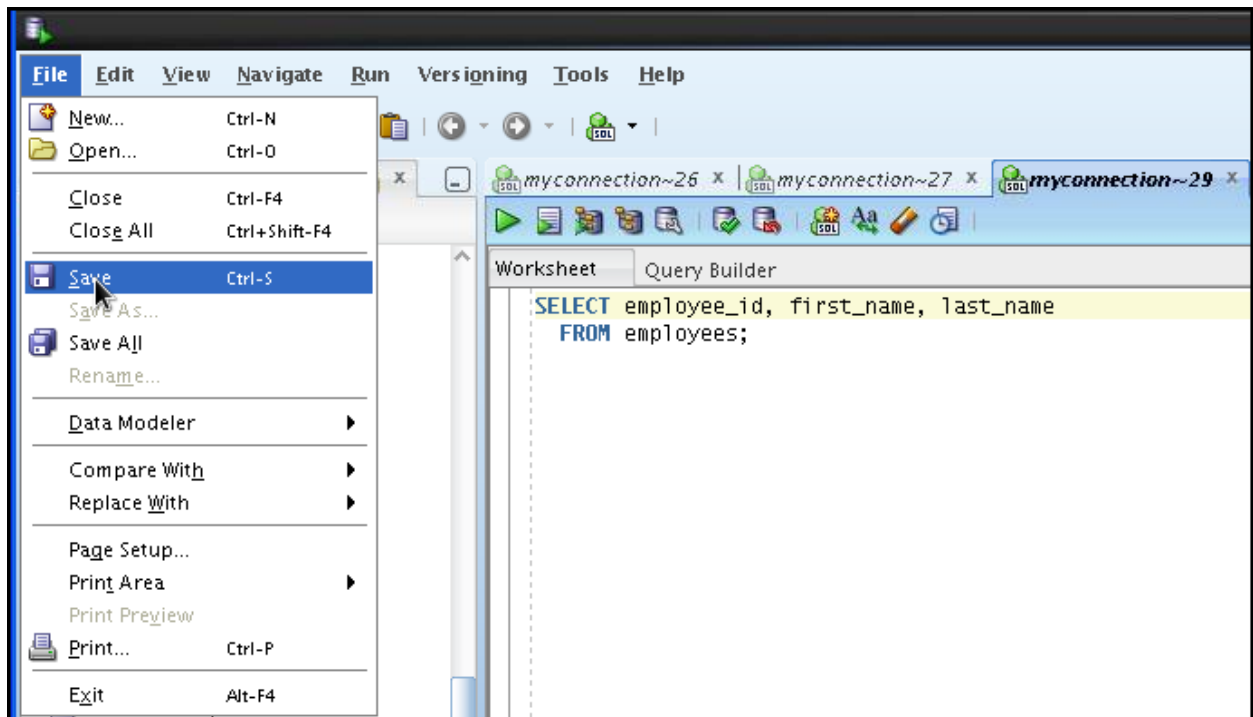
```
set feedback 1;
```



7. Geben Sie folgende SQL-Anweisung ein:

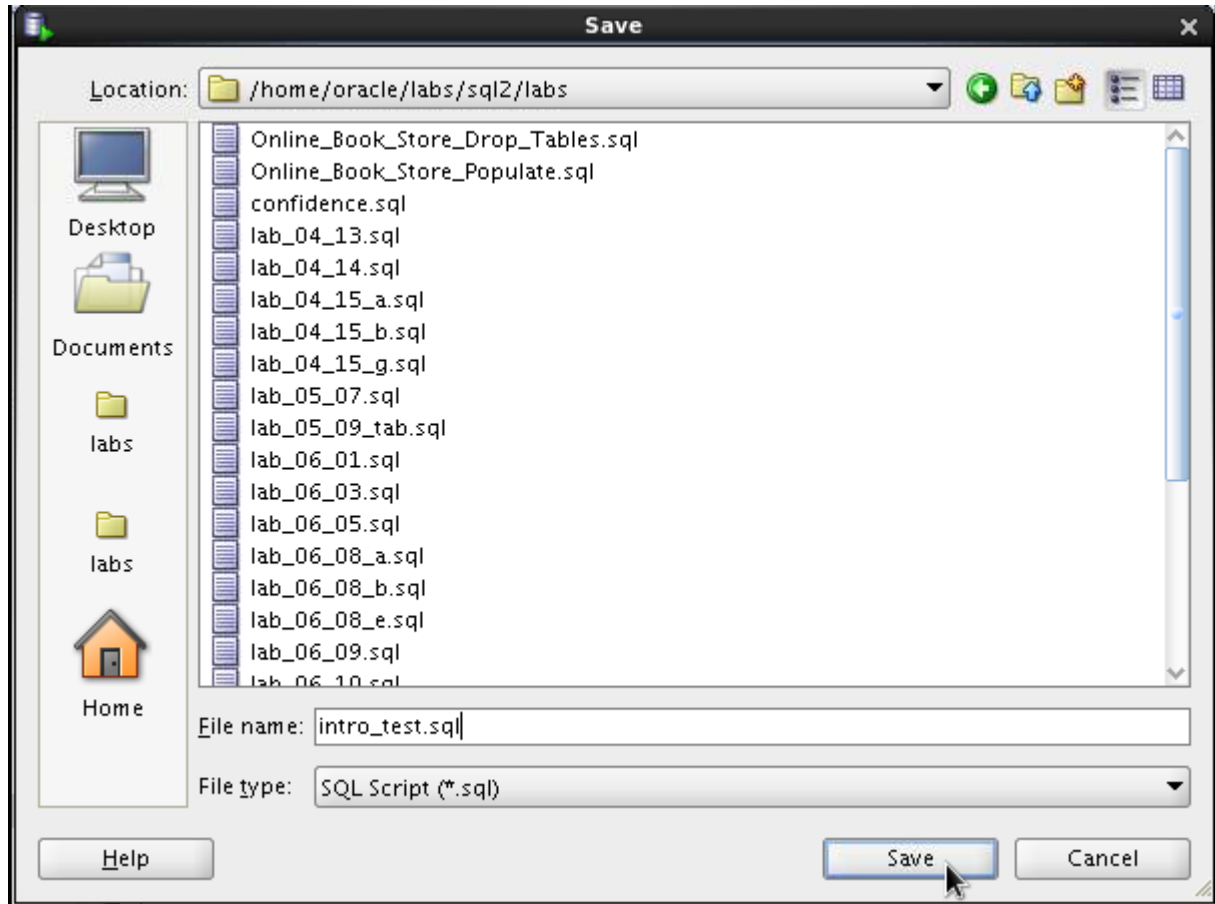
```
SELECT employee_id, first_name, last_name  
FROM employees;
```

8. Speichern Sie die SQL-Anweisung über die Menüoption **File > Save As** in einer Skriptdatei.
- a. Klicken Sie auf **File > Save**.



- b. Nennen Sie die Datei `intro_test.sql`.
Geben Sie im Feld **File name** den Dateinamen `intro_test.sql` ein.

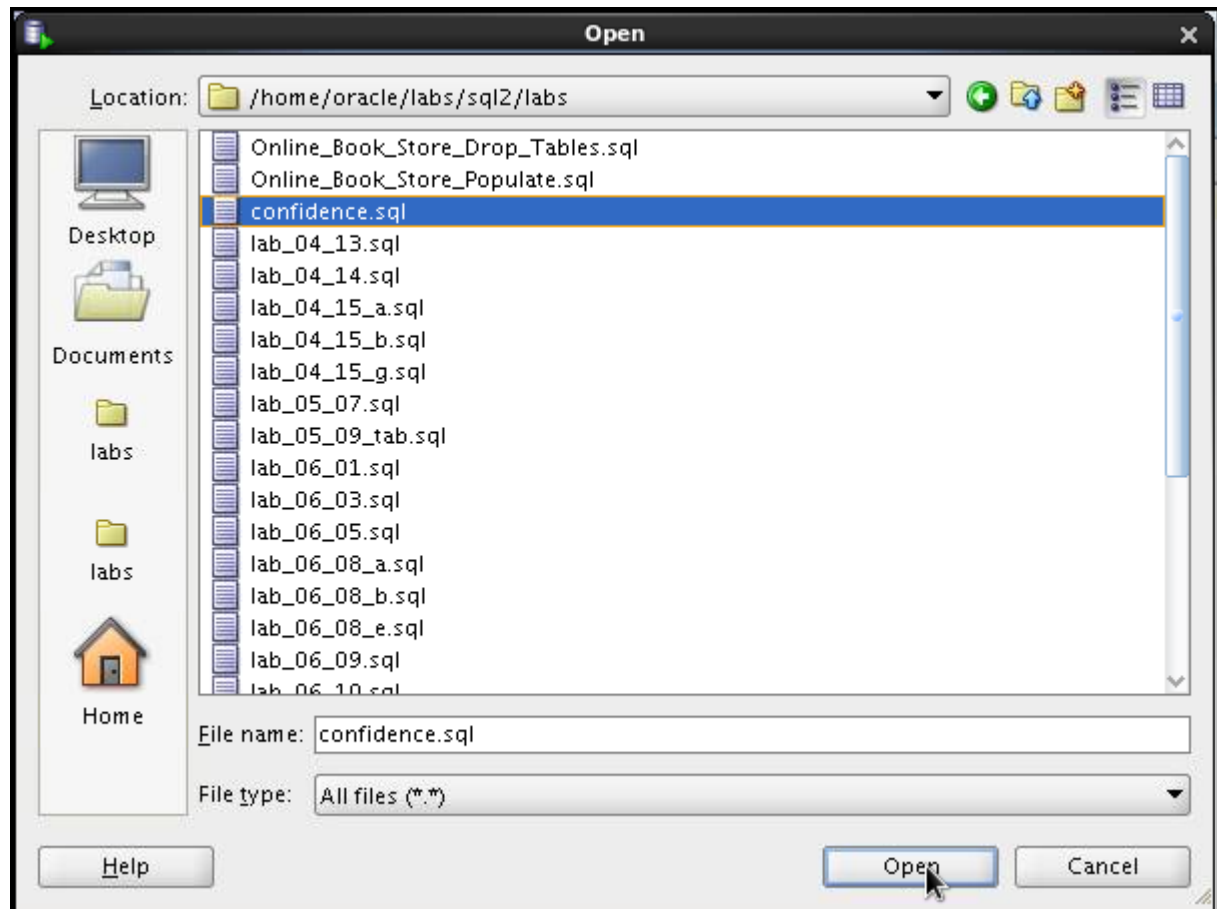
c. Speichern Sie die Datei im Ordner `/home/oracle/labs/sql2/labs`.



Klicken Sie anschließend auf **Save**.

9. Öffnen Sie die Datei `confidence.sql` im Ordner `/home/oracle/labs/sql2/labs`, führen Sie diese aus, und sehen Sie sich die Ausgabe an.

Öffnen Sie die Skriptdatei `confidence.sql` über die Menüoption **File > Open**.



Um das Skript auszuführen, drücken Sie F5.

Das Ergebnis sollte folgendermaßen aussehen:

COUNT(*)

8
COUNT(*)

107
COUNT(*)

25
COUNT(*)

4

COUNT(*)

23
COUNT(*)

27
COUNT(*)

19
COUNT(*)

10

Übungen zu Lektion 2 – Data Dictionary Views – Einführung

Kapitel 2

Übungen zu Lektion 2 – Überblick

Übungsüberblick

Diese Übung behandelt folgende Themen:

- Dictionary Views nach Tabellen- und Spalteninformationen abfragen
- Dictionary Views nach Constraint-Informationen abfragen
- Kommentare zu Tabellen hinzufügen und Dictionary Views nach Kommentarinformationen abfragen

Übung 1 zu Lektion 2 – Data Dictionary Views – Einführung

Überblick

In dieser Übung fragen Sie Dictionary Views ab, um Informationen über Objekte in Ihrem Schema zu erhalten.

Aufgaben

1. Fragen Sie die Data Dictionary View `USER_TABLES` ab, um Informationen über die Tabellen anzuzeigen, deren Eigentümer Sie sind.

	TABLE_NAME
1	REGIONS
2	LOCATIONS
3	DEPARTMENTS
4	JOBS
5	EMPLOYEES

...

2. Fragen Sie die Data Dictionary View `ALL_TABLES` ab, um Informationen über alle Tabellen anzuzeigen, auf die Sie zugreifen können. Schließen Sie Tabellen aus, deren Eigentümer Sie sind.

Hinweis: Ihre Liste weicht möglicherweise von der folgenden Darstellung ab:

	TABLE_NAME	OWNER
1	DUAL	SYS
2	SYSTEM_PRIVILEGE_MAP	SYS
3	TABLE_PRIVILEGE_MAP	SYS
4	USER_PRIVILEGE_MAP	SYS
5	STMT_AUDIT_OPTION_MAP	SYS
6	AUDIT_ACTIONS	SYS
7	WRR\$_REPLAY_CALL_FILTER	SYS
8	HS_BULKLOAD_VIEW_OBJ	SYS
9	HS\$_PARALLEL_METADATA	SYS
10	HS_PARTITION_COL_NAME	SYS
11	HS_PARTITION_COL_TYPE	SYS

...

98	SDO_TOPO_DATA\$	MDSYS
99	SDO_GR_MOSAIC_0	MDSYS
100	SDO_GR_MOSAIC_1	MDSYS
101	SDO_GR_MOSAIC_2	MDSYS
102	SDO_GR_MOSAIC_3	MDSYS
103	SDO_GR_PARALLEL	MDSYS
104	SDO_GR_RDT_1	MDSYS
105	SDO_WFS_LOCAL_TXNS	MDSYS

- Erstellen Sie ein Skript, das für eine gegebene Tabelle die Spaltennamen, die Datentypen und die Länge der Datentypen sowie Angaben darüber ausgibt, ob Nullwerte zulässig sind. Fordern Sie den Benutzer auf, den Tabellennamen einzugeben. Weisen Sie den Spalten `DATA_PRECISION` und `DATA_SCALE` geeignete Aliasnamen zu. Speichern Sie dieses Skript in der Datei `lab_02_03.sql`.

Beispiel: Wenn der Benutzer den Tabellennamen `DEPARTMENTS` eingibt, erhält er folgende Ausgabe:

	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	PRECISION	SCALE	NULLABLE
1	DEPARTMENT_ID	NUMBER	22	4	0	N
2	DEPARTMENT_NAME	VARCHAR2	30	(null)	(null)	N
3	MANAGER_ID	NUMBER	22	6	0	Y
4	LOCATION_ID	NUMBER	22	4	0	Y

- Erstellen Sie ein Skript, das den Spaltennamen, den Constraint-Namen, den Constraint-Typ, das Suchkriterium und den Status für eine gegebene Tabelle ausgibt. Um diese Informationen zu erhalten, müssen Sie die Tabellen `USER_CONSTRAINTS` und `USER_CONS_COLUMNS` verknüpfen. Fordern Sie den Benutzer auf, den Tabellennamen einzugeben. Speichern Sie das Skript in der Datei `lab_02_04.sql`.

Beispiel: Wenn der Benutzer den Tabellennamen `DEPARTMENTS` eingibt, erhält er folgende Ausgabe:

	COLUMN_NAME	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	STATUS
1	MANAGER_ID	DEPT_MGR_FK	R	(null)	ENABLED
2	LOCATION_ID	DEPT_LOC_FK	R	(null)	ENABLED
3	DEPARTMENT_ID	DEPT_ID_PK	P	(null)	ENABLED
4	DEPARTMENT_NAME	DEPT_NAME_NN	C	"DEPARTMENT_NAME" IS NOT NULL	ENABLED

- Fügen Sie einen Kommentar zur Tabelle `DEPARTMENTS` hinzu. Fragen Sie anschließend die View `USER_TAB_COMMENTS` ab, um zu prüfen, ob der Kommentar hinzugefügt wurde.

COMMENTS
1 Company department information including name, code, and location.

- Führen Sie als Voraussetzung für die Übungen 6 bis 9 das Skript `lab_02_06_tab.sql` aus.
Alternativ können Sie die Skriptdatei öffnen, den Code kopieren und in Ihr SQL Worksheet einfügen.
Führen Sie das Skript anschließend aus. Dieses Skript führt folgende Aufgaben aus:
 - Vorhandene Tabellen `DEPT2` und `EMP2` löschen
 - Tabellen `DEPT2` und `EMP2` erstellen

Hinweis: In den Übungen zu Lektion 2 sollten Sie die Tabellen DEPT2 und EMP2 bereits gelöscht haben, sodass sie nicht wiederhergestellt werden können.

7. Prüfen Sie, ob die Tabellen DEPT2 und EMP2 im Data Dictionary gespeichert sind.

	TABLE_NAME
1	DEPT2
2	EMP2

8. Vergewissern Sie sich, dass die Constraints hinzugefügt wurden, indem Sie die View USER_CONSTRAINTS abfragen. Beachten Sie die Typen und Namen der Constraints.

	CONSTRAINT_NAME	CONSTRAINT_TYPE
1	MY_EMP_DEPT_ID_FK	R
2	MY_DEPT_ID_PK	P
3	MY_EMP_ID_PK	P

9. Zeigen Sie in der Data Dictionary View USER_OBJECTS die Objektnamen und -typen für die Tabellen EMP2 und DEPT2 an.

	OBJECT_NAME	OBJECT_TYPE
1	DEPT2	TABLE
2	EMP2	TABLE

Übung 1 zu Lektion 2 – Lösung: Data Dictionary Views – Einführung

Lösung

1. Fragen Sie das Data Dictionary ab, um Informationen über die Tabellen anzuzeigen, deren Eigentümer Sie sind.

```
SELECT table_name
FROM   user_tables;
```

2. Fragen Sie die Data Dictionary View ab, um Informationen über alle Tabellen anzuzeigen, auf die Sie zugreifen können. Schließen Sie Tabellen aus, deren Eigentümer Sie sind.

```
SELECT table_name, owner
FROM   all_tables
WHERE  owner <> 'ORAXX';
```

3. Erstellen Sie ein Skript, das für eine gegebene Tabelle die Spaltennamen, die Datentypen und die Länge der Datentypen sowie Angaben darüber ausgibt, ob Nullwerte zulässig sind. Fordern Sie den Benutzer auf, den Tabellennamen einzugeben. Weisen Sie den Spalten DATA_PRECISION und DATA_SCALE geeignete Aliasnamen zu. Speichern Sie dieses Skript in der Datei lab_02_03.sql.

```
SELECT column_name, data_type, data_length,
       data_precision PRECISION, data_scale SCALE, nullable
FROM   user_tab_columns
WHERE  table_name = UPPER('&tab_name');
```

Um das Skript zu testen, führen Sie es aus und geben als Tabellennamen DEPARTMENTS ein.

4. Erstellen Sie ein Skript, das den Spaltennamen, den Constraint-Namen, den Constraint-Typ, das Suchkriterium und den Status für eine gegebene Tabelle ausgibt. Um diese Informationen zu erhalten, müssen Sie die Tabellen USER_CONSTRAINTS und USER_CONS_COLUMNS verknüpfen. Fordern Sie den Benutzer auf, den Tabellennamen einzugeben. Speichern Sie das Skript in der Datei lab_02_04.sql.

```
SELECT ucc.column_name, uc.constraint_name, uc.constraint_type,
       uc.search_condition, uc.status
FROM   user_constraints uc JOIN user_cons_columns ucc
ON     uc.table_name = ucc.table_name
AND    uc.constraint_name = ucc.constraint_name
AND    uc.table_name = UPPER('&tab_name');
```

Um das Skript zu testen, führen Sie es aus und geben als Tabellennamen DEPARTMENTS ein.

5. Fügen Sie einen Kommentar zur Tabelle `DEPARTMENTS` hinzu. Fragen Sie anschließend die View `USER_TAB_COMMENTS` ab, um zu prüfen, ob der Kommentar hinzugefügt wurde.

```
COMMENT ON TABLE departments IS
    'Company department information including name, code, and
    location.';

SELECT COMMENTS
FROM    user_tab_comments
WHERE   table_name = 'DEPARTMENTS';
```

6. Führen Sie als Voraussetzung für die Übungen 6 bis 9 das Skript `lab_02_06_tab.sql` aus. Alternativ können Sie die Skriptdatei öffnen, den Code kopieren und in Ihr SQL Worksheet einfügen. Führen Sie das Skript anschließend aus. Dieses Skript führt folgende Aufgaben aus:

- Tabellen `DEPT2` und `EMP2` löschen
- Tabellen `DEPT2` und `EMP2` erstellen

7. Prüfen Sie, ob die Tabellen `DEPT2` und `EMP2` im Data Dictionary gespeichert sind.

```
SELECT    table_name
FROM      user_tables
WHERE     table_name IN ('DEPT2', 'EMP2');
```

8. Fragen Sie das Data Dictionary ab, um die Namen und Typen der Constraints für beide Tabellen zu erhalten.

```
SELECT    constraint_name, constraint_type
FROM      user_constraints
WHERE     table_name IN ('EMP2', 'DEPT2');
```

9. Zeigen Sie in der Data Dictionary View `USER_OBJECTS` die Objektnamen und -typen für die Tabellen `EMP2` und `DEPT2` an.

```
SELECT    object_name, object_type
FROM      user_objects
WHERE     object_name= 'EMP2'
OR        object_name= 'DEPT2';
```


Übungen zu Lektion 3 – Sequences, Synonyme und Indizes erstellen

Kapitel 3

Übungen zu Lektion 3 – Überblick

Übungsüberblick

Diese Übung behandelt folgende Themen:

- Sequences erstellen
- Sequences verwenden
- Dictionary Views nach Sequence-Informationen abfragen
- Synonyme erstellen
- Dictionary Views nach Synonyminformationen abfragen
- Indizes erstellen
- Dictionary Views nach Indexinformationen abfragen

Hinweis: Führen Sie das folgende Skript aus, bevor Sie diese Übung beginnen:
`/home/oracle/sql2/code_ex/code_ex_scripts/clean_up_scripts/cleanup_03.sql`

Übung 1 zu Lektion 3 – Sequences, Synonyme und Indizes erstellen

Überblick

Diese Übung enthält eine Reihe von Aufgaben, in denen Sie eine Sequence, einen Index und ein Synonym erstellen und verwenden.

Hinweis: Führen Sie zunächst das Skript `cleanup_03.sql` unter `/home/oracle/sql2/code ex/code_ex scripts/clean up scripts/` aus.

Aufgaben

1. Erstellen Sie die Tabelle `DEPT` auf Basis des folgenden Tabelleninstanzdiagramms. Prüfen Sie, ob die Tabelle erstellt wurde.

Spaltenname	ID	NAME
Schlüsseltyp	Primary key	
NULL/Unique-Werte		
FS-Tabelle		
FS-Spalte		
Datentyp	NUMBER	VARCHAR2
Länge	7	25

2. Sie müssen eine Sequence für die Primärschlüsselspalte der Tabelle DEPT erstellen. Die Sequence soll bei 200 beginnen, einen Höchstwert von 1.000 haben und jeweils um 10 erhöht werden. Geben Sie der Sequence den Namen DEPT_ID_SEQ.
3. Um die Sequence zu testen, schreiben Sie ein Skript, das zwei Zeilen in die Tabelle DEPT einfügt. Nennen Sie das Skript lab_03_03.sql. Verwenden Sie dabei die Sequence, die Sie für die Spalte ID erstellt haben. Fügen Sie zwei Abteilungen hinzu: Education und Administration. Vergewissern Sie sich, dass die Abteilungen hinzugefügt wurden. Führen Sie die Befehle im Skript aus.
4. Ermitteln Sie die Namen Ihrer Sequences. Erstellen Sie eine Abfrage in einem Skript, um folgende Informationen über Ihre Sequences anzuzeigen: Sequence-Name, Höchstwert, Inkrementgröße und letzte Nummer. Nennen Sie das Skript lab_03_04.sql. Führen Sie die Anweisung im Skript aus.

[illegible]

5. Erstellen Sie das Synonym `EMP1` für die Tabelle `EMPLOYEES`. Ermitteln Sie anschließend die Namen aller Synonyme in Ihrem Schema.

	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
1	EMP1	ORA21	EMPLOYEES	(null)

6. Löschen Sie das Synonym EMP1.
7. Erstellen Sie einen nicht eindeutigen Index für die Spalte NAME in der Tabelle DEPT.
8. Erstellen Sie die Tabelle SALES_DEPT auf Basis des folgenden Tabelleninstanzdiagramm. Geben Sie dem Index für die Primärschlüsselspalte den Namen SALES_PK_IDX. Um den Indexnamen und den Tabellennamen zu ermitteln und festzustellen, ob der Index eindeutig ist, fragen Sie anschließend die Data Dictionary View ab.

Column Name	Team_Id	Location
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	3	30

INDEX_NAME	TABLE_NAME	UNIQUENESS
1 SALES_PK_IDX	SALES_DEPT	NONUNIQUE

9. Löschen Sie die in dieser Übung erstellten Tabellen und Sequences.

Übung 1 zu Lektion 3 – Lösung: Sequences, Synonyme und Indizes erstellen

1. Erstellen Sie die Tabelle `DEPT` auf Basis des folgenden Tabelleninstanzdiagramms. Prüfen Sie, ob die Tabelle erstellt wurde.

Spaltenname	ID	NAME
Schlüsseltyp	Primary key	
NULL/Unique-Werte		
FS-Tabelle		
FS-Spalte		
Datentyp	NUMBER	VARCHAR2
Länge	7	25

```
CREATE TABLE dept
(id    NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
name  VARCHAR2(25));
```

Um zu prüfen, ob die Tabelle erstellt wurde, und ihre Struktur anzuzeigen, setzen Sie folgenden Befehl ab:

```
DESCRIBE dept;
```

2. Sie müssen eine Sequence für die Primärschlüsselspalte der Tabelle `DEPT` erstellen. Die Sequence soll bei 200 beginnen, einen Höchstwert von 1.000 haben und jeweils um 10 erhöht werden. Geben Sie der Sequence den Namen `DEPT_ID_SEQ`.

```
CREATE SEQUENCE dept_id_seq
START WITH 200
INCREMENT BY 10
MAXVALUE 1000;
```

3. Um die Sequence zu testen, schreiben Sie ein Skript, das zwei Zeilen in die Tabelle `DEPT` einfügt. Nennen Sie das Skript `lab_03_03.sql`. Verwenden Sie dabei die Sequence, die Sie für die Spalte `ID` erstellt haben. Fügen Sie zwei Abteilungen hinzu: Education und Administration. Vergewissern Sie sich, dass die Abteilungen hinzugefügt wurden. Führen Sie die Befehle im Skript aus.

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

4. Ermitteln Sie die Namen Ihrer Sequences. Erstellen Sie eine Abfrage in einem Skript, um folgende Informationen über Ihre Sequences anzuzeigen: Sequence-Name, Höchstwert, Inkrementgröße und letzte Nummer. Nennen Sie das Skript lab_03_04.sql. Führen Sie die Anweisung im Skript aus.

```
SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

5. Erstellen Sie das Synonym EMP1 für die Tabelle EMPLOYEES. Ermitteln Sie anschließend die Namen aller Synonyme in Ihrem Schema.

```
CREATE SYNONYM emp1 FOR EMPLOYEES;
SELECT *
FROM user_synonyms;
```

6. Löschen Sie das Synonym EMP1.

```
DROP SYNONYM emp1;
```

7. Erstellen Sie einen nicht eindeutigen Index für die Spalte NAME in der Tabelle DEPT.

```
CREATE INDEX dept_name_idx ON dept (name);
```

8. Erstellen Sie die Tabelle SALES_DEPT auf Basis des folgenden Tabelleninstanzdiagramms. Geben Sie dem Index für die Primärschlüsselspalte den Namen SALES_PK_IDX. Um den Indexnamen und den Tabellennamen zu ermitteln und festzustellen, ob der Index eindeutig ist, fragen Sie anschließend die Data Dictionary View ab.

Column Name	Team_Id	Location
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	3	30

```
CREATE TABLE SALES_DEPT
(team_id NUMBER(3)
PRIMARY KEY USING INDEX
(CREATE INDEX sales_pk_idx ON
SALES_DEPT(team_id)),
location VARCHAR2(30));
SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS
FROM USER_INDEXES
WHERE TABLE_NAME = 'SALES_DEPT';
```

9. Löschen Sie die in dieser Übung erstellten Tabellen und Sequences.

```
DROP TABLE DEPT;
DROP TABLE SALES_DEPT;
DROP SEQUENCE dept_id_seq;
```

Übungen zu Lektion 4 – Views erstellen

Kapitel 4

Übungen zu Lektion 4 – Überblick

Übungsüberblick

Diese Übungen behandeln folgende Themen:

- Einfache View erstellen
- Komplexe View erstellen
- View mit einem CHECK-Constraint erstellen
- Datenänderungen in Views versuchen
- Dictionary Views nach View-Informationen abfragen
- Views entfernen

Übung 1 zu Lektion 4 – Views erstellen

Überblick:

Im Rahmen der Übung zu dieser Lektion werden verschiedene Szenarios zur Erstellung, Verwendung, Abfrage und Entfernung von Data Dictionary Views behandelt.

Aufgaben:

1. Die Mitarbeiter der Personalabteilung möchten einige Daten in der Tabelle `EMPLOYEES` ausblenden. Erstellen Sie eine View mit dem Namen `EMPLOYEES_VU`, die auf den Personalnummern, Mitarbeiternamen und Abteilungsnummern aus der Tabelle `EMPLOYEES` basiert. Die Überschrift für die Namen der Mitarbeiter soll `EMPLOYEE` lauten.
2. Prüfen Sie, ob die View funktioniert. Zeigen Sie den Inhalt der View `EMPLOYEES_VU` an.

	EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
5	104	Ernst	60
6	105	Austin	60
7	106	Pataballa	60
8	107	Lorentz	60
9	108	Greenberg	100
10	109	Faviet	100
11	110	Chen	100
12	111	Sciarra	100
13	112	Urman	100

...

3. Erstellen Sie mithilfe der View `EMPLOYEES_VU` eine Abfrage für die Personalabteilung, in der alle Mitarbeiternamen und Abteilungsnummern angezeigt werden.

	EMPLOYEE	DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
5	Ernst	60
6	Austin	60
7	Pataballa	60
8	Lorentz	60
9	Greenberg	100
10	Faviet	100
11	Chen	100

...

4. Die Abteilung 80 benötigt Zugriff auf ihre Mitarbeiterdaten. Erstellen Sie eine View mit dem Namen DEPT80, die die Mitarbeiternummern, Nachnamen und Abteilungsnummern für alle Mitarbeiter der Abteilung 80 enthält. Die View-Spalten sollen EMPNO, EMPLOYEE und DEPTNO heißen. Sorgen Sie aus Sicherheitsgründen dafür, dass Mitarbeiter über die View keiner anderen Abteilung zugeordnet werden können.
5. Zeigen Sie die Struktur und den Inhalt der View DEPT80 an.

```
DESCRIBE dept80
Name      Null      Type
-----
EMPNO     NOT NULL   NUMBER(6)
EMPLOYEE  NOT NULL   VARCHAR2(25)
DEPTNO                    NUMBER(4)
```

	EMPNO	EMPLOYEE	DEPTNO
1	145	Russell	80
2	146	Partners	80
3	147	Errazuriz	80
4	148	Cambrault	80
5	149	Zlotkey	80
6	150	Tucker	80
7	151	Bernstein	80
8	152	Hall	80
9	153	Olsen	80
10	154	Cambrault	80
11	155	Tuvault	80

...

6. Testen Sie Ihre View. Versuchen Sie, Mitarbeiter Abel der Abteilung 80 zuzuweisen.

```
Error report:
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 - "view WITH CHECK OPTION where-clause violation"
*Cause:
*Action:
```

7. Führen Sie die Datei `lab_04_07.sql` aus, um die View `dept50` für diese Übung zu erstellen.
Sie müssen die Namen und Definitionen aller Views in Ihrem Schema ermitteln.
Erstellen Sie dazu einen Bericht, der folgende View-Informationen abrufen: Name der View und Text aus der Data Dictionary View `USER_VIEWS`.

Hinweis: Die View `EMP_DETAILS_VIEW` wurde als Teil Ihres Schemas erstellt.

Hinweis: Um die vollständige Definition der View anzuzeigen, verwenden Sie in SQL Developer **Run Script** (oder F5). Wenn Sie in SQL Developer **Execute Statement** verwenden (oder F9 drücken), scrollen Sie im Bereich **Results** horizontal. Wenn Sie SQL*Plus verwenden, zeigen Sie mit dem Befehl `SET LONG n` weitere Inhalte einer Spalte vom Typ `LONG` an, wobei `n` der Anzahl von Zeichen der jeweiligen Spalte `LONG` entspricht.

VIEW_NAME	TEXT
1 DEPT50	SELECT employee_id empno, last_name employee, department_id deptno FROM
2 DEPT80	SELECT employee_id empno, last_name employee, department_id deptno FROM
3 EMPLOYEES_VU	SELECT employee_id, last_name employee, department_id FROM employees
4 EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.count

8. Entfernen Sie die in dieser Übung erstellten Views.

Übung 1 zu Lektion 4 – Lösung: Views erstellen

1. Die Mitarbeiter der Personalabteilung möchten einige Daten in der Tabelle `EMPLOYEES` ausblenden. Erstellen Sie eine View mit dem Namen `EMPLOYEES_VU`, die auf den Personalnummern, Mitarbeiternamen und Abteilungsnummern aus der Tabelle `EMPLOYEES` basiert. Die Überschrift für die Namen der Mitarbeiter soll `EMPLOYEE` lauten.

```
CREATE OR REPLACE VIEW employees_vu AS
    SELECT employee_id, last_name employee, department_id
    FROM employees;
```

2. Prüfen Sie, ob die View funktioniert. Zeigen Sie den Inhalt der View `EMPLOYEES_VU` an.

```
SELECT *
FROM employees_vu;
```

3. Erstellen Sie mithilfe der View `EMPLOYEES_VU` eine Abfrage für die Personalabteilung, in der alle Mitarbeiternamen und Abteilungsnummern angezeigt werden.

```
SELECT employee, department_id
FROM employees_vu;
```

4. Die Abteilung 80 benötigt Zugriff auf ihre Mitarbeiterdaten. Erstellen Sie eine View mit dem Namen `DEPT80`, die die Mitarbeiternummern, Nachnamen und Abteilungsnummern für alle Mitarbeiter der Abteilung 80 enthält. Die View-Spalten sollen `EMPNO`, `EMPLOYEE` und `DEPTNO` heißen. Sorgen Sie aus Sicherheitsgründen dafür, dass Mitarbeiter über die View keiner anderen Abteilung zugeordnet werden können.

```
CREATE VIEW dept80 AS
    SELECT employee_id empno, last_name employee,
           department_id deptno
    FROM employees
    WHERE department_id = 80
    WITH CHECK OPTION CONSTRAINT emp_dept_80;
```

5. Zeigen Sie die Struktur und den Inhalt der View `DEPT80` an.

```
DESCRIBE dept80

SELECT *
FROM dept80;
```

6. Testen Sie Ihre View. Versuchen Sie, Mitarbeiter Abel der Abteilung 50 zuzuweisen.

```
UPDATE dept80
SET deptno = 50
WHERE employee = 'Abel';
```

Der Fehler tritt auf, weil die View `DEPT50` mit dem Constraint `WITH CHECK OPTION` erstellt wurde. Damit wird sichergestellt, dass die Spalte `DEPTNO` in der View vor Änderungen geschützt ist.

7. Führen Sie die Datei `lab_04_07.sql` aus, um die View `dept50` für diese Übung zu erstellen. Sie müssen die Namen und Definitionen aller Views in Ihrem Schema festlegen. Erstellen Sie dazu einen Bericht, der folgende View-Informationen abrufen: Name der View und Text aus der Data Dictionary View `USER_VIEWS`.

Hinweis: Die View `EMP_DETAILS_VIEW` wurde als Teil Ihres Schemas erstellt.

Hinweis: Um die vollständige Definition der View anzuzeigen, verwenden Sie in SQL Developer **Run Script** (oder F5). Wenn Sie in SQL Developer **Execute Statement** verwenden (oder F9 drücken), scrollen Sie im Bereich **Results** horizontal. Wenn Sie SQL*Plus verwenden, zeigen Sie mit dem Befehl `SET LONG n` weitere Inhalte einer Spalte vom Typ `LONG` an, wobei `n` der Anzahl von Zeichen der jeweiligen Spalte `LONG` entspricht.

```
SELECT    view_name, text
FROM      user_views;
```

8. Entfernen Sie die in dieser Übung erstellten Views.

```
DROP VIEW employees_vu;
DROP VIEW dept80;
DROP VIEW dept50;
```


Übungen zu Lektion 5 – Schemaobjekte verwalten

Kapitel 5

Übungen zu Lektion 5 – Überblick

Übungsüberblick

Diese Übungen behandeln folgende Themen:

- Constraints hinzufügen und löschen
- Constraints verzögern
- Externe Tabellen erstellen

Hinweis: Bevor Sie diese Übung beginnen, führen Sie das Skript `/home/oracle/labs/sql2/code_ex/ /cleanup_scripts/cleanup_05.sql` aus.

Übung 1 zu Lektion 5 – Schemaobjekte verwalten

Überblick

In dieser Übung fügen Sie Constraints hinzu, löschen und verzögern sie. Sie erstellen externe Tabellen.

Hinweis: Führen Sie das Skript `cleanup_05.sql` unter `/home/oracle/labs/sql2/code_ex/ /cleanup_scripts/` aus, bevor Sie die folgenden Aufgaben bearbeiten.

Aufgaben

1. Erstellen Sie die Tabelle `DEPT2` basierend auf dem folgenden Tabelleninstanzdiagramm. Geben Sie die Syntax in das SQL Worksheet ein. Führen Sie dann die Anweisung zum Erstellen der Tabelle aus. Prüfen Sie, ob die Tabelle erstellt wurde.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

```
DESCRIBE dept2
Name Null Type
-----
ID      NUMBER(7)
NAME    VARCHAR2(25)
```

2. Füllen Sie die Tabelle DEPT2 mit Daten der Tabelle DEPARTMENTS. Wählen Sie nur die Spalten, die Sie benötigen. Prüfen Sie, ob die Zeilen erfolgreich eingefügt wurden.

	ID	NAME
1	10	Administration
2	20	Marketing
3	30	Purchasing
4	40	Human Resources
5	50	Shipping
6	60	IT
7	70	Public Relations
8	80	Sales
9	90	Executive
10	100	Finance
11	110	Accounting
12	120	Treasury
13	130	Corporate Tax
14	140	Control And Credit
15	150	Shareholder Services

...

3. Erstellen Sie die Tabelle EMP2 auf der Basis des folgenden Tabelleninstanzdiagramms. Geben Sie die Syntax in das SQL Worksheet ein. Führen Sie dann die Anweisung zum Erstellen der Tabelle aus. Prüfen Sie, ob die Tabelle erstellt wurde.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

```

DESCRIBE emp2
Name      Null Type
-----
ID         NUMBER(7)
LAST_NAME  VARCHAR2(25)
FIRST_NAME VARCHAR2(25)
DEPT_ID    NUMBER(7)

```

4. Fügen Sie der Tabelle EMP2 ein PRIMARY KEY-Constraint auf Tabellenebene hinzu, und verwenden Sie hierzu die Spalte ID. Benennen Sie das Constraint bei der Erstellung. Geben Sie dem Constraint den Namen my_emp_id_pk.

5. Erstellen Sie ein PRIMARY KEY-Constraint für die Tabelle DEPT2, und verwenden Sie hierzu die Spalte ID. Benennen Sie das Constraint bei der Erstellung. Geben Sie dem Constraint den Namen my_dept_id_pk.
6. Fügen Sie in der Tabelle EMP2 einen Fremdschlüsselverweis hinzu, der verhindert, dass der Mitarbeiter einer nicht vorhandenen Abteilung zugewiesen wird. Nennen Sie das Constraint my_emp_dept_id_fk.
7. Ändern Sie die Tabelle EMP2. Fügen Sie die Spalte COMMISSION mit dem Datentyp NUMBER sowie der Gesamtstellenzahl 2 und der Anzahl der Nachkommastellen 2 hinzu. Fügen Sie der Spalte COMMISSION ein Constraint hinzu, das sicherstellt, dass der Provisionswert größer null ist.
8. Löschen Sie die Tabellen EMP2 und DEPT2 so, dass sie nicht wiederhergestellt werden können.
9. Erstellen Sie die externe Tabelle library_items_ext. Verwenden Sie den Zugriffstreiber ORACLE_LOADER.

Hinweis: Das Verzeichnis emp_dir und die Datei library_items.dat wurden bereits für diese Übung erstellt. library_items.dat verfügt über Datensätze im folgenden Format:

```
2354,    2264, 13.21, 150,
2355,    2289, 46.23, 200,
2355,    2264, 50.00, 100,
```

- a. Öffnen Sie die Datei lab_05_09.sql. Sehen Sie sich den Codeauszug zur Erstellung der externen Tabelle library_items_ext an. Ersetzen Sie <TODO1>, <TODO2>, <TODO3> und <TODO4> durch geeigneten Code, und speichern Sie die Datei als lab_05_09_soln.sql. Um die externe Tabelle zu erstellen, führen Sie das Skript aus.
- b. Fragen Sie die Tabelle library_items_ext ab.

	1	2	3	4	5			
	1	CATEGORY_ID	2	BOOK_ID	3	BOOK_PRICE	4	QUANTITY
1		2354		2264		13.21		150
2		2355		2289		46.23		200
3		2355		2264		50		100

10. Die Personalabteilung benötigt einen Bericht mit den Adressen sämtlicher Abteilungen. Erstellen Sie mit dem Zugriffstreiber ORACLE_DATAPUMP eine externe Tabelle mit dem Namen "dept_add_ext". Die Ausgabe des Berichts soll die Standortkennung (LOCATION_ID), die Straße, den Ort, den Bundesstaat oder die Provinz und das Land enthalten. Verwenden Sie einen NATURAL JOIN, um diese Ausgabe zu erzeugen.

Hinweis: Das Verzeichnis emp_dir wurde für diese Übung bereits erstellt.

- a. Öffnen Sie die Datei lab_05_10.sql. Sehen Sie sich den Codeauszug zur Erstellung der externen Tabelle dept_add_ext an. Ersetzen Sie anschließend <TODO1>, <TODO2> und <TODO3> durch geeigneten Code. Ersetzen Sie <oraxx_emp4.exp> und <oraxx_emp5.exp> durch geeignete Dateinamen. Beispiel: Als Benutzer ora21 lauten Ihre Dateinamen ora21_emp4.exp und ora21_emp5.exp. Speichern Sie das Skript als lab_05_10_soln.sql.
- b. Um die externe Tabelle zu erstellen, führen Sie das Skript lab_05_10_soln.sql aus.

- c. Fragen Sie die Tabelle dept_add_ext ab.

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	10001297	Via Cola di Rie	Roma	(null)	Italy
2	110093091	Calle della Testa	Venice	(null)	Italy
3	12002017	Shinjuku-ku	Tokyo	Tokyo Prefecture	Japan
4	13009450	Kamiya-cho	Hiroshima	(null)	Japan
5	14002014	Jabberwocky Rd	Southlake	Texas	United States of America
6	15002011	Interiors Blvd	South San Francisco	California	United States of America
7	16002007	Zagora St	South Brunswick	New Jersey	United States of America
8	17002004	Charade Rd	Seattle	Washington	United States of America
9	1800147	Spadina Ave	Toronto	Ontario	Canada
10	19006092	Boxwood St	Whitehorse	Yukon	Canada

Hinweis: Wenn Sie den vorherigen Schritt ausführen, werden die beiden Dateien oraxx_emp4.exp und oraxx_emp5.exp im Standardverzeichnis emp_dir erstellt.

11. Erstellen Sie die Tabelle emp_books, und füllen Sie sie mit Daten. Legen Sie den Primärschlüssel als verzögert fest, und beobachten Sie, was am Ende der Transaktion geschieht.

- a. Um die Tabelle emp_books zu erstellen, führen Sie die Datei lab_05_11_a.sql aus. Wie Sie sehen, ist der Primärschlüssel emp_books_pk nicht als verzögerbar erstellt.

```
table EMP_BOOKS created.
```

- b. Um die Tabelle emp_books mit Daten zu füllen, führen Sie die Datei lab_05_11_b.sql aus. Was stellen Sie fest?

```
1 rows inserted.

Error starting at line 2 in command:
insert into emp_books values(300,'Change Management')
Error report:
SQL Error: ORA-00001: unique constraint (ORA21.EMP_BOOKS_PK) violated
00001. 00000 - "unique constraint (%s.%s) violated"
*Cause:      An UPDATE or INSERT statement attempted to insert a duplicate key.
              For Trusted Oracle configured in DBMS MAC mode, you may see
              this message if a duplicate entry exists at a different level.
*Action:     Either remove the unique restriction or do not insert the key.
```

- c. Legen Sie das Constraint emp_books_pk als verzögert fest. Was stellen Sie fest?

```
Error starting at line 1 in command:
set constraint emp_books_pk deferred
Error report:
SQL Error: ORA-02447: cannot defer a constraint that is not deferrable
02447. 00000 - "cannot defer a constraint that is not deferrable"
*Cause:      An attempt was made to defer a nondeferrable constraint
*Action:     Drop the constraint and create a new one that is deferrable
```

- d. Löschen Sie das Constraint emp_books_pk.

```
table EMP_BOOKS altered.
```

- e. Um das Constraint emp_books_pk nun als verzögerbares Constraint hinzuzufügen, ändern Sie die Tabellendefinition von emp_books.

```
table EMP_BOOKS altered.
```


- f. Legen Sie das Constraint `emp_books_pk` als verzögert fest.

```
constraint EMP_BOOKS_PK succeeded.
```

- g. Um die Tabelle `emp_books` mit Daten zu füllen, führen Sie die Datei `lab_05_11_g.sql` aus. Was stellen Sie fest?

```
1 rows inserted  
1 rows inserted  
1 rows inserted
```

- h. Schreiben Sie die Transaktion fest. Was stellen Sie fest?

```
Error starting at line 1 in command:  
commit  
Error report:  
SQL Error: ORA-02091: transaction rolled back  
ORA-00001: unique constraint (ORA21.EMP_BOOKS_PK) violated  
02091. 00000 - "transaction rolled back"  
*Cause:      Also see error 2092. If the transaction is aborted at a remote  
              site then you will only see 2091; if aborted at host then you will  
              see 2092 and 2091.  
*Action:     Add rollback segment and retry the transaction.
```

Übung 1 zu Lektion 5 – Lösung: Schemaobjekte verwalten

Lösung

1. Erstellen Sie die Tabelle `DEPT2` basierend auf dem folgenden Tabelleninstanzdiagramm. Geben Sie die Syntax in das SQL Worksheet ein. Führen Sie dann die Anweisung zum Erstellen der Tabelle aus. Prüfen Sie, ob die Tabelle erstellt wurde.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

```
CREATE TABLE dept2
  (id NUMBER(7),
   name VARCHAR2(25));

DESCRIBE dept2
```

2. Füllen Sie die Tabelle `DEPT2` mit Daten der Tabelle `DEPARTMENTS`. Wählen Sie nur die Spalten, die Sie benötigen.

```
INSERT INTO dept2
SELECT department_id, department_name
FROM departments;
```

3. Erstellen Sie die Tabelle `EMP2` basierend auf dem folgenden Tabelleninstanzdiagramm. Geben Sie die Syntax in das SQL Worksheet ein. Führen Sie dann die Anweisung zum Erstellen der Tabelle aus. Prüfen Sie, ob die Tabelle erstellt wurde.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

```
CREATE TABLE emp2
(id          NUMBER(7),
last_name    VARCHAR2(25),
first_name   VARCHAR2(25),
dept_id      NUMBER(7));

DESCRIBE emp2
```

4. Fügen Sie der Tabelle EMP2 ein PRIMARY KEY-Constraint auf Tabellenebene hinzu, und verwenden Sie hierzu die Spalte ID. Benennen Sie das Constraint bei der Erstellung. Geben Sie dem Constraint den Namen my_emp_id_pk.

```
ALTER TABLE emp2
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);
```

5. Erstellen Sie ein PRIMARY KEY-Constraint für die Tabelle DEPT2, und verwenden Sie hierzu die Spalte ID. Benennen Sie das Constraint bei der Erstellung. Geben Sie dem Constraint den Namen my_dept_id_pk.

```
ALTER TABLE dept2
ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(id);
```

6. Fügen Sie in der Tabelle EMP2 einen Fremdschlüsselverweis hinzu, der verhindert, dass der Mitarbeiter einer nicht vorhandenen Abteilung zugewiesen wird. Nennen Sie das Constraint my_emp_dept_id_fk.

```
ALTER TABLE emp2
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES dept2(id);
```

7. Ändern Sie die Tabelle EMP2. Fügen Sie die Spalte COMMISSION mit dem Datentyp NUMBER sowie der Gesamtstellenzahl 2 und der Anzahl der Nachkommastellen 2 hinzu. Fügen Sie der Spalte COMMISSION ein Constraint hinzu, das sicherstellt, dass der Provisionswert größer null ist.

```
ALTER TABLE emp2
ADD commission NUMBER(2,2)
CONSTRAINT my_emp_comm_ck CHECK (commission > 0);
```

8. Löschen Sie die Tabellen EMP2 und DEPT2 so, dass sie nicht wiederhergestellt werden können.

```
DROP TABLE emp2 PURGE;
DROP TABLE dept2 PURGE;
```

9. Erstellen Sie die externe Tabelle `library_items_ext`. Verwenden Sie den Zugriffstreiber `ORACLE_LOADER`.

Hinweis: Die Verzeichnis `emp_dir` und die Datei `library_items.dat` wurden für diese Übung bereits erstellt. Stellen Sie sicher, dass sich die externe Datei und die Datenbank auf demselben System befinden.

`library_items.dat` verfügt über Datensätze im folgenden Format:

```
2354,    2264, 13.21, 150,
2355,    2289, 46.23, 200,
2355,    2264, 50.00, 100,
```

- a. Öffnen Sie die Datei `lab_05_09.sql`. Sehen Sie sich den Codeauszug zur Erstellung der externen Tabelle `library_items_ext` an. Ersetzen Sie **<TODO1>**, **<TODO2>**, **<TODO3>** und **<TODO4>** durch geeigneten Code, und speichern Sie die Datei als `lab_05_09_soln.sql`.

Um die externe Tabelle zu erstellen, führen Sie das Skript aus.

```
CREATE TABLE library_items_ext ( category_id    number(12)
                                , book_id number(6)
                                , book_price number(8,2)
                                , quantity    number(8)
                                )

ORGANIZATION EXTERNAL
  (TYPE ORACLE_LOADER
   DEFAULT DIRECTORY emp_dir
   ACCESS PARAMETERS (RECORDS DELIMITED BY NEWLINE
                     FIELDS TERMINATED BY ',')
   LOCATION ('library_items.dat')
  )
REJECT LIMIT UNLIMITED;
```

- b. Fragen Sie die Tabelle `library_items_ext` ab.

```
SELECT * FROM library_items_ext;
```

10. Die Personalabteilung benötigt einen Bericht mit den Adressen sämtlicher Abteilungen. Erstellen Sie mit dem Zugriffstreiber `ORACLE_DATAPUMP` eine externe Tabelle mit dem Namen `dept_add_ext`. Die Ausgabe des Berichts soll die Standortkennung (`LOCATION_ID`), die Straße, den Ort, den Bundesstaat oder die Provinz und das Land enthalten. Verwenden Sie einen `NATURAL JOIN`, um diese Ausgabe zu erzeugen.

Hinweis: Das Verzeichnis `emp_dir` wurde für diese Übung bereits erstellt. Stellen Sie sicher, dass sich die externe Datei und die Datenbank auf demselben System befinden.

- a. Öffnen Sie die Datei lab_05_10.sql. Sehen Sie sich den Codeauszug zur Erstellung der externen Tabelle dept_add_ext an. Ersetzen Sie anschließend <TODO1>, <TODO2> und <TODO3> durch den geeigneten Code. Ersetzen Sie <oraxx_emp4.exp> und <oraxx_emp5.exp> durch die entsprechenden Dateinamen. Beispiel: Als Benutzer ora21 lauten Ihre Dateinamen ora21_emp4.exp und ora21_emp5.exp. Speichern Sie das Skript als lab_5_10_soln.sql.

```
CREATE TABLE dept_add_ext (location_id,
                           street_address, city,
                           state_province,
                           country_name)

ORGANIZATION EXTERNAL(
TYPE ORACLE_DATAPUMP
DEFAULT DIRECTORY emp_dir
LOCATION ('oraxx_emp4.exp', 'oraxx_emp5.exp'))
PARALLEL
AS
SELECT location_id, street_address, city, state_province,
country_name
FROM locations
NATURAL JOIN countries;
```

Hinweis: Wenn Sie den vorhergehenden Schritt ausführen, werden die beiden Dateien oraxx_emp4.exp und oraxx_emp5.exp im Standardverzeichnis emp_dir erstellt.

- b. Um die externe Tabelle zu erstellen, führen Sie das Skript lab_05_10_soln.sql aus.
c. Fragen Sie die Tabelle dept_add_ext ab.

```
SELECT * FROM dept_add_ext;
```

11. Erstellen Sie die Tabelle emp_books, und füllen Sie sie mit Daten. Legen Sie den Primärschlüssel als verzögert fest, und beobachten Sie, was am Ende der Transaktion geschieht.

- a. Um die Tabelle emp_books zu erstellen, führen Sie das Skript lab_05_11_a.sql aus. Wie Sie sehen, wird der Primärschlüssel emp_books_pk nicht als verzögerbar erstellt.

```
DROP TABLE emp_books CASCADE CONSTRAINTS;
CREATE TABLE emp_books (book_id number,
                        title varchar2(20), CONSTRAINT
emp_books_pk PRIMARY KEY (book_id));
```

- b. Um die Tabelle emp_books mit Daten zu füllen, führen Sie das Skript lab_05_11_b.sql aus.
Was stellen Sie fest?

```
INSERT INTO emp_books VALUES(300,'Organizations');
INSERT INTO emp_books VALUES(300,'Change Management');
```

Die erste Zeile wird eingefügt. Beim Einfügen der zweiten Zeile wird jedoch der Fehler ora-00001 angezeigt.

- c. Legen Sie das Constraint `emp_books_pk` als verzögert fest. Was stellen Sie fest?

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

Sie erhalten die folgende Fehlermeldung: "ORA-02447: Cannot defer a constraint that is not deferrable." (Ein nicht verzögerbares Constraint kann nicht verzögert werden.)

- d. Löschen Sie das Constraint `emp_books_pk`.

```
ALTER TABLE emp_books DROP CONSTRAINT emp_books_pk;
```

- e. Um das Constraint `emp_books_pk` nun als verzögerbares Constraint hinzuzufügen, ändern Sie die Tabellendefinition von `emp_books`.

```
ALTER TABLE emp_books ADD (CONSTRAINT emp_books_pk PRIMARY KEY  
(book_id) DEFERRABLE);
```

- f. Legen Sie das Constraint `emp_books_pk` als verzögert fest.

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

- g. Um die Tabelle `emp_books` mit Daten zu füllen, führen Sie das Skript `lab_05_11_g.sql` aus.
Was stellen Sie fest?

```
INSERT INTO emp_books VALUES (300, 'Change Management');  
INSERT INTO emp_books VALUES (300, 'Personality');  
INSERT INTO emp_books VALUES (350, 'Creativity');
```

Sie sehen, dass alle Zeilen eingefügt werden.

- h. Schreiben Sie die Transaktion fest. Was stellen Sie fest?

```
COMMIT;
```

Wie Sie sehen, wurde die Transaktion von der Datenbank an diesem Punkt zurückgesetzt, da `COMMIT` aufgrund der Constraint-Verletzung nicht ausgeführt werden konnte.

Übungen zu Lektion 6 – Daten mithilfe von Unterabfragen abrufen

Kapitel 6

Übungen zu Lektion 6 – Überblick

Übungsüberblick

Diese Übung behandelt folgende Themen:

- Multiple-Column-Unterabfragen erstellen
- Korrelierte Unterabfragen erstellen
- Operator `EXISTS`
- Skalare Unterabfragen
- Klausel `WITH`

Übung 1 zu Lektion 6 – Daten mithilfe von Unterabfragen abrufen

Überblick

In dieser Übung erstellen Sie Multiple-Column-Unterabfragen sowie korrelierte und skalare Unterabfragen. Außerdem lösen Sie Aufgabenstellungen mithilfe der Klausel `WITH`.

Aufgaben

1. Erstellen Sie eine Abfrage, um Nachname, Abteilungsnummer und Gehalt aller Mitarbeiter anzuzeigen, deren Abteilungsnummer und Gehalt mit der Abteilungsnummer sowie dem Gehalt von Mitarbeitern übereinstimmen, die eine Provision erhalten.

	LAST_NAME	DEPARTMENT_ID	SALARY
1	Russell	80	14000
2	Partners	80	13500
3	Errazuriz	80	12000
4	Abel	80	11000
5	Cambrault	80	11000
6	Vishney	80	10500
7	Zlotkey	80	10500
8	Bloom	80	10000
9	King	80	10000
10	Tucker	80	10000
11	Greene	80	9500

...

2. Zeigen Sie den Nachnamen, den Abteilungsnamen und das Gehalt aller Mitarbeiter an, deren Gehalt und `job_ID` mit dem Gehalt und der `job_ID` der Mitarbeiter übereinstimmen, die am Standort mit der `ID` 1700 tätig sind.

	LAST_NAME	DEPARTMENT_NAME	SALARY
1	Whalen	Administration	4400
2	Colmenares	Purchasing	2500
3	Himuro	Purchasing	2600
4	Tobias	Purchasing	2800
5	Baida	Purchasing	2900
6	Khoo	Purchasing	3100
7	Raphaely	Purchasing	11000
8	Grant	Shipping	2600
9	OConnell	Shipping	2600
10	Walsh	Shipping	3100
11	Jones	Shipping	2800

...

- Erstellen Sie eine Abfrage, um Nachname, Einstellungsdatum und Gehalt für alle Mitarbeiter mit dem gleichen Gehalt und der gleichen `manager_ID` wie Kochhar anzuzeigen.

Hinweis: Kochhar soll nicht in der Ergebnismenge angezeigt werden.

R	LAST_NAME	R	HIRE_DATE	R	SALARY
1	De Haan		13-JAN-01		17000

- Erstellen Sie eine Abfrage, um die Mitarbeiter anzuzeigen, deren Gehalt höher als das Gehalt aller Sales Manager ist (`JOB_ID = 'SA_MAN'`). Sortieren Sie die Ergebnisse vom höchsten bis zum niedrigsten Gehalt.

R	LAST_NAME	R	JOB_ID	R	SALARY
1	King		AD_PRES		24000
2	De Haan		AD_VP		17000
3	Kochhar		AD_VP		17000

- Zeigen Sie die `employee_ID`, den Nachnamen und die `department_ID` der Mitarbeiter an, die in Städten wohnen, deren Name mit *T* beginnt.

R	EMPLOYEE_ID	R	LAST_NAME	R	DEPARTMENT_ID
1	202		Fay		20
2	201		Hartstein		20

- Erstellen Sie eine Abfrage, um alle Mitarbeiter zu ermitteln, deren Gehalt über dem Durchschnittsgehalt in ihrer Abteilung liegt. Zeigen Sie den Nachnamen, das Gehalt und die `department_ID` der Mitarbeiter sowie das Durchschnittsgehalt für die Abteilung an. Sortieren Sie das Ergebnis nach dem Durchschnittsgehalt, und nehmen Sie eine Rundung auf zwei Dezimalstellen vor. Verwenden Sie, wie in der Beispielausgabe gezeigt, Aliasnamen für die von der Abfrage abgerufenen Spalten.

R	ENAME	R	SALARY	R	DEPTNO	R	DEPT_AVG
1	Fripp		8200		50		3475.56
2	Chung		3800		50		3475.56
3	Kaufling		7900		50		3475.56
4	Mourgos		5800		50		3475.56
5	Bell		4000		50		3475.56
6	Rajs		3500		50		3475.56
7	Everett		3900		50		3475.56
8	Sarchand		4200		50		3475.56
9	Bull		4100		50		3475.56
10	Vollman		6500		50		3475.56
11	Ladwig		3600		50		3475.56
12	Dilly		3600		50		3475.56
13	Weiss		8000		50		3475.56

...

7. Suchen Sie alle Mitarbeiter, die keine Vorgesetzten sind.

a. Führen Sie diese Aufgabe zuerst mit dem Operator `NOT EXISTS` durch.

R	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida
7	Banda
8	Bates
9	Bell
10	Bernstein
11	Bissot
12	Bloom
13	Bull
14	Cabrio
15	Cambrault

...

b. Können Sie diese Aufgabe mit dem Operator `NOT IN` lösen? Wie bzw. warum nicht? Wenn nicht, verwenden Sie eine andere Lösung.

R	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida
7	Banda
8	Bates
9	Bell
10	Bernstein
11	Bissot
12	Bloom

...

8. Erstellen Sie eine Abfrage, um die Nachnamen aller Mitarbeiter anzuzeigen, deren Gehalt niedriger als das Durchschnittsgehalt in ihrer Abteilung ist.

R	LAST_NAME
1	Chen
2	Sciarra
3	Urman
4	Popp
5	Khoo
6	Baida
7	Tobias
8	Himuro
9	Colmenares
10	Kochhar
11	De Haan
12	Fay
13	Gietz
14	Nayer

...

9. Erstellen Sie eine Abfrage, um die Nachnamen aller Mitarbeiter anzuzeigen, in deren Abteilung es einen oder mehrere Kollegen gibt, die später eingestellt wurden, jedoch ein höheres Gehalt beziehen.

R	LAST_NAME
1	De Haan
2	Austin
3	Lorentz
4	Pataballa
5	Faviet
6	Sciarra
7	Tobias
8	Bell
9	Sarchand
10	Rajs
11	Ladwig
12	Mallin
13	Kaufling

...

10. Erstellen Sie eine Abfrage, um die `employee_ID`, den Nachnamen und den Abteilungsnamen für alle Mitarbeiter anzuzeigen.

Hinweis: Verwenden Sie eine skalare Unterabfrage, um den Abteilungsnamen in der `SELECT`-Anweisung abzurufen.

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT
1	205	Higgins	Accounting
2	206	Gietz	Accounting
3	200	Whalen	Administration
4	100	King	Executive
5	101	Kochhar	Executive
6	102	De Haan	Executive
7	109	Faviet	Finance
8	108	Greenberg	Finance
9	112	Urman	Finance
10	111	Sciarra	Finance
11	110	Chen	Finance
12	113	Popp	Finance
13	203	Mavris	Human Resources
14	107	Lorentz	IT
15	106	Pataballa	IT

...

102	140	Patel	Shipping
103	141	Rajs	Shipping
104	142	Davies	Shipping
105	143	Matos	Shipping
106	181	Fleaur	Shipping
107	178	Grant	(null)

11. Erstellen Sie eine Abfrage, um die Namen der Abteilungen anzuzeigen, deren Gesamtlohnkosten ein Achtel ($1/8$) der Gesamtlohnkosten des Unternehmens übersteigen. Verwenden Sie für diese Abfrage die Klausel `WITH`. Nennen Sie die Abfrage `SUMMARY`.

	DEPARTMENT_NAME	DEPT_TOTAL
1	Sales	304500
2	Shipping	156400

Übung 1 zu Lektion 6 – Lösung: Daten mithilfe von Unterabfragen abrufen

Lösung

1. Erstellen Sie eine Abfrage, um den Nachnamen, die Abteilungsnummer und das Gehalt aller Mitarbeiter anzuzeigen, deren Abteilungsnummer und Gehalt mit der Abteilungsnummer und dem Gehalt der Mitarbeiter übereinstimmen, die eine Provision erhalten.

```
SELECT last_name, department_id, salary
FROM   employees
WHERE  (salary, department_id) IN
        (SELECT salary, department_id
         FROM   employees
         WHERE  commission_pct IS NOT NULL);
```

2. Zeigen Sie den Nachnamen, den Abteilungsnamen und das Gehalt aller Mitarbeiter an, deren Gehalt und job_ID mit dem Gehalt und der job_ID der Mitarbeiter übereinstimmen, die am Standort mit der ID 1700 tätig sind.

```
SELECT e.last_name, d.department_name, e.salary
FROM   employees e JOIN departments d
ON     e.department_id = d.department_id
AND    (salary, job_id) IN
        (SELECT e.salary, e.job_id
         FROM   employees e JOIN
departments d
         ON     e.department_id = d.department_id
         AND    d.location_id = 1700);
```

3. Erstellen Sie eine Abfrage, um Nachname, Einstellungsdatum und Gehalt für alle Mitarbeiter mit dem gleichen Gehalt und der gleichen manager_ID wie Kochhar anzuzeigen.

Hinweis: Kochhar soll nicht in der Ergebnismenge angezeigt werden.

```
SELECT last_name, hire_date, salary
FROM   employees
WHERE  (salary, manager_id) IN
        (SELECT salary, manager_id
         FROM   employees
         WHERE  last_name = 'Kochhar')
AND    last_name != 'Kochhar';
```

4. Erstellen Sie eine Abfrage, um die Mitarbeiter anzuzeigen, deren Gehalt höher als das Gehalt aller Sales Manager ist (JOB_ID = 'SA_MAN'). Sortieren Sie die Ergebnisse vom höchsten bis zum niedrigsten Gehalt.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary > ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'SA_MAN')
ORDER BY salary DESC;
```

5. Zeigen Sie die employee_ID, den Nachnamen und die department_ID der Mitarbeiter an, die in Städten wohnen, deren Name mit T beginnt.

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM departments
                        WHERE location_id IN
                              (SELECT location_id
                               FROM locations
                               WHERE city LIKE 'T%'));
```

6. Erstellen Sie eine Abfrage, um alle Mitarbeiter zu ermitteln, deren Gehalt über dem Durchschnittsgehalt in ihrer Abteilung liegt. Zeigen Sie den Nachnamen, das Gehalt und die department_ID der Mitarbeiter sowie das Durchschnittsgehalt für die Abteilung an. Sortieren Sie das Ergebnis nach dem Durchschnittsgehalt, und nehmen Sie eine Rundung auf zwei Dezimalstellen vor. Verwenden Sie, wie in der Beispielausgabe gezeigt, Aliasnamen für die von der Abfrage abgerufenen Spalten.

```
SELECT e.last_name ename, e.salary salary,
       e.department_id deptno, ROUND(AVG(a.salary),2)
dept_avg
FROM   employees e, employees a
WHERE  e.department_id = a.department_id
AND    e.salary > (SELECT AVG(salary)
                  FROM   employees
                  WHERE  department_id = e.department_id )
GROUP BY e.last_name, e.salary, e.department_id
ORDER BY AVG(a.salary);
```

7. Suchen Sie alle Mitarbeiter, die keine Vorgesetzten sind.

a. Führen Sie diese Aufgabe zuerst mit dem Operator `NOT EXISTS` durch.

```
SELECT outer.last_name
FROM   employees outer
WHERE  NOT EXISTS (SELECT 'X'
                   FROM employees inner
                   WHERE inner.manager_id =
                        outer.employee_id);
```

b. Können Sie diese Aufgabe mit dem Operator `NOT IN` lösen? Wie bzw. warum nicht?

```
SELECT outer.last_name
FROM   employees outer
WHERE  outer.employee_id
NOT IN (SELECT inner.manager_id
        FROM   employees inner);
```

Diese Alternativlösung ist nicht empfehlenswert. Diese Abfrage ergibt einen `NULL`-Wert, sodass die gesamte Abfrage keine Zeilen zurückgibt. Dies beruht darauf, dass alle Bedingungen, die einen `NULL`-Wert vergleichen, einen `NULL`-Wert ergeben. Wenn es wahrscheinlich ist, dass `NULL`-Werte in der Wertemenge enthalten sind, *sollten* Sie `NOT IN` nicht als Ersatz für `NOT EXISTS` verwenden. Eine wesentlich bessere Lösung wäre beispielsweise die folgende Unterabfrage:

```
SELECT last_name
FROM employees
WHERE employee_id NOT IN (SELECT manager_id
                          FROM employees WHERE manager_id IS NOT
                          NULL);
```

8. Erstellen Sie eine Abfrage, um die Nachnamen aller Mitarbeiter anzuzeigen, deren Gehalt niedriger als das Durchschnittsgehalt in ihrer Abteilung ist.

```
SELECT last_name
FROM   employees outer
WHERE  outer.salary < (SELECT AVG(inner.salary)
                      FROM employees inner
                      WHERE inner.department_id
                          = outer.department_id);
```


9. Erstellen Sie eine Abfrage, um die Nachnamen aller Mitarbeiter anzuzeigen, in deren Abteilung es einen oder mehrere Kollegen gibt, die später eingestellt wurden, jedoch ein höheres Gehalt beziehen.

```
SELECT last_name
FROM employees outer
WHERE EXISTS (SELECT 'X'
              FROM employees inner
              WHERE inner.department_id =
                    outer.department_id
              AND inner.hire_date > outer.hire_date
              AND inner.salary > outer.salary);
```

10. Erstellen Sie eine Abfrage, um die employee_ID, den Nachnamen und den Abteilungs-namen für alle Mitarbeiter anzuzeigen.

Hinweis: Verwenden Sie eine skalare Unterabfrage, um den Abteilungs-namen in der SELECT-Anweisung abzurufen.

```
SELECT employee_id, last_name,
       (SELECT department_name
        FROM departments d
        WHERE e.department_id =
              d.department_id ) department
FROM employees e
ORDER BY department;
```

11. Erstellen Sie eine Abfrage, um die Namen der Abteilungen anzuzeigen, deren Gesamtlohnkosten ein Achtel (1/8) der Gesamtlohnkosten des Unternehmens übersteigen. Verwenden Sie für diese Abfrage die Klausel WITH. Nennen Sie die Abfrage SUMMARY.

```
WITH
summary AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM employees e JOIN departments d
  ON e.department_id = d.department_id
  GROUP BY d.department_name)
SELECT department_name, dept_total
FROM summary
WHERE dept_total > ( SELECT SUM(dept_total) * 1/8
                    FROM summary )
ORDER BY dept_total DESC;
```


Übungen zu Lektion 7 – Daten mit Unterabfragen bearbeiten

Kapitel 7

Übungen zu Lektion 7 – Überblick

Übungsüberblick

Diese Übung behandelt folgende Themen:

- Daten mit Unterabfragen bearbeiten
- Werte einfügen und Unterabfrage als Ziel verwenden
- Schlüsselwort `WITH CHECK OPTION` in DML-Anweisungen
- Zeilen mit korrelierten Unterabfragen aktualisieren und löschen

Übung 1 zu Lektion 7 – Daten mit Unterabfragen bearbeiten

Überblick

In dieser Übung prüfen Sie, was Sie über folgende Bereiche wissen: die Bearbeitung von Daten mit Unterabfragen, die Verwendung des Schlüsselwortes `WITH CHECK OPTION` in DML-Anweisungen und das Aktualisieren und Löschen von Zeilen mit korrelierten Unterabfragen.

Aufgaben

1. Welche der folgenden Aussagen treffen zu?
 - a. Unterabfragen werden zum Abrufen von Daten mithilfe einer Inline View verwendet.
 - b. Mit Unterabfragen können Sie keine Daten aus einer Tabelle in eine andere kopieren.
 - c. Unterabfragen aktualisieren Daten in einer Tabelle auf der Basis von Werten einer anderen Tabelle.
 - d. Unterabfragen löschen Zeilen in einer Tabelle auf der Basis von Zeilen einer anderen Tabelle.
2. Füllen Sie die Leerstellen aus:
 - a. In der Klausel _____ der `INSERT`-Anweisung können Sie eine Unterabfrage statt des Tabellennamens verwenden.Optionen:
 - 1) FROM
 - 2) INTO
 - 3) FOR UPDATE
 - 4) VALUES
3. Das Schlüsselwort `WITH CHECK OPTION` verhindert, dass Sie Zeilen ändern, die nicht in der Unterabfrage vorkommen.
 - a. RICHTIG
 - b. FALSCH
4. Die Liste `SELECT` dieser Unterabfrage muss über dieselbe Anzahl von Spalten wie die Spaltenliste der Klausel `VALUES` verfügen.
 - a. RICHTIG
 - b. FALSCH
5. Sie können mit einer korrelierten Unterabfrage nur die Zeilen löschen, die auch in einer anderen Tabelle vorhanden sind.
 - a. RICHTIG
 - b. FALSCH
6. Um die Konzepte `WITH CHECK OPTION` und korrelierte Unterabfragen kennenzulernen, führen Sie die Demodateien für diese Übung aus.

Übung 1 zu Lektion 7 – Lösung: Daten mit Unterabfragen bearbeiten

1. Welche der folgenden Aussagen treffen zu?
 - a. Unterabfragen werden zum Abrufen von Daten mithilfe einer Inline View verwendet.
 - b. Mit Unterabfragen können Sie keine Daten aus einer Tabelle in eine andere kopieren.
 - c. Unterabfragen aktualisieren Daten in einer Tabelle auf der Basis von Werten einer anderen Tabelle.
 - d. Unterabfragen löschen Zeilen in einer Tabelle auf der Basis von Zeilen einer anderen Tabelle.

Richtige Antworten: a, c und d

2. Füllen Sie die Leerstellen aus:
 - a. In der Klausel _____ der INSERT-Anweisung können Sie eine Unterabfrage statt des Tabellennamens verwenden.

Optionen:

- 1) FROM
- 2) INTO
- 3) FOR UPDATE
- 4) VALUES

Richtige Antwort: 2

3. Das Schlüsselwort WITH CHECK OPTION verhindert, dass Sie Zeilen ändern, die nicht in der Unterabfrage vorkommen.
 - a. RICHTIG
 - b. FALSCH

Richtige Antwort: a

4. Die Liste SELECT dieser Unterabfrage muss über dieselbe Anzahl von Spalten wie die Spaltenliste der Klausel VALUES verfügen.
 - a. RICHTIG
 - b. FALSCH

Richtige Antwort: a

5. Sie können mit einer korrelierten Unterabfrage nur die Zeilen löschen, die auch in einer anderen Tabelle vorhanden sind.
 - a. RICHTIG
 - b. FALSCH

Richtige Antwort: a

6. Um die Konzepte `WITH CHECK OPTION` und korrelierte Unterabfragen kennenzulernen, führen Sie die Demodateien für diese Übung aus.

Übungen zu Lektion 8 – Benutzerzugriff steuern

Kapitel 8

Übungen zu Lektion 8 – Überblick

Übungsüberblick:

Diese Übung behandelt folgende Themen:

- Anderen Benutzern Zugriffsberechtigungen auf Ihre Tabelle erteilen
- Tabellen anderer Benutzer mit den Ihnen zugewiesenen Berechtigungen ändern

Übung 1 zu Lektion 8 – Benutzerzugriff steuern

Überblick

Sie erteilen einem anderen Benutzer die Berechtigung, Ihre Tabelle abzufragen. Sie erfahren, wie Sie den Zugriff auf Datenbankobjekte steuern können.

Aufgaben

1. Welche Berechtigung benötigen Benutzer für die Anmeldung beim Oracle-Server?
Benötigen sie eine Systemberechtigung oder eine Objektberechtigung?

2. Welche Berechtigung benötigen Benutzer für die Erstellung von Tabellen?

3. Wer kann Berechtigungen für eine Tabelle, die Sie erstellt haben, an andere Benutzer weitergeben?

4. Sie sind der DBA. Sie erstellen zahlreiche Benutzer, die dieselben Systemberechtigungen erhalten sollen.
Wie können Sie diese Aufgabe einfacher gestalten?

5. Mit welchem Befehl können Sie Ihr Kennwort ändern?

6. User21 ist der Eigentümer der Tabelle EMP und erteilt User22 die Berechtigung DELETE mit der Klausel WITH GRANT OPTION. Daraufhin erteilt User22 dem Benutzer User23 die Berechtigung DELETE für EMP. User21 stellt nun fest, dass User23 über diese Berechtigung verfügt, und entzieht sie User22. Welcher Benutzer kann nun Löschvorgänge für die Tabelle EMP durchführen?

7. Sie möchten SCOTT die Berechtigung erteilen, Daten in der Tabelle DEPARTMENTS zu aktualisieren. SCOTT soll außerdem die Möglichkeit erhalten, diese Berechtigung an andere Benutzer weiterzugeben. Welchen Befehl verwenden Sie?



Um Aufgabe 8 und die folgenden Aufgaben zu lösen, ist die Anmeldung bei der Datenbank mit SQL Developer erforderlich. Wenn Sie noch nicht angemeldet sind, gehen Sie wie folgt vor:

1. Klicken Sie auf das Desktopsymbol von SQL Developer.
2. Melden Sie sich im Connections Navigator mit dem Account *ora21* und dem entsprechenden von Ihrem Dozenten mitgeteilten Kennwort bei der Datenbank an.
3. Öffnen Sie eine weitere SQL Developer-Session, und melden Sie sich mit *ora22* an.

8. Erteilen Sie einem anderen Benutzer die Berechtigung, Ihre Tabelle abzufragen. Prüfen Sie anschließend, ob dieser Benutzer die Berechtigung verwenden kann.

Hinweis: Öffnen Sie für diese Übung eine weitere SQL Developer-Session, und melden Sie sich mit einem anderen Benutzeraccount an. Wenn Sie beispielsweise gerade den Account `ora21` verwenden, öffnen Sie eine weitere SQL Developer-Session, und melden Sie sich als `ora22` an. Nachfolgend steht "Team 1" für die erste und "Team 2" für die zweite SQL Developer-Session.

- a. Erteilen Sie einem weiteren Benutzer (z. B. `ora22`) die Berechtigung, Datensätze in Ihrer Tabelle `REGIONS` anzuzeigen. Fügen Sie eine Option für diesen Benutzer hinzu, damit er diese Berechtigung an andere Benutzer weitergeben kann.
- b. Lassen Sie den Benutzer die Tabelle `REGIONS` abfragen.

 REGION_ID	 REGION_NAME
1	1 Europe
2	2 Americas
3	3 Asia
4	4 Middle East and Africa

- c. Lassen Sie den Benutzer die Berechtigung an einen dritten Benutzer (z. B. `ora23`) weitergeben.
 - d. Entziehen Sie dem Benutzer, der Schritt b ausführt, die Berechtigung.
9. Erteilen Sie einem anderen Benutzer Berechtigungen, mit denen er in Ihrer Tabelle `COUNTRIES` Daten abfragen und bearbeiten kann. Stellen Sie sicher, dass der Benutzer diese Berechtigungen nicht an andere Benutzer weitergeben kann.
10. Entziehen Sie diesem Benutzer die für die Tabelle `COUNTRIES` erteilten Berechtigungen.
11. Erteilen Sie einem anderen Benutzer Zugriff auf Ihre Tabelle `DEPARTMENTS`. Lassen Sie sich von diesem Benutzer die Berechtigung zur Abfrage seiner Tabelle `DEPARTMENTS` erteilen.

12. Fragen Sie alle Zeilen in der Tabelle `DEPARTMENTS` ab.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
9	90	Executive	100	1700
10	100	Finance	108	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700
14	140	Control And Credit	(null)	1700
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700

...

13. Fügen Sie Ihrer Tabelle `DEPARTMENTS` eine neue Zeile hinzu. Team 1 fügt die Abteilung `Education` mit der Abteilungsnummer 500 hinzu. Team 2 fügt die Abteilung `Human Resources` mit der Abteilungsnummer 510 hinzu. Fragen Sie die Tabelle des anderen Teams ab.

14. Erstellen Sie ein Synonym für die Tabelle `DEPARTMENTS` des anderen Teams.

15. Fragen Sie alle Zeilen in der Tabelle `DEPARTMENTS` des anderen Teams ab, und verwenden Sie dabei Ihr Synonym.

Ergebnisse der Anweisung SELECT von Team 1:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
15	150	Starchordet Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700
28	510	Human Resources	(null)	(null)

Ergebnisse der Anweisung SELECT von Team 2:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
15	150	Starchordet Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700
28	500	Education	(null)	(null)

16. Entziehen Sie dem anderen Team die Berechtigung SELECT.

17. Entfernen Sie die Zeile, die Sie im 13. Schritt in die Tabelle DEPARTMENTS eingefügt haben, und speichern Sie die Änderungen.

18. Löschen Sie die Synonyme "team1" und "team2".

Übung 1 zu Lektion 8 – Lösungen: Benutzerzugriff steuern

1. Welche Berechtigung benötigen Benutzer für die Anmeldung beim Oracle-Server? Benötigen sie eine System- oder ein Objektberechtigung?
Systemberechtigung CREATE SESSION
2. Welche Berechtigung benötigen Benutzer für die Erstellung von Tabellen?
Berechtigung CREATE TABLE
3. Wer kann Berechtigungen für eine Tabelle, die Sie erstellt haben, an andere Benutzer weitergeben?
Sie selbst und alle Benutzer, an die Sie diese Berechtigung mit WITH GRANT OPTION vergeben haben.
4. Sie sind der DBA. Sie erstellen zahlreiche Benutzer, die dieselben Systemberechtigungen erhalten sollen. Wie können Sie vorgehen, um sich diese Aufgabe zu erleichtern?
Sie erstellen eine Rolle mit den Systemberechtigungen und weisen sie den Benutzern zu.
5. Mit welchem Befehl können Sie Ihr Kennwort ändern?
Anweisung ALTER USER

6. User21 ist der Eigentümer der Tabelle EMP und erteilt User22 die Berechtigung DELETE mit der Klausel WITH GRANT OPTION. Daraufhin erteilt User22 Benutzer User23 die Berechtigung DELETE für EMP. User21 stellt nun fest, dass User23 über diese Berechtigung verfügt, und entzieht sie User22. Welcher Benutzer kann nun Löschvorgänge für die Tabelle EMP durchführen?

Nur User21

7. Sie möchten SCOTT die Berechtigung erteilen, Daten in der Tabelle DEPARTMENTS zu aktualisieren. SCOTT soll außerdem die Möglichkeit erhalten, diese Berechtigung an andere Benutzer weiterzugeben. Welchen Befehl verwenden Sie?

```
GRANT UPDATE ON departments TO scott WITH GRANT OPTION;
```

8. Erteilen Sie einem anderen Benutzer die Berechtigung, Ihre Tabelle abzufragen. Prüfen Sie anschließend, ob dieser Benutzer die Berechtigung verwenden kann.

Hinweis: Öffnen Sie für diese Übung eine weitere SQL Developer-Session, und melden Sie sich mit einem anderen Benutzeraccount an. Wenn Sie beispielsweise gerade den Account ora21 verwenden, öffnen Sie eine weitere SQL Developer-Session, und melden Sie sich als ora22 an. Nachfolgend steht "Team 1" für die erste und "Team 2" für die zweite SQL Developer-Session.

- a. Erteilen Sie einem weiteren Benutzer die Berechtigung, Datensätze in Ihrer Tabelle REGIONS anzuzeigen. Fügen Sie eine Option für diesen Benutzer hinzu, damit er diese Berechtigung an andere Benutzer weitergeben kann.

Hinweis: Ersetzen Sie <team2_oraxx> durch ora22, <team1_oraxx> durch ora21 und <team3_oraxx> durch ora23.

Team 1 führt folgende Anweisung aus:

```
GRANT select
ON regions
TO <team2_oraxx> WITH GRANT OPTION;
```

- b. Lassen Sie den Benutzer die Tabelle `REGIONS` abfragen.

Team 2 führt folgende Anweisung aus:

```
SELECT * FROM <team1_oraxx>.regions;
```

- c. Lassen Sie den Benutzer die Berechtigung an einen dritten Benutzer (`ora23`) weitergeben.

Team 2 führt folgende Anweisung aus.

```
GRANT select
ON <team1_oraxx>.regions
TO <team3_oraxx>;
```

- d. Entziehen Sie dem Benutzer, der Schritt b ausführt, die Berechtigung.

Team 1 führt folgende Anweisung aus:

```
REVOKE select
ON regions
FROM <team2_oraxx>;
```

9. Erteilen Sie einem anderen Benutzer Berechtigungen, mit denen er Daten in Ihrer Tabelle `COUNTRIES` abfragen und bearbeiten kann. Stellen Sie sicher, dass der Benutzer diese Berechtigungen nicht an andere Benutzer weitergeben kann.

Team 1 führt folgende Anweisung aus:

```
GRANT select, update, insert
ON COUNTRIES
TO <team2_oraxx>;
```

10. Entziehen Sie diesem Benutzer die für die Tabelle `COUNTRIES` erteilten Berechtigungen.

Team 1 führt folgende Anweisung aus.

```
REVOKE select, update, insert ON COUNTRIES FROM <team2_oraxx>;
```

11. Erteilen Sie einem anderen Benutzer Zugriff auf Ihre Tabelle `DEPARTMENTS`. Lassen Sie sich von diesem Benutzer die Berechtigung zur Abfrage seiner Tabelle `DEPARTMENTS` erteilen.

- a. Team 2 führt die Anweisung `GRANT` aus.

```
GRANT select
ON departments
TO <team1_oraxx>;
```

- b. Team 1 führt die Anweisung `GRANT` aus.

```
GRANT select
ON departments
TO <team2_oraxx>;
```

Hier ist `<team1_oraxx>` der Benutzername von Team 1 und `<team2_oraxx>` der Benutzername von Team 2.

12. Fragen Sie alle Zeilen in Ihrer Tabelle `DEPARTMENTS` ab.

```
SELECT *
FROM departments;
```


13. Fügen Sie Ihrer Tabelle `DEPARTMENTS` eine neue Zeile hinzu. Team 1 fügt die Abteilung "Education" mit der Abteilungsnummer 500 hinzu. Team 2 fügt die Abteilung "Human Resources" mit der Abteilungsnummer 510 hinzu. Fragen Sie die Tabelle des anderen Teams ab.

- a. Team 1 führt die Anweisung `INSERT` aus:

```
INSERT INTO departments(department_id, department_name)
VALUES (500, 'Education');
COMMIT;
```

- b. Team 2 führt die Anweisung `INSERT` aus:

```
INSERT INTO departments(department_id, department_name)
VALUES (510, 'Human Resources');
COMMIT;
```

14. Erstellen Sie ein Synonym für die Tabelle `DEPARTMENTS` des anderen Teams.

- a. Team 1 erstellt das Synonym "team2".

```
CREATE SYNONYM team2
FOR <team2_oraxx>.DEPARTMENTS;
```

- b. Team 2 erstellt das Synonym "team1".

```
CREATE SYNONYM team1
FOR <team1_oraxx>. DEPARTMENTS;
```

15. Fragen Sie alle Zeilen in der Tabelle `DEPARTMENTS` des anderen Teams ab, und verwenden Sie dabei Ihr Synonym.

- a. Team 1 führt die Anweisung `SELECT` aus:

```
SELECT *
FROM team2;
```

- b. Team 2 führt die Anweisung `SELECT` aus:

```
SELECT *
FROM team1;
```

16. Entziehen Sie dem anderen Team die Berechtigung `SELECT`.

- a. Team 1 entzieht die Berechtigung:

```
REVOKE select
ON departments
FROM <team2_oraxx>;
```

- b. Team 2 entzieht die Berechtigung:

```
REVOKE select
ON departments
FROM <team1_oraxx>;
```

17. Entfernen Sie die Zeile, die Sie im 13. Schritt in die Tabelle `DEPARTMENTS` eingefügt haben, und speichern Sie die Änderungen.

- a. Team 1 führt die Anweisung `DELETE` aus:

```
DELETE FROM departments
WHERE department_id = 500;
COMMIT;
```

b. Team 2 führt die Anweisung DELETE aus:

```
DELETE FROM departments
WHERE department_id = 510;
COMMIT;
```

18. Löschen Sie die Synonyme "team1" und "team2".

```
DROP SYNONYM team1;
DROP SYNONYM team2;
```

Übungen zu Lektion 9 – Daten bearbeiten

Kapitel 9

Übungen zu Lektion 9 – Überblick

Übungsüberblick:

Diese Übung behandelt folgende Themen:

- `INSERT`-Vorgänge für mehrere Tabellen ausführen
- `MERGE`-Vorgänge ausführen
- Flashback-Vorgänge ausführen
- Zeilenversionen überwachen

Hinweis: Bevor Sie diese Übung beginnen, führen Sie das Skript `/home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_09.sql` aus.

Übung 1 zu Lektion 9 – Daten bearbeiten

Überblick

In dieser Übung führen Sie `INSERT`-Vorgänge für mehrere Tabellen sowie `MERGE`- und `Flashback`-Vorgänge aus und überwachen Zeilenversionen.

Hinweis: Führen Sie das Skript `cleanup_09.sql` unter `//home/oracle/labs/sql2/code_ex/ /cleanup_scripts/` aus, bevor Sie die folgenden Aufgaben bearbeiten.

Aufgaben

1. Führen Sie im Ordner **lab** das Skript `lab_09_01.sql` aus, um die Tabelle `SAL_HISTORY` zu erstellen.
2. Zeigen Sie die Struktur der Tabelle `SAL_HISTORY` an.

```
DESC sal_history
Name          Null Type
-----
EMPLOYEE_ID    NUMBER(6)
HIRE_DATE      DATE
SALARY         NUMBER(8,2)
```

3. Um die Tabelle `MGR_HISTORY` zu erstellen, führen Sie im Ordner **lab** das Skript `lab_09_03.sql` aus.
4. Zeigen Sie die Struktur der Tabelle `MGR_HISTORY` an.

```
DESC mgr_history
Name          Null Type
-----
EMPLOYEE_ID    NUMBER(6)
MANAGER_ID     NUMBER(6)
SALARY         NUMBER(8,2)
```

5. Um die Tabelle `SPECIAL_SAL` zu erstellen, führen Sie im Ordner **lab** das Skript `lab_09_05.sql` aus.
6. Zeigen Sie die Struktur der Tabelle `SPECIAL_SAL` an.

```
DESC special_sal
Name          Null Type
-----
EMPLOYEE_ID    NUMBER(6)
SALARY         NUMBER(8,2)
```

7.
 - a. Erstellen Sie eine Abfrage, um folgende Aufgaben durchzuführen:
 - Details wie Personalnummer, Einstellungsdatum, Gehalt und Managernummer der Mitarbeiter, deren Personalnummer kleiner als 125 ist, aus der Tabelle `EMPLOYEES` abrufen
 - Personalnummer und Gehalt in die Tabelle `SPECIAL_SAL` einfügen, wenn das Gehalt höher als \$ 20.000 ist

- Wenn das Gehalt niedriger ist als \$ 20.000:
 - Personalnummer, Einstellungsdatum und Gehalt in die Tabelle `SAL_HISTORY` einfügen
 - Personalnummer, Manager-ID und Gehalt in die Tabelle `MGR_HISTORY` einfügen

b. Zeigen Sie die Datensätze aus der Tabelle `SPECIAL_SAL` an.

	EMPLOYEE_ID	SALARY
1	100	24000

c. Zeigen Sie die Datensätze aus der Tabelle `SAL_HISTORY` an.

	EMPLOYEE_ID	HIRE_DATE	SALARY
1	101	21-SEP-05	17000
2	102	13-JAN-01	17000
3	103	03-JAN-06	9000
4	104	21-MAY-07	6000
5	105	25-JUN-05	4800
6	106	05-FEB-06	4800
7	107	07-FEB-07	4200
8	108	17-AUG-02	12008
9	109	16-AUG-02	9000
10	110	28-SEP-05	8200
11	111	30-SEP-05	7700
12	112	07-MAR-06	7800
13	113	07-DEC-07	6900
14	114	07-DEC-02	11000

...

d. Zeigen Sie die Datensätze aus der Tabelle `MGR_HISTORY` an.

	EMPLOYEE_ID	MANAGER_ID	SALARY
1	101	100	17000
2	102	100	17000
3	103	102	9000
4	104	103	6000
5	105	103	4800
6	106	103	4800
7	107	103	4200
8	108	101	12008
9	109	108	9000
10	110	108	8200
11	111	108	7700
12	112	108	7800
13	113	108	6900

...

8.

- a. Um die Tabelle `SALES_WEEK_DATA` zu erstellen, führen Sie im Ordner **lab** das Skript `lab_09_08_a.sql` aus.
- b. Um Datensätze in die Tabelle `SALES_WEEK_DATA` einzufügen, führen Sie im Ordner **lab** das Skript `lab_09_08_b.sql` aus.
- c. Zeigen Sie die Struktur der Tabelle `SALES_WEEK_DATA` an.

```
DESC sales_week_data
Name      Null Type
-----
```

ID		NUMBER(6)
WEEK_ID		NUMBER(2)
QTY_MON		NUMBER(8,2)
QTY_TUE		NUMBER(8,2)
QTY_WED		NUMBER(8,2)
QTY_THUR		NUMBER(8,2)
QTY_FRI		NUMBER(8,2)

- d. Zeigen Sie die Datensätze der Tabelle `SALES_WEEK_DATA` an.

	ID	WEEK_ID	QTY_MON	QTY_TUE	QTY_WED	QTY_THUR	QTY_FRI
1	200	6	2050	2200	1700	1200	3000

- e. Um die Tabelle `EMP_SALES_INFO` zu erstellen, führen Sie im Ordner **lab** das Skript `lab_09_08_e.sql` aus.
- f. Zeigen Sie die Struktur der Tabelle `EMP_SALES_INFO` an.

```
DESC emp_sales_info
Name      Null Type
-----
```

ID		NUMBER(6)
WEEK		NUMBER(2)
QTY_SALES		NUMBER(8,2)

- g. Erstellen Sie eine Abfrage, um folgende Aufgaben durchzuführen:
 - Aus der Tabelle `SALES_WEEK_DATA` Details wie Personalnummer, Wochennummer, Umsatz am Montag, Umsatz am Dienstag, Umsatz am Mittwoch, Umsatz am Donnerstag und Umsatz am Freitag abrufen
 - Transformation erstellen, damit jeder aus der Tabelle `SALES_SOURCE_DATA` abgerufene Datensatz in mehrere Datensätze für die Tabelle `SALES_INFO` konvertiert wird.

Hinweis: Verwenden Sie eine `INSERT`-Anweisung mit `Pivoting`.
- h. Zeigen Sie Datensätze aus der Tabelle `EMP_SALES_INFO` an.

	ID	WEEK	QTY_SALES
1	200	6	2050
2	200	6	2200
3	200	6	1700
4	200	6	1200
5	200	6	3000

9. Sie haben Daten zu früheren Mitarbeitern in der Flat File `emp.data` gespeichert. Sie möchten die Namen und E-Mail-Nummern aller ehemaligen und aktuellen Mitarbeiter in einer Tabelle speichern. Erstellen Sie hierzu mit der Quelldatei `emp.dat` im Verzeichnis `emp_dir` die externe Tabelle `EMP_DATA`. Verwenden Sie hierzu das Skript `lab_09_09.sql`.
10. Führen Sie das Skript `lab_09_10.sql` aus, um die Tabelle `EMP_HIST` zu erstellen.
 - a. Erhöhen Sie die Größe der Spalte `EMAIL` auf 45.
 - b. Führen Sie die Daten in der Tabelle `EMP_DATA`, die im letzten Schritt erstellt wurde, mit den Daten in der Tabelle `EMP_HIST` zusammen. Die Daten in der externen Tabelle `EMP_DATA` sind die aktuellen Daten. Wenn eine Zeile in der Tabelle `EMP_DATA` mit der Tabelle `EMP_HIST` übereinstimmt, aktualisieren Sie die Spalte `EMAIL` der Tabelle `EMP_HIST`, sodass sie mit der Tabellenzeile in `EMP_DATA` übereinstimmt. Wenn für eine Zeile in der Tabelle `EMP_DATA` keine Übereinstimmung vorhanden ist, fügen Sie die Zeile in die Tabelle `EMP_HIST` ein. Zeilen gelten als übereinstimmend, wenn Vorname und Nachname des Mitarbeiters identisch sind.
 - c. Nachdem die Zeilen zusammengeführt wurden, rufen Sie sie aus `EMP_HIST` ab.

R	FIRST_NAME	R	LAST_NAME	R	EMAIL
1	Ellen	Abel	EABEL		
2	Sundar	Ande	SANDE		
3	Mozhe	Atkinson	MATKINSO		
4	David	Austin	DAUSTIN		
5	Hermann	Baer	HBAER		
6	Shelli	Baida	SBAIDA		
7	Amit	Banda	ABANDA		
8	Elizabeth	Bates	EBATES		
9	Sarah	Bell	SBELL		
10	David	Bernstein	DBERNSTE		
11	Laura	Bissot	LBISSOT		
12	Harrison	Bloom	HBLOOM		

...

11. Erstellen Sie die Tabelle EMP2 auf der Basis des folgenden Tabelleninstanzdiagramms. Geben Sie die Syntax in das SQL Worksheet ein. Führen Sie dann die Anweisung zum Erstellen der Tabelle aus. Prüfen Sie, ob die Tabelle erstellt wurde.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

12. Löschen Sie die Tabelle EMP2.
13. Um zu prüfen, ob die Tabelle vorhanden ist, fragen Sie den Papierkorb ab.
14. Stellen Sie den Zustand der Tabelle EMP2 zu einem Zeitpunkt vor der DROP-Anweisung wieder her.
15. Erstellen Sie mit dem Skript lab_09_11.sql die Tabelle EMP3. Ändern Sie in der Tabelle EMP3 die Abteilung für Kochhar in 60, und schreiben Sie die Änderung fest. Ändern Sie anschließend die Abteilung für Kochhar in 50, und schreiben Sie die Änderung fest. Überwachen Sie die Änderungen für Kochhar mithilfe des Zeilenversionsfeatures.

```

UPDATE emp3 SET department_id = 60
WHERE last_name = 'Kochhar';
COMMIT;
UPDATE emp3 SET department_id = 50
WHERE last_name = 'Kochhar';
COMMIT;

SELECT VERSIONS_STARTTIME "START_DATE",
       VERSIONS_ENDTIME "END_DATE", DEPARTMENT_ID
FROM EMP3
      VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE LAST_NAME = 'Kochhar';

```

	START_DATE	END_DATE	DEPARTMENT_ID
1	28-APR-14 10.11.40.000000000 PM (null)		50
2	28-APR-14 10.11.37.000000000 PM	28-APR-14 10.11.40.000000000 PM	60
3	(null)	28-APR-14 10.11.37.000000000 PM	90

16. Löschen Sie die Tabellen EMP2 und EMP3 so, dass sie nicht wiederhergestellt werden können. Prüfen Sie den Papierkorb.

Übung 1 zu Lektion 9 – Lösung: Daten bearbeiten

Lösung

1. Führen Sie im Ordner **lab** das Skript `lab_09_01.sql` aus, um die Tabelle `SAL_HISTORY` zu erstellen.
2. Zeigen Sie die Struktur der Tabelle `SAL_HISTORY` an.

```
DESC sal_history
```

3. Um die Tabelle `MGR_HISTORY` zu erstellen, führen Sie im Ordner **lab** das Skript `lab_09_03.sql` aus.
4. Zeigen Sie die Struktur der Tabelle `MGR_HISTORY` an.

```
DESC mgr_history
```

5. Um die Tabelle `SPECIAL_SAL` zu erstellen, führen Sie im Ordner **lab** das Skript `lab_09_05.sql` aus.
6. Zeigen Sie die Struktur der Tabelle `SPECIAL_SAL` an.

```
DESC special_sal
```

7.
 - a. Erstellen Sie eine Abfrage, um folgende Aufgaben durchzuführen:
 - Details wie Personalnummer, Einstellungsdatum, Gehalt und Managernummer der Mitarbeiter, deren Personalnummer kleiner als 125 ist, aus der Tabelle `EMPLOYEES` abrufen
 - Personalnummer und Gehalt in die Tabelle `SPECIAL_SAL` einfügen, wenn das Gehalt höher als \$ 20.000 ist
 - Wenn das Gehalt niedriger ist als \$ 20.000:
 - Personalnummer, Einstellungsdatum und Gehalt in die Tabelle `SAL_HISTORY` einfügen
 - Personalnummer, Manager-ID und Gehalt in die Tabelle `MGR_HISTORY` einfügen

```
INSERT ALL
WHEN SAL > 20000 THEN
INTO special_sal VALUES (EMPID, SAL)
ELSE
INTO sal_history VALUES(EMPID, HIREDATE, SAL)
INTO mgr_history VALUES(EMPID, MGR, SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
salary SAL, manager_id MGR
FROM employees
WHERE employee_id < 125;
```

- b. Zeigen Sie die Datensätze aus der Tabelle SPECIAL_SAL an.

```
SELECT * FROM special_sal;
```

- c. Zeigen Sie die Datensätze aus der Tabelle SAL_HISTORY an.

```
SELECT * FROM sal_history;
```

- d. Zeigen Sie die Datensätze aus der Tabelle MGR_HISTORY an.

```
SELECT * FROM mgr_history;
```

8.

- a. Um die Tabelle SALES_WEEK_DATA zu erstellen, führen Sie im Ordner **lab** das Skript lab_09_08_a.sql aus.
- b. Um Datensätze in die Tabelle SALES_WEEK_DATA einzufügen, führen Sie im Ordner **lab** das Skript lab_09_08_b.sql aus.
- c. Zeigen Sie die Struktur der Tabelle SALES_WEEK_DATA an.

```
DESC sales_week_data
```

- d. Zeigen Sie die Datensätze aus der Tabelle SALES_WEEK_DATA an.

```
SELECT * FROM SALES_WEEK_DATA;
```

- e. Um die Tabelle EMP_SALES_INFO zu erstellen, führen Sie im Ordner **lab** das Skript lab_09_08_e.sql aus.
- f. Zeigen Sie die Struktur der Tabelle EMP_SALES_INFO an.

```
DESC emp_sales_info
```

- g. Erstellen Sie eine Abfrage, um folgende Aufgaben durchzuführen:

- Aus der Tabelle SALES_WEEK_DATA Details wie Personalnummer, Wochennummer, Umsatz am Montag, Umsatz am Dienstag, Umsatz am Mittwoch, Umsatz am Donnerstag und Umsatz am Freitag abrufen
- Transformation erstellen, damit jeder aus der Tabelle SALES_SOURCE_DATA abgerufene Datensatz in mehrere Datensätze für die Tabelle SALES_INFO konvertiert wird.

Hinweis: Verwenden Sie eine INSERT-Anweisung mit Pivoting.

```
INSERT ALL
  INTO emp_sales_info VALUES (id, week_id, QTY_MON)
  INTO emp_sales_info VALUES (id, week_id, QTY_TUE)
  INTO emp_sales_info VALUES (id, week_id, QTY_WED)
  INTO emp_sales_info VALUES (id, week_id, QTY_THUR)
  INTO emp_sales_info VALUES (id, week_id, QTY_FRI)
SELECT ID, week_id, QTY_MON, QTY_TUE, QTY_WED,
       QTY_THUR, QTY_FRI FROM sales_week_data;
```

- h. Zeigen Sie die Datensätze aus der Tabelle `SALES_INFO` an.

```
SELECT * FROM emp_sales_info;
```

9. Sie haben Daten zu früheren Mitarbeitern in der Flat File `emp.data` gespeichert. Sie möchten die Namen und E-Mail-Nummern aller ehemaligen und aktuellen Mitarbeiter in einer Tabelle speichern. Erstellen Sie hierzu mit der Quelldatei `emp.dat` im Verzeichnis `emp_dir` die externe Tabelle `EMP_DATA`. Dazu können Sie das Skript in `lab_09_09.sql` verwenden.

```
CREATE TABLE emp_data
(
  first_name VARCHAR2(20)
, last_name  VARCHAR2(20)
, email      VARCHAR2(30)
)
ORGANIZATION EXTERNAL
(
  TYPE oracle_loader
  DEFAULT DIRECTORY emp_dir
  ACCESS PARAMETERS
  (
    RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    NOBADFILE
    NOLOGFILE
    FIELDS
    ( first_name POSITION ( 1:20) CHAR
    , last_name  POSITION (22:41) CHAR
    , email      POSITION (43:72) CHAR )
  )
  LOCATION ('emp.dat') ) ;
```

10. Führen Sie das Skript `lab_09_10.sql` aus, um die Tabelle `EMP_HIST` zu erstellen.

- a. Erhöhen Sie die Größe der Spalte `EMAIL` auf 45.

```
ALTER TABLE emp_hist MODIFY email varchar(45);
```

- b. Führen Sie die Daten in der Tabelle `EMP_DATA`, die im letzten Schritt erstellt wurde, mit den Daten in der Tabelle `EMP_HIST` zusammen. Die Daten in der externen Tabelle `EMP_DATA` sind die aktuellen Daten. Wenn eine Zeile in der Tabelle `EMP_DATA` mit der Tabelle `EMP_HIST` übereinstimmt, aktualisieren Sie die Spalte `EMAIL` der Tabelle `EMP_HIST`, sodass sie mit der Tabellenzeile in `EMP_DATA` übereinstimmt. Wenn für eine Zeile in der Tabelle `EMP_DATA` keine Übereinstimmung vorhanden ist, fügen Sie die Zeile in die Tabelle `EMP_HIST` ein. Zeilen gelten als übereinstimmend, wenn Vorname und Nachname des Mitarbeiters identisch sind.

```
MERGE INTO EMP_HIST f USING EMP_DATA h
ON (f.first_name = h.first_name
```

```

AND f.last_name = h.last_name)
WHEN MATCHED THEN
  UPDATE SET f.email = h.email
WHEN NOT MATCHED THEN
  INSERT (f.first_name
        , f.last_name
        , f.email)
VALUES (h.first_name
      , h.last_name
      , h.email);

```

- c. Nachdem die Zeilen zusammengeführt wurden, rufen Sie sie aus EMP_HIST ab.

```
SELECT * FROM emp_hist;
```

11. Erstellen Sie die Tabelle EMP2 auf der Basis des folgenden Tabelleninstanzdiagramms. Geben Sie die Syntax in das SQL Worksheet ein. Führen Sie dann die Anweisung zum Erstellen der Tabelle aus. Prüfen Sie, ob die Tabelle erstellt wurde.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

```

CREATE TABLE emp2
(id          NUMBER(7) ,
 last_name   VARCHAR2(25) ,
 first_name  VARCHAR2(25) ,
 dept_id     NUMBER(7));

DESCRIBE emp2

```

12. Löschen Sie die Tabelle EMP2.

```
DROP TABLE emp2;
```

13. Um zu prüfen, ob die Tabelle vorhanden ist, fragen Sie den Papierkorb ab.

```

SELECT original_name, operation, droptime
FROM recyclebin;

```

14. Stellen Sie den Zustand der Tabelle EMP2 zu einem Zeitpunkt vor der DROP-Anweisung wieder her.

```
FLASHBACK TABLE emp2 TO BEFORE DROP;  
DESC emp2;
```

15. Erstellen Sie mit dem Skript lab_09_11.sql die Tabelle EMP3. Ändern Sie in der Tabelle EMP3 die Abteilung für Kochhar in 60, und schreiben Sie die Änderung fest. Ändern Sie anschließend die Abteilung für Kochhar in 50, und schreiben Sie die Änderung fest. Überwachen Sie die Änderungen für Kochhar mithilfe des Zeilenversionsfeatures.

```
UPDATE emp3 SET department_id = 60  
WHERE last_name = 'Kochhar';  
COMMIT;  
UPDATE emp3 SET department_id = 50  
WHERE last_name = 'Kochhar';  
COMMIT;  
  
SELECT VERSIONS_STARTTIME "START_DATE",  
       VERSIONS_ENDTIME "END_DATE", DEPARTMENT_ID  
FROM EMP3  
       VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE  
WHERE LAST_NAME = 'Kochhar';
```

16. Löschen Sie die Tabellen EMP2 und EMP3 so, dass sie nicht wiederhergestellt werden können. Prüfen Sie den Papierkorb.

```
DROP TABLE emp2 PURGE;  
DROP TABLE emp3 PURGE;  
  
SELECT original_name, operation, droptime  
FROM recyclebin;
```

Übungen zu Lektion 10 – Daten in verschiedenen Zeitzonen verwalten

Kapitel 10

Übungen zu Lektion 10 – Überblick

Übungsüberblick:

Diese Übung behandelt die Verwendung der Datetime-Funktionen.

Hinweis: Bevor Sie diese Übung beginnen, führen Sie das Skript

`/home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_10.sql` aus.

Übung 1 zu Lektion 10 – Daten in verschiedenen Zeitzonen verwalten

Überblick

In dieser Übung zeigen Sie Zeitzonendifferenzen sowie `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` an. Außerdem stellen Sie Zeitzonen ein und verwenden die Funktion `EXTRACT`.

Hinweis: Führen Sie das Skript `cleanup_10.sql` unter `/home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_10.sql` aus, bevor Sie die folgenden Aufgaben bearbeiten.


Aufgaben

1. Ändern Sie die Session, und stellen Sie `NLS_DATE_FORMAT` auf `DD-MON-YYYY HH24:MI:SS` ein.
2.
 - a. Erstellen Sie Abfragen, um die Zeitzonendifferenzen (`TZ_OFFSET`) für die folgenden Zeitzonen anzuzeigen.


- US/Pacific-New

 <code>TZ_OFFSET('US/PACIFIC-NEW')</code>
1 -07:00




- Singapore

 <code>TZ_OFFSET('SINGAPORE')</code>
1 +08:00

- Egypt




 <code>TZ_OFFSET('EGYPT')</code>
1 +02:00

- b. Ändern Sie die Session. Stellen Sie den Wert des Parameters `TIME_ZONE` auf die Zeitzonendifferenz von "US/Pacific-New" ein.
- c. Zeigen Sie `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` für die Session an.

 <code>CURRENT_DATE</code>	 <code>CURRENT_TIMESTAMP</code>	 <code>LOCALTIMESTAMP</code>
1 16-SEP-2012 21:03:47	16-SEP-12 09.03.47.344991000 PM -07:00	16-SEP-12 09.03.47.344991000 PM

- d. Ändern Sie die Session. Stellen Sie den Parameter `TIME_ZONE` auf die Zeitzonendifferenz von "Singapore" ein.
- e. Zeigen Sie `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` für die Session an.

Hinweis: Die Ausgabe kann entsprechend dem Ausführungsdatum des Befehls variieren.

 <code>CURRENT_DATE</code>	 <code>CURRENT_TIMESTAMP</code>	 <code>LOCALTIMESTAMP</code>
1 17-SEP-2012 12:05:56	17-SEP-12 12.05.56.424157000 PM +08:00	17-SEP-12 12.05.56.424157000 PM

Hinweis: In der vorherigen Übung sind CURRENT_DATE, CURRENT_TIMESTAMP und LOCALTIMESTAMP von der Zeitzone der Session abhängig.

3. Erstellen Sie eine Abfrage, die DBTIMEZONE und SESSIONTIMEZONE anzeigt.

	DBTIMEZONE	SESSIONTIMEZONE
1	+00:00	+08:00

4. Erstellen Sie eine Abfrage, die für alle Mitarbeiter aus Abteilung 80 die Jahreszahl aus der Spalte HIRE_DATE der Tabelle EMPLOYEES extrahiert.

	LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
1	Russell	2004
2	Partners	2005
3	Errazuriz	2005
4	Cambrault	2007
5	Zlotkey	2008
6	Tucker	2005
7	Bernstein	2005
8	Hall	2005
9	Olsen	2006
10	Cambrault	2006
11	Tuvault	2007

...

5. Ändern Sie die Session, und stellen Sie NLS_DATE_FORMAT auf DD-MON-YYYY ein.
6. Prüfen Sie das Skript lab_10_06.sql, und führen Sie es aus, um die Tabelle SAMPLE_DATES zu erstellen und zu füllen.

Hinweis: Das im Screenshot angezeigte Datum wird an sysdate angepasst.

- a. Wählen Sie Daten aus der Tabelle, und zeigen Sie sie an.

	DATE_COL
1	17-SEP-2012

- b. Ändern Sie den Datentyp der Spalte DATE_COL in TIMESTAMP. Wählen Sie Daten aus der Tabelle, und zeigen Sie sie an.

	DATE_COL
1	17-SEP-12 04.09.12.000000000 AM

- c. Ändern Sie den Datentyp der Spalte DATE_COL in TIMESTAMP WITH TIME ZONE. Was geschieht?

```
Error report:
SQL Error: ORA-01439: column to be modified must be empty to change datatype
01439. 00000 - "column to be modified must be empty to change datatype"
*Cause:
*Action:
```

7. Erstellen Sie eine Abfrage, um die Nachnamen aus der Tabelle `EMPLOYEES` abzurufen und den Überprüfungsstatus zu ermitteln. Wenn das Einstellungsjahr 2008 ist, soll als Überprüfungsstatus `Needs Review` angezeigt werden. Andernfalls soll `not this year!` angezeigt werden. Nennen Sie die Spalte für den Überprüfungsstatus `Review`. Sortieren Sie die Ergebnisse nach der Spalte `HIRE_DATE`.

Tipp: Um den Überprüfungsstatus zu ermitteln, verwenden Sie den Ausdruck `CASE` mit der Funktion `EXTRACT`.

	LAST_NAME	Review
1	De Haan	not this year!
2	Gietz	not this year!
3	Baer	not this year!
4	Mavris	not this year!
5	Higgins	not this year!
6	Faviet	not this year!
7	Greenberg	not this year!
8	Raphaely	not this year!
9	Kaufling	not this year!

...

8. Erstellen Sie eine Abfrage, um für alle Mitarbeiter den Nachnamen und die Beschäftigungsdauer auszugeben. Für Mitarbeiter, die fünf Jahre oder länger beschäftigt sind, soll `5 years of service` ausgegeben werden. Für Mitarbeiter, die 10 Jahre oder länger beschäftigt sind, soll `10 years of service` ausgegeben werden. Für Mitarbeiter, die 15 Jahre oder länger beschäftigt sind, soll `15 years of service` ausgegeben werden. Wenn keine dieser Bedingungen zutrifft, soll `maybe next year!` ausgegeben werden. Sortieren Sie die Ergebnisse nach der Spalte `HIRE_DATE`. Verwenden Sie die Tabelle `EMPLOYEES`.

Tipp: Verwenden Sie `CASE`-Ausdrücke und `TO_YMINTERVAL`.

	LAST_NAME	HIRE_DATE	SYSDATE	Awards
1	King	17-JUN-03	17-SEP-2012	5 years of service
2	Kochhar	21-SEP-05	17-SEP-2012	5 years of service
3	De Haan	13-JAN-01	17-SEP-2012	10 years of service
4	Hunold	03-JAN-06	17-SEP-2012	5 years of service
5	Ernst	21-MAY-07	17-SEP-2012	5 years of service
6	Austin	25-JUN-05	17-SEP-2012	5 years of service
7	Pataballa	05-FEB-06	17-SEP-2012	5 years of service
8	Lorentz	07-FEB-07	17-SEP-2012	5 years of service
9	Greenberg	17-AUG-02	17-SEP-2012	10 years of service
10	Faviet	16-AUG-02	17-SEP-2012	10 years of service
11	Chen	28-SEP-05	17-SEP-2012	5 years of service
12	Sciarra	30-SEP-05	17-SEP-2012	5 years of service
13	Urman	07-MAR-06	17-SEP-2012	5 years of service

...

Übung 1 zu Lektion 10 – Lösung: Daten in verschiedenen Zeitzonen verwalten

Lösung

1. Ändern Sie die Session, und stellen Sie `NLS_DATE_FORMAT` auf `DD-MON-YYYY HH24:MI:SS` ein.

```
ALTER SESSION SET NLS_DATE_FORMAT =  
'DD-MON-YYYY HH24:MI:SS';
```

2.

- a. Erstellen Sie Abfragen, um die Zeitzonendifferenzen (`TZ_OFFSET`) für die folgenden Zeitzonen anzuzeigen: *US/Pacific-New*, *Singapore* und *Egypt*.

```
US/Pacific-New  
SELECT TZ_OFFSET ('US/Pacific-New') from dual;  
Singapore  
SELECT TZ_OFFSET ('Singapore') from dual;  
Egypt  
SELECT TZ_OFFSET ('Egypt') from dual;
```

- b. Ändern Sie die Session. Stellen Sie den Wert des Parameters `TIME_ZONE` auf die Zeitzonendifferenz von "US/Pacific-New" ein.

```
ALTER SESSION SET TIME_ZONE = '-7:00';
```

- c. Zeigen Sie `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` für diese Session an.

Hinweis: Die Ausgabe kann entsprechend dem Ausführungsdatum des Befehls variieren.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

- d. Ändern Sie die Session. Stellen Sie den Parameter `TIME_ZONE` auf die Zeitzonendifferenz von "Singapore" ein.

```
ALTER SESSION SET TIME_ZONE = '+8:00';
```

- e. Zeigen Sie `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` für diese Session an.

Hinweis: Die Ausgabe kann entsprechend dem Ausführungsdatum des Befehls variieren.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

Hinweis: Wie Sie sehen, sind in der vorherigen Übung `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` von der Zeitzone der Session abhängig.

3. Erstellen Sie eine Abfrage, die DBTIMEZONE und SESSIONTIMEZONE anzeigt.

```
SELECT DBTIMEZONE, SESSIONTIMEZONE  
FROM DUAL;
```

4. Erstellen Sie eine Abfrage, die für alle Mitarbeiter aus Abteilung 80 die Jahreszahl aus der Spalte HIRE_DATE der Tabelle EMPLOYEES extrahiert.

```
SELECT last_name, EXTRACT (YEAR FROM HIRE_DATE)  
FROM employees  
WHERE department_id = 80;
```

5. Ändern Sie die Session, und stellen Sie NLS_DATE_FORMAT auf DD-MON-YYYY ein.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

6. Prüfen Sie das Skript lab_10_06.sql, und führen Sie es aus, um die Tabelle SAMPLE_DATES zu erstellen und zu füllen.

- a. Wählen Sie Daten aus der Tabelle, und zeigen Sie sie an.

```
SELECT * FROM sample_dates;
```

- b. Ändern Sie den Datentyp der Spalte DATE_COL in TIMESTAMP. Wählen Sie Daten aus der Tabelle, und zeigen Sie sie an.

```
ALTER TABLE sample_dates MODIFY date_col TIMESTAMP;  
SELECT * FROM sample_dates;
```

- c. Ändern Sie den Datentyp der Spalte DATE_COL in TIMESTAMP WITH TIME ZONE. Was geschieht?

```
ALTER TABLE sample_dates MODIFY date_col  
TIMESTAMP WITH TIME ZONE;
```

Der Datentyp der Spalte DATE_COL kann nicht geändert werden, da der Oracle-Server keine Konvertierung von TIMESTAMP in TIMESTAMP WITH TIMEZONE mit der Anweisung ALTER zulässt.

7. Erstellen Sie eine Abfrage, um die Nachnamen aus der Tabelle `EMPLOYEES` abzurufen und den Überprüfungsstatus zu ermitteln. Wenn das Einstellungsjahr 2008 ist, soll als Überprüfungsstatus `Needs Review` angezeigt werden. Andernfalls soll `not this year!` angezeigt werden. Nennen Sie die Spalte für den Überprüfungsstatus `Review`. Sortieren Sie die Ergebnisse nach der Spalte `HIRE_DATE`.

Tipp: Um den Überprüfungsstatus zu ermitteln, verwenden Sie den Ausdruck `CASE` mit der Funktion `EXTRACT`.

```
SELECT e.last_name
      , (CASE extract(year from e.hire_date)
          WHEN 2008 THEN 'Needs Review'
          ELSE 'not this year!'
        END )          AS "Review "
FROM   employees e
ORDER BY e.hire_date;
```

8. Erstellen Sie eine Abfrage, um für alle Mitarbeiter den Nachnamen und die Beschäftigungsdauer auszugeben. Wenn der Mitarbeiter fünf Jahre oder länger beschäftigt ist, soll `5 years of service` ausgegeben werden. Wenn der Mitarbeiter 10 Jahre oder länger beschäftigt ist, soll `10 years of service` ausgegeben werden. Wenn der Mitarbeiter 15 Jahre oder länger beschäftigt ist, soll `15 years of service` ausgegeben werden. Wenn keine dieser Bedingungen zutrifft, soll `maybe next year!` ausgegeben werden. Sortieren Sie die Ergebnisse nach der Spalte `HIRE_DATE`. Verwenden Sie die Tabelle `EMPLOYEES`.

Tipp: Verwenden Sie `CASE`-Ausdrücke und `TO_YMINTERVAL`.

```
SELECT e.last_name, hire_date, sysdate,
      (CASE
        WHEN (sysdate -TO_YMINTERVAL('15-0'))>=
             hire_date THEN      '15 years of service'
        WHEN (sysdate -TO_YMINTERVAL('10-0'))>= hire_date
             THEN '10 years of service'
        WHEN (sysdate - TO_YMINTERVAL('5-0'))>= hire_date
             THEN '5 years of service'
        ELSE 'maybe next year!'
      END) AS "Awards"
FROM   employees e;
```

Zusätzliche Übungen und Lösungen

Kapitel 11

Zusätzliche Übungen und Lösungen

Übungsüberblick

In diesen Übungen können Sie zusätzliche praktische Erfahrungen zu folgenden Themen sammeln:

- DML-(Data Manipulation Language-)Anweisungen
- DDL-(Data Definition Language-)Anweisungen
- Datetime-Funktionen
- Fortgeschrittene Unterabfragen

Zusätzliche Übungen

Überblick

Die folgenden Übungen können als zusätzliche Übung dienen, nachdem Sie die DML-(Data Manipulation Language-) und DDL-(Data Definition Language-)Anweisungen in den Lektionen "Schemaobjekte verwalten" und "Daten bearbeiten" behandelt haben.

Hinweis:Führen Sie die Skripte `lab_ap_cre_special_sal.sql`, `lab_ap_cre_sal_history.sql` und `lab_ap_cre_mgr_history.sql` aus dem Übungsordner aus, um die Tabellen `SPECIAL_SAL`, `SAL_HISTORY` und `MGR_HISTORY` zu erstellen.

Aufgaben

1. Die Personalabteilung benötigt eine Liste der Mitarbeiter mit dem niedrigsten Gehalt, eine Gehaltshistorie der Mitarbeiter und eine Gehaltshistorie der Manager auf der Basis einer branchenbezogenen Gehaltsübersicht. Sie wurden daher aufgefordert, die folgenden Aufgaben auszuführen:

Erstellen Sie eine Anweisung, um folgende Aufgaben auszuführen:

- Details wie Personalnummer, Einstellungsdatum, Gehalt und Manager-ID der Mitarbeiter, deren Personalnummer größer oder gleich 200 ist, aus der Tabelle `EMPLOYEES` abrufen
- Für ein Gehalt unter \$ 5.000 Details wie Personalnummer und Gehalt in die Tabelle `SPECIAL_SAL` einfügen
- Personalnummer, Einstellungsdatum und Gehalt in die Tabelle `SAL_HISTORY` einfügen
- Personalnummer, Manager-ID und Gehalt in die Tabelle `MGR_HISTORY` einfügen

2. Fragen Sie die Tabellen `SPECIAL_SAL`, `SAL_HISTORY` und `MGR_HISTORY` ab, um die eingefügten Datensätze anzuzeigen.

SPECIAL_SAL

	EMPLOYEE_ID	SALARY
1	200	4400

SAL_HISTORY

	EMPLOYEE_ID	HIRE_DATE	SALARY
1	201	17-FEB-04	13000
2	202	17-AUG-05	6000
3	203	07-JUN-02	6500
4	204	07-JUN-02	10000
5	205	07-JUN-02	12008
6	206	07-JUN-02	8300

MGR_HISTORY

	EMPLOYEE_ID	MANAGER_ID	SALARY
1	201	100	13000
2	202	201	6000
3	203	101	6500
4	204	101	10000
5	205	101	12008
6	206	205	8300

3. Die DBA Nita beauftragt Sie, eine Tabelle mit einem Constraint vom Typ PRIMARY KEY zu erstellen. Der Index soll aber einen anderen Namen aufweisen als das Constraint. Erstellen Sie anhand des folgenden Tabelleninstanzdiagramms die Tabelle LOCATIONS_NAMED_INDEX. Geben Sie dem Index für die Spalte PRIMARY KEY den Namen LOCATIONS_PK_IDX.

Column Name	Deptno	Dname
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	4	30

4. Fragen Sie die Tabelle USER_INDEXES ab, um INDEX_NAME für die Tabelle LOCATIONS_NAMED_INDEX anzuzeigen.

INDEX_NAME	TABLE_NAME
1 LOCATIONS_PK_IDX	LOCATIONS_NAMED_INDEX

In den folgenden Übungen können Sie zusätzliche praktische Erfahrungen zu den Datetime-Funktionen sammeln.

Sie arbeiten für ein internationales Unternehmen. Der neue Vice President of Operations möchte die verschiedenen Zeitzonen aller Unternehmensniederlassungen wissen. Er hat die folgenden Informationen angefordert:

5. Ändern Sie die Session, und stellen Sie NLS_DATE_FORMAT auf DD-MON-YYYY HH24:MI:SS ein.

6.

- a. Erstellen Sie Abfragen, um die Zeitzonendifferenzen (TZ_OFFSET) für die folgenden Zeitzonen anzuzeigen:

- Australia/Sydney

R	Z	TZ_OFFSET('AUSTRALIA/SYDNEY')
1		+10:00

- Chile/Easter Island

R	Z	TZ_OFFSET('CHILE/EASTERISLAND')
1		-06:00

- b. Ändern Sie die Session, und stellen Sie den Wert des Parameters TIME_ZONE auf die Zeitzonendifferenz von "Australia/Sydney" ein.
- c. Zeigen Sie SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP und LOCALTIMESTAMP für diese Session an.

Hinweis: Die Ausgabe kann entsprechend dem Ausführungsdatum des Befehls variieren.

R	SYSDATE	R	CURRENT_DATE	R	CURRENT_TIMESTAMP	R	LOCALTIMESTAMP
1	17-SEP-2012 22:59:31	18-SEP-2012 08:59:31	18-SEP-12 08.59.31.390170000 AM	+10:00	18-SEP-12 08.59.31.		

- d. Ändern Sie die Session, und stellen Sie den Wert des Parameters TIME_ZONE auf die Zeitzonendifferenz von Chile/Easter Island ein.

Hinweis: Die Ergebnisse dieser Aufgabe basieren auf einem anderen Datum und entsprechen in einigen Fällen nicht den Ergebnissen, die die Kursteilnehmer erzielen. Darüber hinaus kann die Zeitzonendifferenz der verschiedenen Länder aufgrund der Sommerzeit variieren.

- e. Zeigen Sie SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP und LOCALTIMESTAMP für diese Session an.

Hinweis: Die Ausgabe kann entsprechend dem Ausführungsdatum des Befehls variieren.

R	SYSDATE	R	CURRENT_DATE	R	CURRENT_TIMESTAMP	R	LOCALTIMESTAMP
1	17-SEP-2012 23:00:01	17-SEP-2012 17:00:01	17-SEP-12 05.00.01.121124000 PM	-06:00	17-SEP-12 05.00.01.		

- f. Ändern Sie die Session, und stellen Sie NLS_DATE_FORMAT auf DD-MON-YYYY ein.

Hinweis:

- Wie Sie sehen, richten sich bei der vorherigen Aufgabe CURRENT_DATE, CURRENT_TIMESTAMP und LOCALTIMESTAMP jeweils nach der Sessionzeitzone, SYSDATE jedoch nicht.
- Die Ergebnisse dieser Aufgabe basieren auf einem anderen Datum und entsprechen in einigen Fällen nicht den Ergebnissen, die die Kursteilnehmer erzielen. Darüber hinaus kann die Zeitzonendifferenz der verschiedenen Länder aufgrund der Sommerzeit variieren.

7. Die Personalabteilung benötigt eine Liste der Mitarbeiter, die im Januar geprüft werden sollen. Sie wurden beauftragt, die folgenden Aufgaben auszuführen:

Erstellen Sie eine Abfrage, mit der Nachname, Einstellungsmonat und Einstellungsdatum der Mitarbeiter angezeigt werden, die, unabhängig vom Einstellungsjahr, im Januar in die Firma eingetreten sind.

R	LAST_NAME	R	EXTRACT(MONTHFROMHIRE_DATE)	R	HIRE_DATE
1	De Haan			1	13-JAN-2001
2	Hunold			1	03-JAN-2006
3	Landry			1	14-JAN-2007
4	Davies			1	29-JAN-2005
5	Partners			1	05-JAN-2005
6	Zlotkey			1	29-JAN-2008
7	Tucker			1	30-JAN-2005
8	King			1	30-JAN-2004
9	Marvins			1	24-JAN-2008
10	Fox			1	24-JAN-2006
11	Johnson			1	04-JAN-2008
12	Taylor			1	24-JAN-2006
13	Sarchand			1	27-JAN-2004
14	Grant			1	13-JAN-2008

Bei den folgenden Übungen können Sie zusätzliche praktische Erfahrungen zu fortgeschrittenen Unterabfragen sammeln.

8. Der CEO benötigt für die Ausschüttung der Gewinnbeteiligung einen Bericht über die drei Spitzenverdiener im Unternehmen. Sie müssen diese Liste für den CEO erstellen. Erstellen Sie eine Abfrage, um die drei Spitzenverdiener in der Tabelle `EMPLOYEES` anzuzeigen. Zeigen Sie ihre Nachnamen und Gehälter an.

R	LAST_NAME	R	SALARY
1	King		24000
2	Kochhar		17000
3	De Haan		17000

9. Die staatlichen Leistungen haben sich im Bundesstaat Kalifornien aufgrund einer Verfügung geändert. Der zuständige Sachbearbeiter bittet Sie, eine Liste der hiervon betroffenen Personen zu erstellen.
Erstellen Sie eine Abfrage, um die Personalnummern und die Nachnamen der Mitarbeiter anzuzeigen, die in Kalifornien arbeiten.

Tipp: Verwenden Sie skalare Unterabfragen.

	EMPLOYEE_ID	LAST_NAME
1	120	Weiss
2	121	Fripp
3	122	Kaufling
4	123	Vollman
5	124	Mourgos
6	125	Nayer
7	126	Mikkilineni
8	127	Landry
9	128	Markle
10	129	Bissot
11	130	Atkinson
12	131	Marlow
13	132	Olson
14	133	Mallin
15	134	Rogers
16	135	Gee
17	136	Philtanker
18	137	Ladwig

...

10. Die DBA Nita möchte alte Informationen aus der Datenbank entfernen. Hierzu gehören auch alte Mitarbeiterdatensätze. Sie wurden beauftragt, die folgenden Aufgaben auszuführen:

Erstellen Sie eine Abfrage zum Löschen der ältesten `JOB_HISTORY`-Zeile eines Mitarbeiters. Hierfür muss die Tabelle `JOB_HISTORY` nach dem `MIN (START_DATE)` des Mitarbeiters durchsucht werden. Löschen Sie *nur* die Datensätze der Mitarbeiter, die mindestens zweimal die Tätigkeit gewechselt haben.



Tipp: Verwenden Sie einen korrelierten `DELETE`-Befehl.

11. Der Leiter der Personalabteilung benötigt die vollständigen Mitarbeiterdatensätze für seine jährliche Rede zur Auszeichnung von Mitarbeitern. Er ruft Sie kurz an, damit Sie die Arbeit am Auftrag der DBA einstellen.

Rollen Sie die Transaktion zurück.

12. Die schwache Wirtschaftslage zwingt das Management, Kostensenkungsmaßnahmen zu ergreifen. Der CEO möchte die Tätigkeiten mit den höchsten Gehältern im Unternehmen prüfen. Sie müssen diese Liste für den CEO auf der Basis folgender Angaben erstellen.

Erstellen Sie eine Abfrage, um die Tätigkeits-IDs der Tätigkeiten anzuzeigen, deren Höchstgehalt höher ist als 50 % des unternehmensweit höchsten Gehalts. Verwenden Sie für diese Abfrage die Klausel `WITH`. Nennen Sie die Abfrage `MAX_SAL_CALC`.

 JOB_TITLE	 JOB_TOTAL
1 President	24000
2 Administration Vice President	17000
3 Sales Manager	14000
4 Marketing Manager	13000
5 Finance Manager	12008
6 Accounting Manager	12008

Zusätzliche Übungen – Lösungen

Lösung

Die folgenden Übungen können als zusätzliche Übung dienen, nachdem Sie die DML-(Data Manipulation Language-) und DDL-(Data Definition Language-)Anweisungen in den Lektionen "Schemaobjekte verwalten" und "Daten bearbeiten" behandelt haben.

Hinweis: Führen Sie die Skripte `lab_ap_cre_special_sal.sql`, `lab_ap_cre_sal_history.sql` und `lab_ap_cre_mgr_history.sql` aus dem Übungsordner aus, um die Tabellen `SPECIAL_SAL`, `SAL_HISTORY` und `MGR_HISTORY` zu erstellen.

1. Die Personalabteilung benötigt eine Liste der Mitarbeiter mit dem niedrigsten Gehalt, eine Gehaltshistorie der Mitarbeiter und eine Gehaltshistorie der Manager auf der Basis einer branchenbezogenen Gehaltsübersicht. Sie wurden daher aufgefordert, die folgenden Aufgaben auszuführen:

Erstellen Sie eine Anweisung, um folgende Aufgaben auszuführen:

- Details wie Personalnummer, Einstellungsdatum, Gehalt und Manager-ID der Mitarbeiter, deren Personalnummer größer oder gleich 200 ist, aus der Tabelle `EMPLOYEES` abrufen
- Für ein Gehalt unter \$ 5.000 Details wie Personalnummer und Gehalt in die Tabelle `SPECIAL_SAL` einfügen
- Personalnummer, Einstellungsdatum und Gehalt in die Tabelle `SAL_HISTORY` einfügen
- Personalnummer, Manager-ID und Gehalt in die Tabelle `MGR_HISTORY` einfügen

```
INSERT ALL
WHEN SAL < 5000 THEN
INTO special_sal VALUES (EMPID, SAL)
ELSE
INTO sal_history VALUES(EMPID, HIREDATE, SAL)
INTO mgr_history VALUES(EMPID, MGR, SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
       salary SAL, manager_id MGR
FROM employees
WHERE employee_id >=200;
```

2. Um die eingefügten Datensätze anzuzeigen, fragen Sie die Tabellen `SPECIAL_SAL`, `SAL_HISTORY` und `MGR_HISTORY` ab.

```
SELECT * FROM special_sal;
SELECT * FROM sal_history;
SELECT * FROM mgr_history;
```

3. Die DBA Nita beauftragt Sie, eine Tabelle mit einem Constraint vom Typ PRIMARY KEY zu erstellen. Der Index soll aber einen anderen Namen aufweisen als das Constraint. Erstellen Sie anhand des folgenden Tabelleninstanzdiagramms die Tabelle LOCATIONS_NAMED_INDEX. Geben Sie dem Index für die Spalte PRIMARY KEY den Namen LOCATIONS_PK_IDX.

Spaltenname	Deptno	Dname
Primärschlüssel	Ja	
Datentyp	Number	VARCHAR2
Länge	4	30

```
CREATE TABLE LOCATIONS_NAMED_INDEX
(location_id NUMBER(4) PRIMARY KEY USING INDEX
(CREATE INDEX locations_pk_idx ON
LOCATIONS_NAMED_INDEX(location_id)),
location_name VARCHAR2(20));
```

4. Fragen Sie die Tabelle USER_INDEXES ab, um INDEX_NAME für die Tabelle LOCATIONS_NAMED_INDEX anzuzeigen.

```
SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'LOCATIONS_NAMED_INDEX';
```

In den folgenden Übungen können Sie zusätzliche praktische Erfahrungen zu den Datetime-Funktionen sammeln.

Sie arbeiten für ein internationales Unternehmen. Der neue Vice President of Operations möchte die verschiedenen Zeitzonen aller Unternehmensniederlassungen wissen. Er hat die folgenden Informationen angefordert:

5. Ändern Sie die Session, und stellen Sie NLS_DATE_FORMAT auf DD-MON-YYYY HH24:MI:SS ein.

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

- 6.
- a. Erstellen Sie Abfragen, um die Zeitzonendifferenzen (TZ_OFFSET) für die folgenden Zeitzonen anzuzeigen:
- Australia/Sydney

```
SELECT TZ_OFFSET ('Australia/Sydney') from dual;
```


- Chile/Easter Island

```
SELECT TZ_OFFSET ('Chile/EasterIsland') from dual;
```

- b. Ändern Sie die Session, und stellen Sie den Wert des Parameters `TIME_ZONE` auf die Zeitzonendifferenz von "Australia/Sydney" ein.

```
ALTER SESSION SET TIME_ZONE = '+10:00';
```

- c. Zeigen Sie `SYSDATE`, `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` für diese Session an.

Hinweis: Die Ausgabe kann entsprechend dem Ausführungsdatum des Befehls variieren.

```
SELECT SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP,
LOCALTIMESTAMP FROM DUAL;
```

- d. Ändern Sie die Session, und stellen Sie den Wert des Parameters `TIME_ZONE` auf die Zeitzonendifferenz von Chile/Easter Island ein.

Hinweis: Die Ergebnisse dieser Aufgabe basieren auf einem anderen Datum und entsprechen in einigen Fällen nicht den Ergebnissen, die die Kursteilnehmer erzielen. Darüber hinaus kann die Zeitzonendifferenz der verschiedenen Länder aufgrund der Sommerzeit variieren.

```
ALTER SESSION SET TIME_ZONE = '-06:00';
```

- e. Zeigen Sie `SYSDATE`, `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` für diese Session an.

Hinweis: Die Ausgabe kann entsprechend dem Ausführungsdatum des Befehls variieren.

```
SELECT SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP,
LOCALTIMESTAMP FROM DUAL;
```

- f. Ändern Sie die Session, und stellen Sie `NLS_DATE_FORMAT` auf `DD-MON-YYYY` ein.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

Hinweis:

- Wie Sie sehen, richten sich bei der vorherigen Aufgabe `CURRENT_DATE`, `CURRENT_TIMESTAMP` und `LOCALTIMESTAMP` jeweils nach der Sessionzeitzone, `SYSDATE` jedoch nicht.
- Die Ergebnisse dieser Aufgabe basieren auf einem anderen Datum und entsprechen in einigen Fällen nicht den Ergebnissen, die die Kursteilnehmer erzielen. Darüber hinaus kann die Zeitzonendifferenz der verschiedenen Länder aufgrund der Sommerzeit variieren.

7. Die Personalabteilung benötigt eine Liste der Mitarbeiter, die im Januar geprüft werden sollen. Sie wurden beauftragt, die folgenden Aufgaben auszuführen:

Erstellen Sie eine Abfrage, mit der Nachname, Einstellungsmonat und Einstellungsdatum der Mitarbeiter angezeigt werden, die, unabhängig vom Einstellungsjahr, im Januar in die Firma eingetreten sind.

```
SELECT last_name, EXTRACT (MONTH FROM HIRE_DATE), HIRE_DATE
FROM employees
WHERE EXTRACT (MONTH FROM HIRE_DATE) = 1;
```

Bei den folgenden Übungen können Sie zusätzliche praktische Erfahrungen zu fortgeschrittenen Unterabfragen sammeln.

8. Der CEO benötigt für die Ausschüttung der Gewinnbeteiligung einen Bericht über die drei Spitzenverdiener im Unternehmen. Sie müssen diese Liste für den CEO erstellen. Erstellen Sie eine Abfrage, um die drei Spitzenverdiener in der Tabelle `EMPLOYEES` anzuzeigen. Zeigen Sie ihre Nachnamen und Gehälter an.

```
SELECT last_name, salary
FROM employees e
WHERE 3 > (SELECT COUNT (*)
FROM employees
WHERE e.salary < salary);
```

9. Die staatlichen Leistungen haben sich im Bundesstaat Kalifornien aufgrund einer Verfügung geändert. Der zuständige Sachbearbeiter bittet Sie, eine Liste der hiervon betroffenen Personen zu erstellen. Erstellen Sie eine Abfrage, um die Personalnummern und Nachnamen der Mitarbeiter anzuzeigen, die in Kalifornien arbeiten.

Tipp: Verwenden Sie skalare Unterabfragen.

```
SELECT employee_id, last_name
FROM employees e
WHERE ((SELECT location_id
FROM departments d
WHERE e.department_id = d.department_id )
IN (SELECT location_id
FROM locations l
WHERE state_province = 'California'));
```

10. Die DBA Nita möchte alte Informationen aus der Datenbank entfernen. Hierzu gehören auch alte Mitarbeiterdatensätze. Sie wurden beauftragt, die folgenden Aufgaben auszuführen:

Erstellen Sie eine Abfrage zum Löschen der ältesten `JOB_HISTORY`-Zeile eines Mitarbeiters. Hierfür muss die Tabelle `JOB_HISTORY` nach dem `MIN(START_DATE)` des Mitarbeiters durchsucht werden. Löschen Sie *nur* die Datensätze der Mitarbeiter, die mindestens zweimal die Tätigkeit gewechselt haben.

Tipp: Verwenden Sie einen korrelierten `DELETE`-Befehl.

```
DELETE FROM job_history JH
WHERE employee_id =
(SELECT employee_id
FROM employees E
WHERE JH.employee_id = E.employee_id
AND START_DATE = (SELECT MIN(start_date)
FROM job_history JH
WHERE JH.employee_id =
E.employee_id)
AND 3 > (SELECT COUNT(*)
FROM job_history JH
```

```
WHERE JH.employee_id =  
      E.employee_id  
  
      GROUP BY EMPLOYEE_ID  
      HAVING COUNT(*) >= 2));
```

11. Der Leiter der Personalabteilung benötigt die vollständigen Mitarbeiterdatensätze für seine jährliche Rede zur Auszeichnung von Mitarbeitern. Er ruft Sie kurz an, damit Sie die Arbeit am Auftrag der DBA einstellen.

Rollen Sie die Transaktion zurück.

```
ROLLBACK;
```

12. Die schleppende Wirtschaftslage zwingt das Management, Kostensenkungsmaßnahmen zu ergreifen. Der CEO möchte die Tätigkeiten mit den höchsten Gehältern im Unternehmen prüfen. Sie müssen diese Liste für den CEO auf der Basis folgender Angaben erstellen.

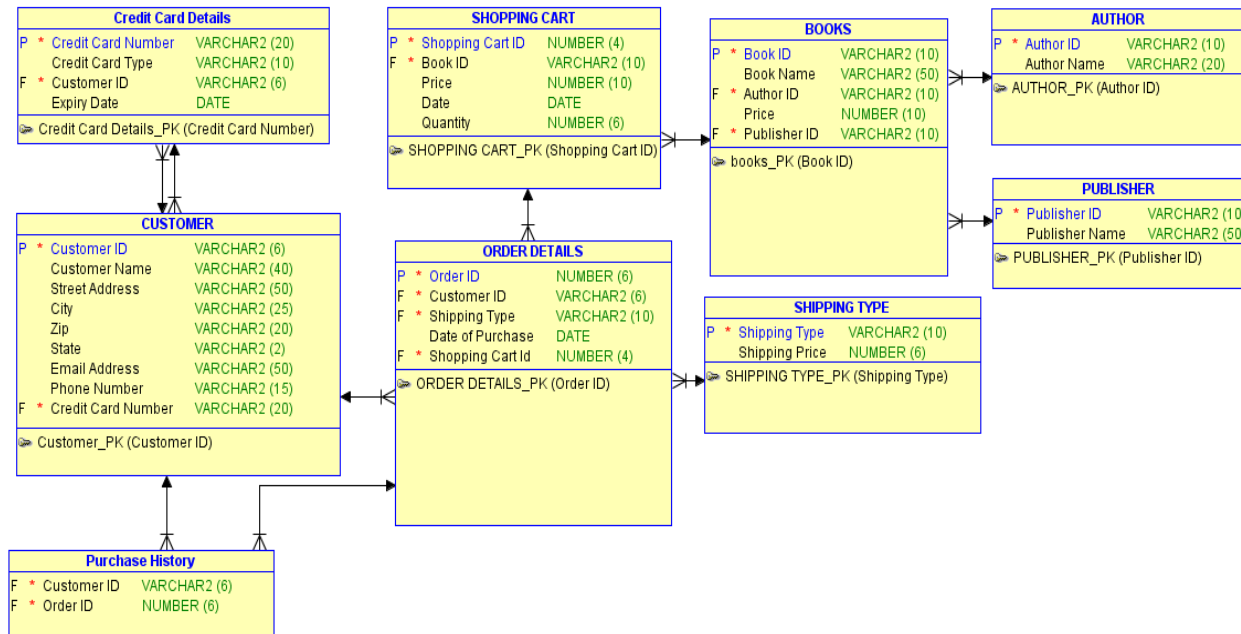
Erstellen Sie eine Abfrage, um die Tätigkeits-IDs der Tätigkeiten anzuzeigen, deren Höchstgehalt höher ist als 50 % des unternehmensweit höchsten Gehalts. Verwenden Sie für diese Abfrage die Klausel `WITH`. Nennen Sie die Abfrage `MAX_SAL_CALC`.

```
WITH  
MAX_SAL_CALC AS (SELECT job_title, MAX(salary) AS  
job_total  
FROM employees, jobs  
WHERE employees.job_id = jobs.job_id  
GROUP BY job_title)  
SELECT job_title, job_total  
FROM MAX_SAL_CALC  
WHERE job_total > (SELECT MAX(job_total) * 1/2  
FROM MAX_SAL_CALC)  
ORDER BY job_total DESC;
```

Zusätzliche Übungen – Fallbeispiel

Im Fallbeispiel für den Kurs *SQL WORKSHOP I* haben Sie mehrere Datenbanktabellen für eine Onlinebuchhandlung erstellt. Darüber hinaus haben Sie Datensätze in die Datenbank einer Onlinebuchhaltung eingefügt, aktualisiert und gelöscht sowie einen Bericht generiert.

Die folgende Abbildung führt die für die Videoanwendung erstellten Tabellen und Spalten auf:



Hinweis: Führen Sie zunächst das Skript `Online_Book_Store_Drop_Tables.sql` im Übungsordner aus, um Tabellen zu löschen, falls sie bereits vorhanden sind. Führen Sie anschließend das Skript `Online_Book_Store_Populate.sql` im Übungsordner aus, um die Tabellen zu erstellen und mit Daten zu füllen.

1. Prüfen Sie, ob die Tabellen korrekt erstellt wurden. Führen Sie hierzu einen Bericht aus, um die Liste der Tabellen und deren Spaltendefinitionen anzuzeigen.





	TABLE_NAME	COLUMN_NAME	DATA_TYPE	NULLABLE
1	AUTHOR	AUTHOR_ID	VARCHAR2	N
2	AUTHOR	AUTHOR_NAME	VARCHAR2	Y
3	BOOKS	BOOK_ID	VARCHAR2	N
4	BOOKS	BOOK_NAME	VARCHAR2	Y
5	BOOKS	AUTHOR_ID	VARCHAR2	N
6	BOOKS	PRICE	NUMBER	Y
7	BOOKS	PUBLISHER_ID	VARCHAR2	N
8	CREDIT_CARD_DETAILS	CREDIT_CARD_NUMBER	VARCHAR2	N
9	CREDIT_CARD_DETAILS	CREDIT_CARD_TYPE	VARCHAR2	Y
10	CREDIT_CARD_DETAILS	EXPIRY_DATE	DATE	Y
11	CUSTOMER	CUSTOMER_ID	VARCHAR2	N
12	CUSTOMER	CUSTOMER_NAME	VARCHAR2	Y
13	CUSTOMER	STREET_ADDRESS	VARCHAR2	Y
14	CUSTOMER	CITY	VARCHAR2	Y
15	CUSTOMER	PHONE_NUMBER	VARCHAR2	Y
16	CUSTOMER	CREDIT_CARD_NUMBER	VARCHAR2	N

17	ORDER_DETAILS	ORDER_ID	VARCHAR2	N
18	ORDER_DETAILS	CUSTOMER_ID	VARCHAR2	Y
19	ORDER_DETAILS	SHIPPING_TYPE	VARCHAR2	N
20	ORDER_DETAILS	DATE_OF_PURCHASE	DATE	Y
21	ORDER_DETAILS	SHOPPING_CART_ID	VARCHAR2	N
22	PUBLISHER	PUBLISHER_ID	VARCHAR2	N
23	PUBLISHER	PUBLISHER_NAME	VARCHAR2	Y
24	PURCHASE_HISTORY	CUSTOMER_ID	VARCHAR2	Y
25	PURCHASE_HISTORY	ORDER_ID	VARCHAR2	N
26	SHIPPING_TYPE	SHIPPING_TYPE	VARCHAR2	N
27	SHIPPING_TYPE	SHIPPING_PRICE	NUMBER	Y
28	SHOPPING_CART	SHOPPING_CART_ID	VARCHAR2	N
29	SHOPPING_CART	BOOK_ID	VARCHAR2	N
30	SHOPPING_CART	PRICE	NUMBER	Y
31	SHOPPING_CART	SHOPPING_CART_DATE	DATE	Y
32	SHOPPING_CART	QUANTITY	NUMBER	Y



2. Prüfen Sie im Data Dictionary, ob die Sequence ORDER_ID_SEQ vorhanden sind.

SEQUENCE_NAME
1 DEPARTMENTS_SEQ
2 EMPLOYEES_SEQ
3 LOCATIONS_SEQ
4 ORDER_ID_SEQ

3. Sie möchten einige Benutzer erstellen, die nur auf die eigene Kaufhistorie Zugriff haben. Erstellen Sie den Benutzer "Carmen", und erteilen Sie ihr die Berechtigung, aus der Tabelle `PURCHASE_HISTORY` wählen zu können.
Hinweis: Stellen Sie dem Benutzernamen Ihren Datenbankaccount voran. Beispiel: Wenn Ihr Benutzername `oraxx` lautet, erstellen Sie den Benutzer `oraxx_Carmen`.
4. Erweitern Sie die Tabelle `BOOKS` um eine Spalte "Edition" (`varchar2 (6)`), in der Informationen zur Buchauflage gespeichert werden sollen.
5. Fügen Sie die Tabelle `CREDIT_CARD_TYPE` hinzu, in der `CREDIT_CARD_TYPE` und `CREDIT_CARD_DESCRIPTION` gespeichert werden. Die Tabelle hat einen Fremdschlüssel mit der Spalte `CREDIT_CARD_TYPE` in der Tabelle `CREDIT_CARD_DETAILS`.
6. Wählen Sie alle Tabellen aus dem Data Dictionary.
7. Erstellen Sie die Tabelle `SHOPPING_HISTORY`, in der die Details der Kaufhistorie der Kunden gespeichert werden sollen.
(**Tipp:** Sie können die Tabelle `PURCHASE_HISTORY` kopieren.)
8. Zeigen Sie die Kundendetails der zehn Kunden an, die im letzten Monat Aufträge erteilt haben. Sortieren Sie die Datensätze nach der Kunden-ID.

	 CUSTOMER_ID	 ORDER_ID	 DATE_OF_PURCHASE	 CUSTOMER_NAME
1	CN0001	OD0001	12-JUN-01	VelasquezCarmen
2	CN0003	OD0003	31-JUL-05	Nagayama Midori
3	CN0004	OD0004	14-AUG-06	Quick-To-See Mark
4	CN0009	OD0009	25-NOV-09	Catchpole Antoinette

9. Zeigen Sie eine Liste von Kunden, die mehr als einen Auftrag erteilt haben.

	 CUSTOMER_ID	 CUSTOMER_NAME
1	CN0001	VelasquezCarmen
2	CN0003	Nagayama Midori
3	CN0004	Quick-To-See Mark
4	CN0009	Catchpole Antoinette

Zusätzliche Übungen – Lösungen: Fallbeispiel

Lösung

Führen Sie zunächst das Skript `Online_Book_Store_Drop_Tables.sql` im Übungsordner aus, um bereits vorhandene Tabellen zu löschen. Führen Sie anschließend das Skript `Online_Book_Store_Populate.sql` im Übungsordner aus, um die Tabellen zu erstellen und mit Daten zu füllen.

1. Prüfen Sie, ob die Tabellen korrekt erstellt wurden. Führen Sie hierzu einen Bericht aus, um die Liste der Tabellen und deren Spaltendefinitionen anzuzeigen.

```
SELECT table_name, column_name, data_type, nullable
FROM user_tab_columns
WHERE table_name
IN('CUSTOMER', 'CREDIT_CARD_DETAILS', 'SHOPPING_CART',
'ORDER_DETAILS', 'BOOKS', 'AUTHOR', 'PUBLISHER', 'SHIPPING_TYPE',
'PURCHASE_HISTORY');
```

2. Prüfen Sie im Data Dictionary, ob die `ORDER_ID_SEQ`-Sequences vorhanden sind.

```
SELECT sequence_name FROM user_sequences;
```

3. Sie möchten einige Benutzer erstellen, die nur auf die eigene Kaufhistorie Zugriff haben. Erstellen Sie den Benutzer "Carmen", und erteilen Sie ihr die Berechtigung, aus der Tabelle `PURCHASE_HISTORY` wählen zu können.

Hinweis: Stellen Sie dem Benutzernamen Ihren Datenbankaccount voran. Beispiel: Wenn Ihr Benutzername `oraxx` lautet, erstellen Sie den Benutzer `oraxx_Carmen`.

```
CREATE USER oraxx_carmen IDENTIFIED BY oracle ;
GRANT select ON purchase_history TO oraxx_carmen;
```

4. Erweitern Sie die Tabelle `BOOKS` um eine Spalte "Edition" (`varchar2(6)`), in der Informationen zur Buchauflage gespeichert werden sollen.

```
ALTER TABLE books ADD(edition VARCHAR2(6));
```

5. Fügen Sie die Tabelle `CREDIT_CARD_TYPE` hinzu, in der `CREDIT_CARD_TYPE` und `CREDIT_CARD_DESCRIPTION` gespeichert werden. Die Tabelle hat einen Fremdschlüssel mit der Spalte `CREDIT_CARD_TYPE` in der Tabelle `CREDIT_CARD_DETAILS`.

```
CREATE TABLE CREDIT_CARD_TYPE
( CREDIT_CARD_TYPE VARCHAR2(10) NOT NULL ENABLE,
  CREDIT_CARD_DESCRIPTION VARCHAR2(4000 BYTE),
  CONSTRAINT CREDIT_CARD_TYPE_PK PRIMARY KEY
  (CREDIT_CARD_TYPE) )
;
```

6. Wählen Sie alle Tabellen aus dem Data Dictionary.

```
SELECT table_name FROM user_tables order by table_name;
```

7. Erstellen Sie die Tabelle SHOPPING_HISTORY, in der Details zur Kaufhistorie von Kunden gespeichert werden sollen.

(**Tipp:** Sie können die Tabelle PURCHASE_HISTORY kopieren.)

```
CREATE TABLE shopping_history as select * from purchase_history
where '1' = '1';
```

8. Zeigen Sie die Kundendetails der zehn Kunden an, die im letzten Monat Aufträge erteilt haben. Sortieren Sie die Datensätze nach der Kunden-ID.

```
SELECT o.CUSTOMER_ID, o.ORDER_ID, o.DATE_OF_PURCHASE,
c.CUSTOMER_NAME
FROM ORDER_DETAILS o JOIN PURCHASE_HISTORY p
ON o.CUSTOMER_ID = p.CUSTOMER_ID JOIN CUSTOMER c
ON o.CUSTOMER_ID= c.CUSTOMER_ID
AND rownum < 10
ORDER BY CUSTOMER_ID;
```

9. Zeigen Sie eine Liste von Kunden, die mehr als einen Auftrag erteilt haben.

```
SELECT customer_id, customer_name FROM customer c
WHERE 1 <= (select count(*) from purchase_history where
customer_id = c.customer_id);
```