



Hardware and Software
Engineered to Work Together

Oracle Database 12c: SQL Workshop I

Schulungsunterlagen – Band II

D80190DE11

Production 1.1 | Dezember 2014 | D88603

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

Autor
Dimpi Rani Sarmah

**Technischer Inhalt und
Überarbeitung**

Nancy Greenberg
Swarnapriya Shridhar
Bryan Roberts
Laszlo Czinkoczki
KimSeong Loh
Brent Dayley
Jim Spiller
Christopher Wensley
Manish Pawar
Clair Bennett
Yanti Chang
Joel Goodman
Gerlinde Frenzen
Madhavi Siddireddy

Redaktion
Smita Kommini
Aju Kumar

Herausgeber
Pavithran Adka
Giri Venugopal

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.

Inhaltsverzeichnis

1 Einführung

Ziele 1-2

Lektionsagenda 1-3

Kursziele 1-4

Kursagenda 1-5

In diesem Kurs verwendete Anhänge und Übungen 1-7

Lektionsagenda 1-8

Oracle Database 12c: Kernbereiche 1-9

Oracle Database 12c 1-10

Oracle Fusion Middleware 1-12

Oracle Cloud 1-14

Oracle Cloud-Services 1-15

Cloud-Deployment-Modelle 1-16

Lektionsagenda 1-17

Relationale und objektrelationale Datenbankmanagementsysteme 1-18

Datenspeicherung auf verschiedenen Medien 1-19

Relationale Datenbanken – Konzept 1-20

Relationale Datenbanken – Definition 1-21

Datenmodelle 1-22

Entity-Relationship-Modell 1-23

Entity-Relationship-Modellierung – Konventionen 1-25

Beziehungen zwischen mehreren Tabellen 1-27

Relationale Datenbanken – Terminologie 1-29

Lektionsagenda 1-31

Datenbanken mit SQL abfragen 1-32

Im Kurs verwendete SQL-Anweisungen 1-33

SQL-Entwicklungsumgebungen 1-34

Lektionsagenda 1-35

Schema Human Resources (HR) 1-36

In diesem Kurs verwendete Tabellen 1-37

Lektionsagenda 1-38

Oracle Database – Dokumentation 1-39

Zusätzliche Ressourcen 1-40

Zusammenfassung 1-41

Übungen zu Lektion 1 – Überblick 1-42

2 Daten mit der SQL SELECT-Anweisung abrufen

Ziele 2-2

Lektionsagenda 2-3

Einfache SELECT-Anweisungen 2-4

Alle Spalten wählen 2-5

Bestimmte Spalten wählen 2-6

SQL-Anweisungen schreiben 2-7

Standardwerte für Spaltenüberschriften 2-8

Lektionsagenda 2-9

Arithmetische Ausdrücke 2-10

Arithmetische Operatoren 2-11

Operatorpriorität 2-12

Nullwerte definieren 2-13

Nullwerte in arithmetischen Ausdrücken 2-14

Lektionsagenda 2-15

Spaltenaliasnamen definieren 2-16

Spaltenaliasnamen 2-17

Lektionsagenda 2-18

Verkettungsoperator 2-19

Literale Zeichenfolgen 2-20

Literale Zeichenfolgen – Beispiel 2-21

Operator α für alternative Anführungszeichen 2-22

Mehrfach vorhandene Zeilen 2-23

Lektionsagenda 2-24

Tabellenstrukturen anzeigen 2-25

Befehl DESCRIBE 2-26

Quiz 2-27

Zusammenfassung 2-28

Übungen zu Lektion 2 – Überblick 2-29

3 Daten einschränken und sortieren

Ziele 3-2

Lektionsagenda 3-3

Zeilen durch Auswahl begrenzen 3-4

Auszugebende Zeilen einschränken 3-5

Klausel WHERE 3-6

Zeichenfolgen und Datumsangaben 3-7

Vergleichsoperatoren 3-8

Vergleichsoperatoren – Beispiel 3-9

Bereichsbedingungen mit dem Operator <code>BETWEEN</code>	3-10
Operator <code>IN</code> – Beispiel	3-11
Muster mit dem Operator <code>LIKE</code> vergleichen	3-12
Platzhalterzeichen kombinieren	3-13
<code>NULL</code> -Bedingungen	3-14
Bedingungen mit logischen Operatoren definieren	3-15
Operator <code>AND</code>	3-16
Operator <code>OR</code>	3-17
Operator <code>NOT</code>	3-18
Lektionsagenda	3-19
Prioritätsregeln	3-20
Lektionsagenda	3-22
Klausel <code>ORDER BY</code>	3-23
Daten sortieren	3-24
Lektionsagenda	3-26
SQL-Klausel zur Zeilenbeschränkung	3-27
SQL-Klausel zur Zeilenbeschränkung in Abfragen	3-28
SQL-Klausel zur Zeilenbeschränkung – Beispiel	3-29
Lektionsagenda	3-30
Substitutionsvariablen	3-31
Substitutionsvariable <code>&</code>	3-33
Zeichen- und Datumswerte mit Substitutionsvariablen	3-35
Spaltennamen, Ausdrücke und Text angeben	3-36
Substitutionsvariable <code>&&</code>	3-37
Substitutionsvariablen in SQL*Plus	3-38
Lektionsagenda	3-39
Befehl <code>DEFINE</code>	3-40
Befehl <code>VERIFY</code>	3-41
Quiz	3-42
Zusammenfassung	3-43
Übungen zu Lektion 3 – Überblick	3-44

4 Ausgabe mit Single-Row-Funktionen anpassen

Ziele	4-2
Lektionsagenda	4-3
SQL-Funktionen	4-4
Zwei Typen von SQL-Funktionen	4-5
Single-Row-Funktionen	4-6
Lektionsagenda	4-8
Zeichenfunktionen	4-9

Funktionen zur Umwandlung der Groß-/Kleinschreibung von Zeichenfolgen 4-11
Funktionen zur Umwandlung der Groß-/Kleinschreibung von Zeichenfolgen –
 Beispiel 4-12
Funktionen zum Bearbeiten von Zeichen 4-13
Funktionen zum Bearbeiten von Zeichen – Beispiel 4-14
Lektionsagenda 4-15
Funktionen verschachteln 4-16
Funktionen verschachteln – Beispiel 4-17
Lektionsagenda 4-18
Numerische Funktionen 4-19
Funktion ROUND 4-20
Funktion TRUNC 4-21
Funktion MOD 4-22
Lektionsagenda 4-23
Mit Datumswerten arbeiten 4-24
Datumsformat RR 4-25
Funktion SYSDATE 4-27
Funktionen CURRENT_DATE und CURRENT_TIMESTAMP 4-28
Mit Datumswerten rechnen 4-29
Arithmetische Operatoren für Datumswerte 4-30
Lektionsagenda 4-31
Funktionen zum Bearbeiten von Datumswerten 4-32
Datumsfunktionen 4-33
Funktionen ROUND und TRUNC für Datumswerte 4-34
Quiz 4-35
Zusammenfassung 4-36
Übungen zu Lektion 4 – Überblick 4-37

5 Konvertierungsfunktionen und bedingte Ausdrücke

Ziele 5-2
Lektionsagenda 5-3
Konvertierungsfunktionen 5-4
Implizite Datentypkonvertierung 5-5
Explizite Datentypkonvertierung 5-7
Lektionsagenda 5-9
Funktion TO_CHAR in Verbindung mit Datumswerten 5-10
Datumsformatmasken – Elemente 5-11
Funktion TO_CHAR in Verbindung mit Datumswerten 5-14
Funktion TO_CHAR in Verbindung mit Zahlenwerten 5-15
Funktionen TO_NUMBER und TO_DATE 5-18

Funktionen `TO_CHAR` und `TO_DATE` in Verbindung mit dem Datumsformat `RR` 5-20

Lektionsagenda 5-21

Allgemeine Funktionen 5-22

Funktion `NVL` 5-23

Funktion `NVL` – Beispiel 5-24

Funktion `NVL2` – Beispiel 5-25

Funktion `NULLIF` – Beispiel 5-26

Funktion `COALESCE` – Beispiel 5-27

Lektionsagenda 5-29

Bedingte Ausdrücke 5-30

`CASE`-Ausdrücke 5-31

`CASE`-Ausdrücke – Beispiel 5-32

Searched `CASE`-Ausdrücke 5-33

Funktion `DECODE` 5-34

Funktion `DECODE` – Beispiel 5-35

Quiz 5-37

Zusammenfassung 5-38

Übungen zu Lektion 5 – Überblick 5-39

6 Mit Gruppenfunktionen Berichte aus aggregierten Daten erstellen

Ziele 6-2

Lektionsagenda 6-3

Gruppenfunktionen 6-4

Typen von Gruppenfunktionen – Wiederholung 6-5

Gruppenfunktionen – Syntax 6-6

Funktionen `AVG` und `SUM` 6-7

Funktionen `MIN` und `MAX` 6-8

Funktion `COUNT` 6-9

Schlüsselwort `DISTINCT` 6-10

Gruppenfunktionen und Nullwerte 6-11

Lektionsagenda 6-12

Datengruppen erstellen 6-13

Datengruppen erstellen – Syntax der Klausel `GROUP BY` 6-14

Klausel `GROUP BY` 6-15

Nach mehreren Spalten gruppieren 6-17

Klausel `GROUP BY` über mehrere Spalten 6-18

Unzulässige Abfragen mit Gruppenfunktionen 6-19

Gruppenergebnisse einschränken 6-21

Gruppenergebnisse mit der Klausel `HAVING` einschränken 6-22

Klausel `HAVING` 6-23
Lektionsagenda 6-25
Gruppenfunktionen verschachteln 6-26
Quiz 6-27
Zusammenfassung 6-28
Übungen zu Lektion 6 – Überblick 6-29

7 Daten aus mehreren Tabellen mit Joins anzeigen

Ziele 7-2
Lektionsagenda 7-3
Daten aus mehreren Tabellen abrufen 7-4
Typen von Joins 7-5
Tabellen mithilfe der SQL:1999-Syntax verknüpfen 7-6
Lektionsagenda 7-7
Natural Joins erstellen 7-8
Datensätze mit Natural Joins abrufen 7-9
Joins mit der Klausel `USING` erstellen 7-10
Spaltennamen verknüpfen 7-11
Datensätze mit der Klausel `USING` abrufen 7-12
Mehrdeutige Spaltennamen eindeutig kennzeichnen 7-13
Tabellenaliasnamen mit der Klausel `USING` 7-14
Joins mit der Klausel `ON` erstellen 7-15
Datensätze mit der Klausel `ON` abrufen 7-16
3-Way Joins erstellen 7-17
Zusätzliche Bedingungen in Joins anwenden 7-18
Lektionsagenda 7-19
Tabellen mit sich selbst verknüpfen 7-20
Self Joins mit der Klausel `ON` 7-21
Lektionsagenda 7-22
Non Equij Joins 7-23
Datensätze mit Non-Equi Joins abrufen 7-24
Lektionsagenda 7-25
Mit `OUTER`-Joins Datensätze ohne direkte Übereinstimmung zurückgeben 7-26
`INNER`- und `OUTER`-Joins – Vergleich 7-27
`LEFT OUTER JOINS` 7-28
`RIGHT OUTER JOINS` 7-29
`FULL OUTER JOINS` 7-30
Lektionsagenda 7-31
Kartesische Produkte 7-32
Kartesische Produkte generieren 7-33

Cross Joins erstellen 7-34
Quiz 7-35
Zusammenfassung 7-36
Übungen zu Lektion 7 – Überblick 7-37

8 Unterabfragen in Abfragen

Ziele 8-2
Lektionsagenda 8-3
Problemstellungen mit Unterabfragen lösen 8-4
Unterabfragen – Syntax 8-5
Unterabfragen 8-6
Unterabfragen – Regeln und Richtlinien 8-7
Typen von Unterabfragen 8-8
Lektionsagenda 8-9
Single-Row-Unterabfragen 8-10
Single-Row-Unterabfragen ausführen 8-11
Gruppenfunktionen in Unterabfragen – Beispiel 8-12
Unterabfragen in `HAVING`-Klauseln 8-13
Wo liegt der Fehler in dieser Anweisung? 8-14
Innere Abfrage gibt keine Zeilen zurück 8-15
Lektionsagenda 8-16
Multiple-Row-Unterabfragen 8-17
Operator `ANY` in Multiple-Row-Unterabfragen – Beispiel 8-18
Operator `ALL` in Multiple-Row-Unterabfragen – Beispiel 8-19
Multiple-Column-Unterabfragen 8-20
Multiple-Column-Unterabfragen – Beispiel 8-21
Lektionsagenda 8-22
Nullwerte in Unterabfragen 8-23
Quiz 8-25
Zusammenfassung 8-26
Übungen zu Lektion 8 – Überblick 8-27

9 Mengenoperatoren

Ziele 9-2
Lektionsagenda 9-3
Mengenoperatoren 9-4
Mengenoperatoren – Richtlinien 9-5
Oracle-Server und Mengenoperatoren 9-6
Lektionsagenda 9-7
In dieser Lektion verwendete Tabellen 9-8

Lektionsagenda 9-12
Operator UNION 9-13
Operator UNION – Beispiel 9-14
Operator UNION ALL 9-15
Operator UNION ALL – Beispiel 9-16
Lektionsagenda 9-17
Operator INTERSECT 9-18
Operator INTERSECT – Beispiel 9-19
Lektionsagenda 9-20
Operator MINUS 9-21
Operator MINUS – Beispiel 9-22
Lektionsagenda 9-23
SELECT-Anweisungen abgleichen 9-24
SELECT-Anweisungen abgleichen – Beispiel 9-25
Lektionsagenda 9-26
ORDER BY-Klauseln in Mengenoperationen 9-27
Quiz 9-28
Zusammenfassung 9-29
Übungen zu Lektion 9 – Überblick 9-30

10 Tabellen mit DML-Anweisungen verwalten

Ziele 10-2
Lektionsagenda 10-3
Data Manipulation Language 10-4
Tabellen neue Zeilen hinzufügen 10-5
Anweisung INSERT – Syntax 10-6
Neue Zeilen einfügen 10-7
Zeilen mit Nullwerten einfügen 10-8
Spezielle Werte einfügen 10-9
Spezielle Datums- und Uhrzeitwerte einfügen 10-10
Skripte erstellen 10-11
Zeilen aus anderen Tabellen kopieren 10-12
Lektionsagenda 10-13
Daten in Tabellen ändern 10-14
UPDATE-Anweisungen – Syntax 10-15
Zeilen einer Tabelle aktualisieren 10-16
Zwei Spalten mithilfe einer Unterabfrage aktualisieren 10-17
Zeilen auf Basis einer anderen Tabelle aktualisieren 10-18
Lektionsagenda 10-19
Zeilen aus Tabellen entfernen 10-20

DELETE-Anweisungen	10-21
Zeilen aus Tabellen löschen	10-22
Zeilen auf Basis einer anderen Tabelle löschen	10-23
TRUNCATE-Anweisungen	10-24
Lektionsagenda	10-25
Datenbanktransaktionen	10-26
Datenbanktransaktionen – Beginn und Ende	10-27
COMMIT- und ROLLBACK-Anweisungen – Vorteile	10-28
Explizite Steueranweisungen für Transaktionen	10-29
Änderungen bis zu einer Markierung zurückrollen	10-30
Implizite Transaktionsverarbeitung	10-31
Zustand der Daten vor COMMIT oder ROLLBACK	10-33
Zustand der Daten nach COMMIT	10-34
Daten festschreiben	10-35
Zustand der Daten nach ROLLBACK	10-36
Zustand der Daten nach ROLLBACK – Beispiel	10-37
Rollbacks auf Anweisungsebene	10-38
Lektionsagenda	10-39
Lesekonsistenz	10-40
Lesekonsistenz implementieren	10-41
Lektionsagenda	10-42
Klausel FOR UPDATE in SELECT-Anweisungen	10-43
Klausel FOR UPDATE – Beispiele	10-44
Quiz	10-46
Zusammenfassung	10-47
Übungen zu Lektion 10 – Überblick	10-48

11 Data Definition Language – Einführung

Ziele	11-2
Lektionsagenda	11-3
Datenbankobjekte	11-4
Benennungsregeln	11-5
Lektionsagenda	11-6
CREATE TABLE-Anweisungen	11-7
Tabellen erstellen	11-8
Lektionsagenda	11-9
Datetime-Datentypen	11-12
Option DEFAULT	11-13
Lektionsagenda	11-14
Constraints hinzufügen	11-15

Constraints – Richtlinien	11-16
Constraints definieren	11-17
Constraint NOT NULL	11-19
Constraint UNIQUE	11-20
Constraint PRIMARY KEY	11-22
Constraint FOREIGN KEY	11-23
Constraint FOREIGN KEY – Schlüsselwörter	11-25
Constraint CHECK	11-26
CREATE TABLE – Beispiel	11-27
Constraints verletzen	11-28
Lektionsagenda	11-30
Tabellen mithilfe von Unterabfragen erstellen	11-31
Lektionsagenda	11-33
ALTER TABLE-Anweisungen	11-34
Spalten hinzufügen	11-36
Spalten ändern	11-37
Spalten löschen	11-38
Option SET UNUSED	11-39
Schreibgeschützte Tabellen	11-41
Lektionsagenda	11-42
Tabellen löschen	11-43
Quiz	11-44
Zusammenfassung	11-45
Übungen zu Lektion 11 – Überblick	11-46

A Tabellenbeschreibungen

B SQL Developer

Ziele	B-2
Was ist Oracle SQL Developer?	B-3
SQL Developer – Spezifikationen	B-4
SQL Developer 3.2 – Benutzeroberfläche	B-5
Datenbankverbindungen erstellen	B-7
Datenbankobjekte durchsuchen	B-10
Tabellenstrukturen anzeigen	B-11
Dateien durchsuchen	B-12
Schemaobjekte erstellen	B-13
Neue Tabellen erstellen – Beispiel	B-14
SQL Worksheet	B-15
SQL-Anweisungen ausführen	B-19

SQL-Skripte speichern B-20
Gespeicherte Skriptdateien ausführen – Methode 1 B-21
Gespeicherte Skriptdateien ausführen – Methode 2 B-22
SQL-Code formatieren B-23
Snippets B-24
Snippets – Beispiel B-25
Papierkorb B-26
Prozeduren und Funktionen debuggen B-27
Datenbankberichte B-28
Benutzerdefinierte Berichte erstellen B-29
Suchmaschinen und externe Tools B-30
Voreinstellungen festlegen B-31
SQL Developer-Layout zurücksetzen B-33
Data Modeler in SQL Developer B-34
Zusammenfassung B-35

C SQL*Plus

Ziele C-2
SQL und SQL*Plus – Interaktion C-3
SQL-Anweisungen und SQL*Plus-Befehle – Vergleich C-4
SQL*Plus – Überblick C-5
Bei SQL*Plus anmelden C-6
Tabellenstrukturen anzeigen C-7
SQL*Plus – Bearbeitungsbefehle C-9
LIST, n und APPEND C-11
Befehl CHANGE C-12
SQL*Plus – Dateibefehle C-13
Befehle SAVE und START C-14
Befehl SERVEROUTPUT C-15
SQL*Plus-Befehl SPOOL C-16
Befehl AUTOTRACE C-17
Zusammenfassung C-18

D Häufig verwendete SQL-Befehle

Ziele D-2
Einfache SELECT-Anweisungen D-3
SELECT-Anweisungen D-4
WHERE-Klauseln D-5
ORDER BY-Klauseln D-6
GROUP BY-Klauseln D-7

Data Definition Language	D-8
CREATE TABLE-Anweisungen	D-9
ALTER TABLE-Anweisungen	D-10
DROP TABLE-Anweisungen	D-11
GRANT-Anweisungen	D-12
Typen von Berechtigungen	D-13
REVOKE-Anweisungen	D-14
TRUNCATE TABLE-Anweisungen	D-15
Data Manipulation Language	D-16
INSERT-Anweisungen	D-17
UPDATE-Anweisungen – Syntax	D-18
DELETE-Anweisungen	D-19
Anweisungen zur Transaktionskontrolle	D-20
COMMIT-Anweisungen	D-21
ROLLBACK-Anweisungen	D-22
SAVEPOINT-Anweisungen	D-23
Joins	D-24
Typen von Joins	D-25
Mehrdeutige Spaltennamen eindeutig kennzeichnen	D-26
Natural Joins	D-27
Equi Joins	D-28
Datensätze mit Equi Joins abrufen	D-29
Zusätzliche Suchbedingungen mit den Operatoren AND und WHERE	D-30
Datensätze mit Non-Equi Joins abrufen	D-31
Datensätze mit USING-Klauseln abrufen	D-32
Datensätze mit ON-Klauseln abrufen	D-33
Left Outer Joins	D-34
Right Outer Joins	D-35
Full Outer Joins	D-36
Self Joins – Beispiel	D-37
Cross Joins	D-38
Zusammenfassung	D-39

9 Mengenoperatoren

ORACLE®

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

Ziele

Nach Ablauf dieser Lektion haben Sie folgende Ziele erreicht:

- Mengenoperatoren beschreiben
- Mehrere Abfragen mit einem Mengenoperator zu einer einzelnen Abfrage zusammenfassen
- Reihenfolge der zurückgegebenen Zeilen steuern

ORACLE

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

In dieser Lektion wird beschrieben, wie Sie Abfragen erstellen, die Mengenoperatoren enthalten.

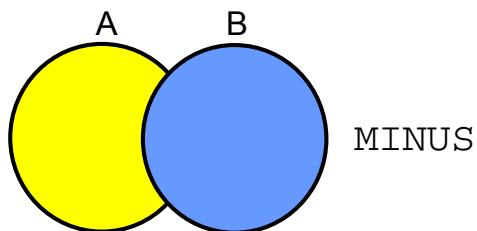
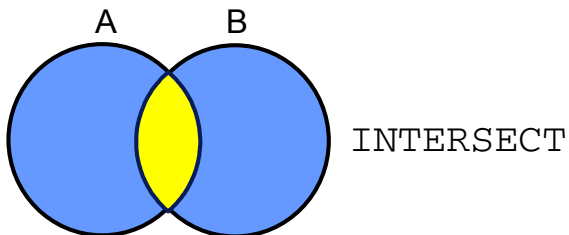
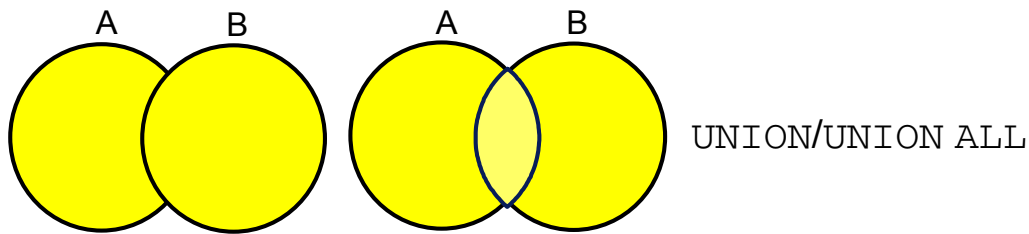
Lektionsagenda

- Mengenoperatoren – Typen und Richtlinien
- In dieser Lektion verwendete Tabellen
- Operatoren UNION und UNION ALL
- Operator INTERSECT
- Operator MINUS
- SELECT-Anweisungen abgleichen
- Klausel ORDER BY in Mengenoperationen

ORACLE®

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

Mengenoperatoren



ORACLE

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

Mengenoperatoren fassen die Ergebnisse aus zwei oder mehr Teilabfragen zu einem Ergebnis zusammen. Abfragen, die Mengenoperatoren enthalten, werden als *zusammengesetzte Abfragen* bezeichnet.

Operator	Rückgabe
UNION	Zeilen aus beiden Abfragen. Doppelte Zeilen wurde zuvor entfernt.
UNION ALL	Zeilen aus beiden Abfragen einschließlich aller mehrfach vorhandenen Zeilen
INTERSECT	Nur Zeilen, die in beiden Abfragen vorkommen
MINUS	Zeilen in der ersten Abfrage, die in der zweiten Abfrage nicht vorkommen

Alle Mengenoperatoren haben die gleiche Priorität. Der Oracle-Server wertet mehrere Mengenoperatoren in SQL-Anweisungen von links (oben) nach rechts (unten) aus, sofern nicht explizit durch Klammern eine andere Reihenfolge festgelegt wird. In Abfragen, in denen der Operator `INTERSECT` zusammen mit anderen Mengenoperatoren verwendet wird, sollten Sie die Auswertungsreihenfolge explizit durch Klammern angeben.

Mengenoperatoren – Richtlinien

- Die `SELECT`-Listen müssen dieselbe Anzahl von Ausdrücken enthalten.
- Die für die entsprechenden Spalten der ersten und nachfolgenden Abfragen verwendeten Datentypen müssen übereinstimmen.
- Die Ausführungsreihenfolge kann mit Klammern geändert werden.
- Die Klausel `ORDER BY` muss am Ende der Anweisung stehen.

ORACLE

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

- Die Ausdrücke in den `SELECT`-Listen der Abfragen müssen in Anzahl und Datentyp übereinstimmen. In Abfragen, in denen die Operatoren `UNION`, `UNION ALL`, `INTERSECT` und `MINUS` verwendet werden, müssen die jeweiligen `SELECT`-Listen dieselbe Anzahl von Spalten mit jeweils demselben Datentyp enthalten. In zusammengesetzten Abfragen müssen die Datentypen der entsprechenden Spalten in den `SELECT`-Listen der Teilabfragen nicht exakt gleich sein. Die Spalte in der zweiten Abfrage muss jedoch der gleichen Datentypgruppe (beispielsweise `NUMERIC` oder `CHARACTER`) angehören wie die entsprechende Spalte in der ersten Abfrage.
- Mengenoperatoren können in Unterabfragen verwendet werden.
- In Abfragen, in denen der Operator `INTERSECT` zusammen mit anderen Mengenoperatoren verwendet wird, sollten Sie die Auswertungsreihenfolge durch Klammern festlegen. Dies gewährleistet die Konformität mit neuen SQL-Standards, die dem Operator `INTERSECT` eine höhere Priorität als den anderen Mengenoperatoren zuweisen.

Oracle-Server und Mengenoperatoren

- Alle Operatoren außer `UNION ALL` entfernen mehrfach vorhandene Zeilen automatisch.
- Im Ergebnis werden die Spaltennamen aus der ersten Abfrage angezeigt.
- Die Ausgabe wird standardmäßig in aufsteigender Reihenfolge sortiert (außer bei `UNION ALL`).

ORACLE

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

In Abfragen mit Mengenoperatoren werden mehrfach vorhandene Zeilen vom Oracle-Server automatisch entfernt. (Dies gilt nicht für den Operator `UNION ALL`.) Welche Spaltennamen in der Ausgabe angezeigt werden, richtet sich nach der Spaltenliste in der ersten `SELECT`-Anweisung. Standardmäßig wird die Ausgabe in aufsteigender Reihenfolge nach der ersten Spalte der Klausel `SELECT` sortiert.

In zusammengesetzten Abfragen müssen Anzahl und Datentyp der in den `SELECT`-Listen der jeweiligen Teilabfragen enthaltenen Ausdrücke übereinstimmen. Werden durch die Teilabfragen Zeichendaten gewählt, ergibt sich der Datentyp der zurückgegebenen Werte wie folgt:

- Wählen beide Abfragen Werte gleicher Länge vom Datentyp `CHAR` aus, haben die Rückgabewerte den Datentyp `CHAR` und die entsprechende Länge. Wählen die Abfragen Werte unterschiedlicher Länge vom Datentyp `CHAR`, weist der Rückgabewert den Datentyp `VARCHAR2` und die Länge des längeren `CHAR`-Wertes auf.
- Wenn eine oder beide Abfragen Werte vom Datentyp `VARCHAR2` wählen, weisen die zurückgegebenen Werte den Datentyp `VARCHAR2` auf.

In Teilabfragen, die numerische Daten wählen, richtet sich der Datentyp der zurückgegebenen Werte nach dem numerischen Vorrang. Wählen alle Abfragen Werte vom Datentyp `NUMBER`, weisen die Rückgabewerte ebenfalls den Datentyp `NUMBER` auf. In Abfragen mit Mengenoperatoren werden keine impliziten Konvertierungen zwischen den Datentypgruppen durchgeführt. Daher gibt der Oracle-Server einen Fehler zurück, wenn die entsprechenden Ausdrücke der Teilabfragen sowohl Zeichendaten als auch numerische Daten ergeben.

Lektionsagenda

- Mengenoperatoren – Typen und Richtlinien
- **In dieser Lektion verwendete Tabellen**
- Operatoren UNION und UNION ALL
- Operator INTERSECT
- Operator MINUS
- SELECT-Anweisungen abgleichen
- Klausel ORDER BY in Mengenoperationen

ORACLE®

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

In dieser Lektion verwendete Tabellen

In dieser Lektion werden folgende Tabellen verwendet:

- `EMPLOYEES`: Enthält Details zu allen aktuellen Mitarbeitern
- `RETIRED_EMPLOYEES`: Enthält Details zu allen früheren Mitarbeitern

ORACLE

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

In dieser Lektion werden zwei Tabellen verwendet: `EMPLOYEES` und `RETIRED_EMPLOYEES`.

Sie kennen bereits die Tabelle `EMPLOYEES`, in der Mitarbeiterdetails wie eindeutige Personalnummer, E-Mail-Adresse, Tätigkeits-ID (`ST_CLERK`, `SA_REP` usw.), Gehalt und Manager gespeichert werden.

`RETIRED_EMPLOYEES` enthält die Details der Mitarbeiter, die das Unternehmen verlassen haben.

Struktur und Daten der Tabellen `EMPLOYEES` und `RETIRED_EMPLOYEES` werden auf den nächsten Seiten vorgestellt.

DESCRIBE employees

DESCRIBE employees		
Name	Null	Type

EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

```
SELECT employee_id, last_name, job_id, hire_date, department_id
FROM employees;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
1	100	King	AD_PRES	17-JUN-03	90
2	101	Kochhar	AD_VP	21-SEP-05	90
3	102	De Haan	AD_VP	13-JAN-01	90
4	103	Hunold	IT_PROG	03-JAN-06	60
5	104	Ernst	IT_PROG	21-MAY-07	60
6	107	Lorentz	IT_PROG	07-FEB-07	60
7	124	Mourgos	ST_MAN	16-NOV-07	50
8	141	Rajs	ST_CLERK	17-OCT-03	50
9	142	Davies	ST_CLERK	29-JAN-05	50
10	143	Matos	ST_CLERK	15-MAR-06	50
11	144	Vargas	ST_CLERK	09-JUL-06	50
12	149	Zlotkey	SA_MAN	29-JAN-08	80
13	174	Abel	SA_REP	11-MAY-04	80
14	176	Taylor	SA_REP	24-MAR-06	80
15	178	Grant	SA_REP	24-MAY-07	(null)
16	200	Whalen	AD_ASST	17-SEP-03	10
17	201	Hartstein	MK_MAN	17-FEB-04	20
18	202	Fay	MK_REP	17-AUG-05	20
19	205	Higgins	AC_MGR	07-JUN-02	110
20	206	Gietz	AC_ACCOUNT	07-JUN-02	110

```
DESCRIBE retired_employees
```

Name	Null	Type
EMPLOYEE_ID		NUMBER(7)
FIRST_NAME		VARCHAR2(20)
LAST_NAME		VARCHAR2(20)
EMAIL		VARCHAR2(25)
RETIRED_DATE		DATE
JOB_ID		VARCHAR2(20)
SALARY		NUMBER(8,2)
MANAGER_ID		NUMBER(4)
DEPARTMENT_ID		NUMBER(6)


```
SELECT * FROM retired_employees;
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	RETIRED_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT_ID
1	301	Rick	Dayle	RDAYLE	18-MAR-10	AD_PRES	8000	124	90
2	302	Meena	Rac	MRAC	21-SEP-11	AD_VP	11000	149	90
3	303	Mex	Haan	MHAAN	13-JAN-10	AD_VP	9500	149	80
4	304	Alexandera	Runold	ARUNOLD	03-JAN-11	IT_PROG	7500	124	60
5	305	Bruk	Ernst	BERNST	21-MAY-10	IT_PROG	6000	149	60
6	306	Dravid	Aust	DAUST	25-JUN-09	IT_PROG	4800	124	60
7	307	Raj	Patil	RPATIL	05-FEB-12	IT_PROG	4800	201	60
8	308	Rahul	Bose	RBOSE	17-AUG-12	FI_MGR	12008	124	100
9	309	Dany	Fav	DFAV	16-AUG-11	FI_ACCOUNT	9000	101	100
10	310	James	Ken	JKHEN	28-SEP-10	FI_ACCOUNT	8200	101	90
11	311	Shana	Garg	SGARG	30-SEP-10	FI_ACCOUNT	7700	201	100
12	313	Lui	Pops	LPOPS	07-DEC-10	FI_ACCOUNT	6900	201	100
13	314	Del	Raph	DRAPH	07-DEC-12	PU_MAN	11000	101	30
14	315	Alex	Khurl	AKHURL	18-MAY-11	PU_CLERK	3100	149	30
15	312	Supriya	Ananth	SANANTH	07-JUN-14	FI_ACCOUNT	7800	124	100

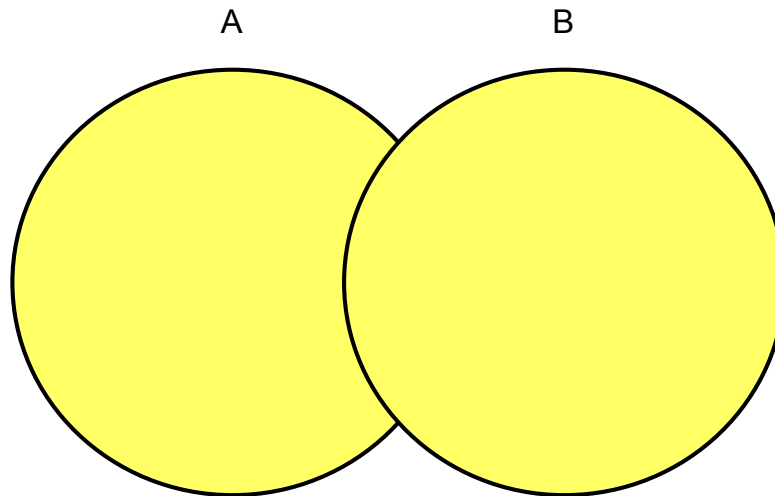
Lektionsagenda

- Mengenoperatoren – Typen und Richtlinien
- In dieser Lektion verwendete Tabellen
- **Operatoren UNION und UNION ALL**
- Operator INTERSECT
- Operator MINUS
- SELECT-Anweisungen abgleichen
- Klausel ORDER BY in Mengenoperationen

ORACLE®

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

Operator UNION



Der Operator `UNION` gibt alle Zeilen von beiden Abfragen zurück. Doppelte Zeilen werden zuvor entfernt.

ORACLE

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

Der Operator `UNION` vereinigt die Zeilen, die von der einen und der anderen Abfrage gewählt werden. Mit dem Operator `UNION` können Sie alle Zeilen aus mehreren Tabellen zurückgeben und dabei mehrfach vorhandene Zeilen entfernen.

Richtlinien

- Die Anzahl der gewählten Spalten muss gleich sein.
- Die Datentypen der gewählten Spalten müssen zur gleichen Datentypgruppe gehören (beispielsweise `NUMERIC` oder `CHARACTER`).
- Die Namen der Spalten müssen nicht identisch sein.
- Der Operator `UNION` bezieht sich auf alle gewählten Spalten.
- Nullwerte werden bei der Suche nach mehrfach vorhandenen Werten berücksichtigt.
- Die Ausgabe wird standardmäßig in aufsteigender Reihenfolge nach den Spalten in der Klausel `SELECT` sortiert.

Operator UNION – Beispiel

Informationen zur Tätigkeit aller aktuellen und pensionierten Mitarbeiter anzeigen, wobei jede Tätigkeit nur einmal ausgegeben werden soll

```
SELECT job_id
FROM employees
UNION
SELECT job_id
FROM retired_employees
```

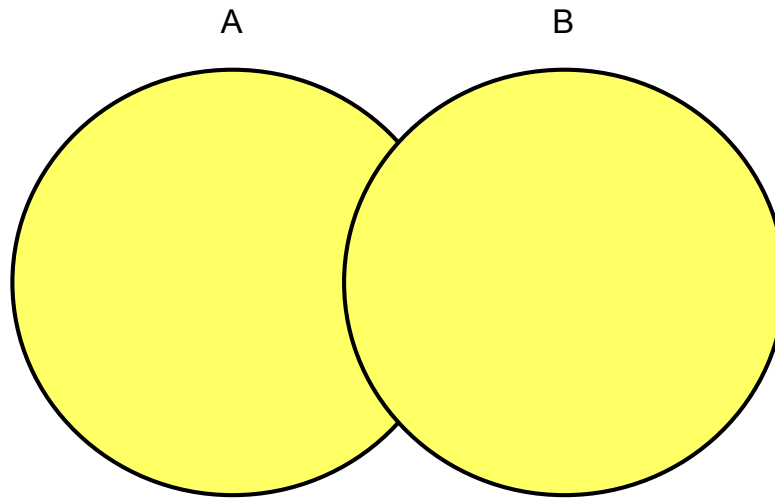
JOB_ID
1 AC_ACCOUNT
2 AC_MGR
3 AD_ASST
4 AD_PRES
5 AD_VP
6 FI_ACCOUNT
7 FI_MGR
8 IT_PROG
9 MK_MAN
10 MK_REP
11 PU_CLERK
12 PU_MAN
13 SA_MAN
14 SA_REP
15 ST_CLERK
16 ST_MAN

ORACLE

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

Der Operator UNION entfernt mehrfach vorkommende Datensätze. Identische Datensätze, die in der Tabelle EMPLOYEES und in der Tabelle RETIRED_EMPLOYEES enthalten sind, werden nur einmal angezeigt.

Operator UNION ALL



Der Operator `UNION ALL` gibt die Zeilen aus beiden Abfragen einschließlich aller mehrfach vorhandenen Zeilen zurück.

ORACLE

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

Mit dem Operator `UNION ALL` geben Sie die Vereinigungsmenge aller Zeilen aus mehreren Abfragen zurück.

Richtlinien

Für `UNION ALL` gelten dieselben Richtlinien wie für `UNION`. Ausnahmen: Im Gegensatz zu `UNION` bleiben mehrfach vorhandene Zeilen erhalten, und die Ausgabe wird standardmäßig nicht sortiert.

Operator UNION ALL – Beispiel

Tätigkeiten und Abteilungen aller aktuellen und pensionierten Mitarbeiter anzeigen

```
SELECT job_id, department_id
FROM employees
UNION ALL
SELECT job_id, department_id
FROM retired_employees
ORDER BY job_id;
```

JOB_ID	DEPARTMENT_ID
1 AC_ACCOUNT	110
2 AC_MGR	110
3 AD_ASST	10
4 AD_PRES	90
5 AD_PRES	90
6 AD_VP	90
7 AD_VP	80
8 AD_VP	90
9 AD_VP	90

...

28 SA_REP	80
29 SA_REP	80
30 SA_REP	(null)
31 ST_CLERK	50
32 ST_CLERK	50
33 ST_CLERK	50
34 ST_CLERK	50
35 ST_MAN	50

ORACLE

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

Im Beispiel werden 35 Zeilen gewählt. Die Kombination der beiden Tabellen ergibt insgesamt 35 Zeilen. Der Operator `UNION ALL` entfernt keine mehrfach vorhandenen Zeilen. Der Operator `UNION` gibt alle sowohl von der einen als auch von der anderen Abfrage gewählten eindeutigen Zeilen zurück. Der Operator `UNION ALL` gibt alle sowohl von der einen als auch von der anderen Abfrage gewählten Zeilen einschließlich der mehrfach vorhandenen Zeilen zurück. Im folgenden Beispiel wird die Abfrage der Folie mit der Klausel `UNION` durchgeführt:

```
SELECT job_id, department_id
FROM employees
UNION
SELECT job_id, department_id
FROM retired_employees
ORDER BY job_id;
```

Diese Abfrage gibt nur 19 Zeilen zurück, da alle doppelt vorhandenen Zeilen entfernt wurden.

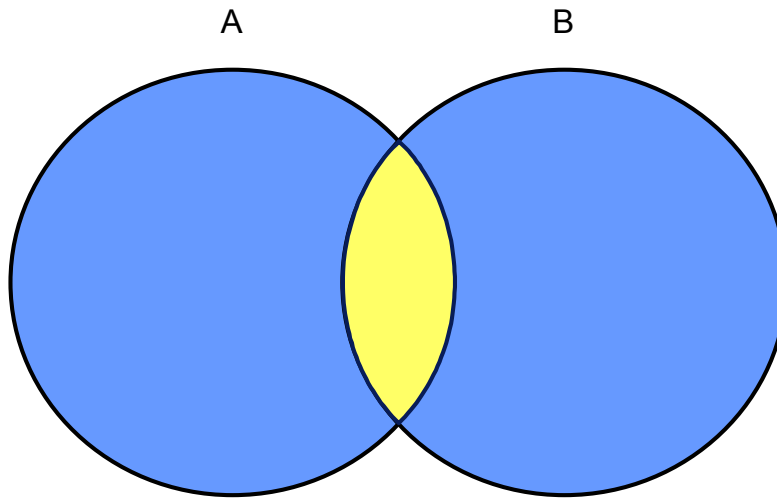
Lektionsagenda

- Mengenoperatoren – Typen und Richtlinien
- In dieser Lektion verwendete Tabellen
- Operatoren UNION und UNION ALL
- **Operator INTERSECT**
- Operator MINUS
- SELECT-Anweisungen abgleichen
- Klausel ORDER BY in Mengenoperationen

ORACLE

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

Operator INTERSECT



Der Operator `INTERSECT` gibt nur diejenigen Zeilen zurück, die in beiden Abfragen vorkommen.

ORACLE

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

Mit dem Operator `INTERSECT` wird die Schnittmenge der Zeilen zurückgegeben, die in mehreren Abfragen vorkommen.

Richtlinien

- Anzahl und Datentyp der von den `SELECT`-Anweisungen gewählten Spalten müssen in allen in der Abfrage verwendeten `SELECT`-Anweisungen identisch sein. Die Spaltennamen müssen nicht identisch sein.
- Eine Umkehrung der Reihenfolge der die Schnittmenge bildenden Tabellen hat keinen Einfluss auf das Ergebnis.
- `INTERSECT` berücksichtigt Nullwerte.

Operator INTERSECT – Beispiel

Gemeinsame Manager- und Abteilungs-IDs für aktuelle und frühere Mitarbeiter anzeigen

```
SELECT manager_id, department_id  
FROM employees  
INTERSECT  
SELECT manager_id, department_id  
FROM retired_employees
```

	MANAGER_ID	DEPARTMENT_ID
1	149	80

ORACLE

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

Im Beispiel auf der Folie gibt die Abfrage nur die Datensätze zurück, die in den gewählten Spalten beider Tabellen dieselben Werte aufweisen.

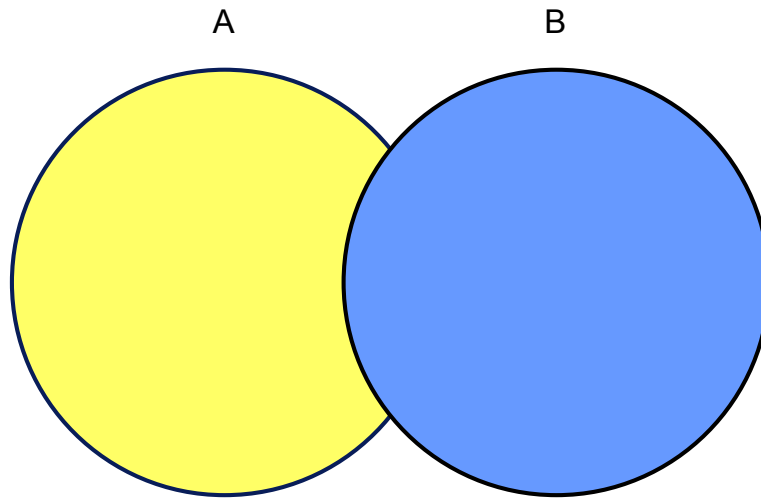
Lektionsagenda

- Mengenoperatoren – Typen und Richtlinien
- In dieser Lektion verwendete Tabellen
- Operatoren UNION und UNION ALL
- Operator INTERSECT
- **Operator MINUS**
- SELECT-Anweisungen abgleichen
- Klausel ORDER BY in Mengenoperationen

ORACLE

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

Operator MINUS



Der Operator `MINUS` gibt alle eindeutigen Zeilen zurück, die von der ersten Abfrage gewählt wurden, aber in der Ergebnismenge der zweiten Abfrage nicht enthalten sind.

ORACLE

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

Mit dem Operator `MINUS` werden alle eindeutigen Zeilen zurückgegeben, die von der ersten Abfrage gewählt wurden, aber in der Ergebnismenge der zweiten Abfrage nicht enthalten sind (erste `SELECT`-Anweisung `MINUS` zweite `SELECT`-Anweisung).

Hinweis: In allen von der Abfrage verwendeten `SELECT`-Anweisungen muss die Anzahl der Spalten identisch sein, und die Datentypen der von den `SELECT`-Anweisungen gewählten Spalten müssen der gleichen Datentypgruppe angehören. Die Spaltennamen müssen nicht identisch sein.

Operator MINUS – Beispiel

Personalnummer und Tätigkeits-ID aller Mitarbeiter aus der Vertriebsabteilung anzeigen

```
SELECT employee_id, job_id
FROM employees
WHERE department_id = 80
MINUS
SELECT employee_id, job_id
FROM retired_employees
WHERE department_id = 90;
```

	EMPLOYEE_ID	JOB_ID
1	149	SA_MAN
2	174	SA_REP
3	176	SA_REP

ORACLE

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

Im Beispiel auf der Folie werden die Personalnummern in der Tabelle `RETIRED_EMPLOYEES` von denen in der Tabelle `EMPLOYEES` subtrahiert. Die angezeigte Ergebnismenge umfasst die nach der Subtraktion verbleibenden Mitarbeiter. Sie werden durch Zeilen repräsentiert, die in der Tabelle `EMPLOYEES`, aber nicht in der Tabelle `RETIRED_EMPLOYEES` enthalten sind. Diese Zeilen entsprechen den Datensätzen der Mitarbeiter, die in der Vertriebsabteilung arbeiten.

Lektionsagenda

- Mengenoperatoren – Typen und Richtlinien
- In dieser Lektion verwendete Tabellen
- Operatoren UNION und UNION ALL
- Operator INTERSECT
- Operator MINUS
- **SELECT-Anweisungen abgleichen**
- Klausel ORDER BY in Mengenoperationen

ORACLE

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

SELECT-Anweisungen abgleichen

Datentyp abgleichen (mit der Funktion `TO_CHAR` oder jeder anderen Konvertierungsfunktion), wenn bestimmte Spalten in einer der Tabellen nicht vorhanden sind

```
SELECT location_id, department_name "Department",  
       TO_CHAR(NULL) "Warehouse location"  
FROM departments  
UNION  
SELECT location_id, TO_CHAR(NULL) "Department",  
       state_province  
FROM locations;
```

ORACLE

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

Da alle `SELECT`-Listen in den Abfragen die gleiche Anzahl von Ausdrücken enthalten müssen, können Sie zur Einhaltung dieser Regel die Dummyspalten und die Funktionen zur Datentypkonvertierung verwenden. Um die Spaltenliste explizit abzugleichen, können Sie an den fehlenden Positionen `NULL`-Spalten einfügen, damit Anzahl und Datentyp der gewählten Spalten in allen `SELECT`-Anweisungen übereinstimmen. Auf der Folie wird `Warehouse location` als Überschrift der Dummyspalte angegeben. Mit der Funktion `TO_CHAR` wird in der ersten Abfrage der Datentyp `VARCHAR2` der Spalte `state_province` abgeglichen, die von der zweiten Abfrage abgerufen wird. Entsprechend wird mit der Funktion `TO_CHAR` in der zweiten Abfrage der Datentyp `VARCHAR2` der Spalte `department_name` abgeglichen, die von der ersten Abfrage abgerufen wird.

SELECT-Anweisungen abgleichen – Beispiel

Mit dem Operator UNION Name, JOB_ID und HIRE_DATE aller Mitarbeiter anzeigen

```
SELECT FIRST_NAME, JOB_ID, TO_DATE(hire_date) "HIRE_DATE"
FROM employees
UNION
SELECT FIRST_NAME, JOB_ID, TO_DATE(NULL) "HIRE_DATE"
FROM retired_employees;
```

	FIRST_NAME	JOB_ID	HIRE_DATE
1	Alex	PU_CLERK	(null)
2	Alexander	IT_PROG	03-JAN-06
3	Alexandera	IT_PROG	(null)
4	Bruce	IT_PROG	21-MAY-07
5	Bruk	IT_PROG	(null)
6	Curtis	ST_CLERK	29-JAN-05
7	Dany	FI_ACCOUNT	(null)
8	Del	PU_MAN	(null)
9	Diana	IT_PROG	07-FEB-07

...

ORACLE

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

Es gibt mehrere Spalten, die sowohl in der Tabelle `EMPLOYEES` als auch in der Tabelle `RETIRED_EMPLOYEES` vorhanden sind (beispielsweise `EMPLOYEE_ID`, `JOB_ID` und `DEPARTMENT_ID`). Wie gehen Sie aber vor, wenn Sie mit dem Operator `UNION` die Angaben zu `FIRST_NAME`, `JOB_ID` und `HIRE_DATE` anzeigen möchten und wissen, dass `HIRE_DATE` nur in der Tabelle `EMPLOYEES` enthalten ist?

Im Codebeispiel auf der Folie werden die Spalten `FIRST_NAME` und `JOB_ID` in den Tabellen `EMPLOYEES` und `RETIRED_EMPLOYEES` abgeglichen. Die `SELECT`-Anweisung für `RETIRED_EMPLOYEES` wird um `NULL` ergänzt, um der Spalte `HIRE_DATE` in der `SELECT`-Anweisung für `EMPLOYEES` zu entsprechen.

In den Ergebnissen auf der Folie enthält jede Ausgabezeile, die einem Datensatz aus der Tabelle `RETIRED_EMPLOYEES` entspricht, in der Spalte `HIRE_DATE` den Wert `NULL`.

Lektionsagenda

- Mengenoperatoren – Typen und Richtlinien
- In dieser Lektion verwendete Tabellen
- Operatoren UNION und UNION ALL
- Operator INTERSECT
- Operator MINUS
- SELECT-Anweisungen abgleichen
- **Klausel ORDER BY in Mengenoperationen**

ORACLE®

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

Klausel ORDER BY in Mengenoperationen

- Die Klausel ORDER BY darf nur einmal am Ende der zusammengesetzten Abfrage angegeben werden.
- Zusammengesetzte Abfragen dürfen keine einzelnen ORDER BY-Klauseln enthalten.
- Die Klausel ORDER BY erkennt nur die Spalten aus der ersten SELECT-Abfrage.
- Standardmäßig wird die Ausgabe in aufsteigender Reihenfolge nach der ersten Spalte der ersten SELECT-Abfrage sortiert.

ORACLE

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

Die Klausel ORDER BY darf in einer zusammengesetzten Abfrage nur einmal verwendet werden. Dabei muss die Klausel ORDER BY an das Ende der Abfrage gestellt werden. Die Klausel ORDER BY akzeptiert den Spaltennamen oder einen Alias. Standardmäßig wird die Ausgabe in aufsteigender Reihenfolge entsprechend der ersten Spalte in der ersten SELECT-Abfrage sortiert.

Hinweis: Die Klausel ORDER BY erkennt die Spaltennamen der zweiten SELECT-Abfrage nicht. Um Verwechslungen bei Spaltennamen zu vermeiden, wird ORDER BY in der Regel für die Sortierung nach Spaltenpositionen eingesetzt.

Beispiel: In der folgenden Anweisung wird die Ausgabe in aufsteigender Reihenfolge nach job_id angezeigt.

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   retired_employees
ORDER BY 2;
```

Wenn Sie ORDER BY nicht angeben, wird die Ausgabe standardmäßig in aufsteigender Reihenfolge nach employee_id sortiert. Sie können die Ausgabe nicht anhand der Spalten aus der zweiten Abfrage sortieren.

Quiz

Welche beiden Richtlinien gelten für Mengenoperatoren?

- a. Die `SELECT`-Listen müssen dieselbe Anzahl von Ausdrücken enthalten.
- b. Die Ausführungsreihenfolge lässt sich nicht mithilfe von Klammern ändern.
- c. Der Datentyp jeder Spalte in der zweiten Abfrage muss mit dem Datentyp der entsprechenden Spalte in der ersten Abfrage übereinstimmen.
- d. Die Klausel `ORDER BY` darf in einer zusammengesetzten Abfrage nur einmal verwendet werden, es sei denn, der Operator `UNION ALL` wird verwendet.

ORACLE

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

Richtige Antwort: a, c

Zusammenfassung

In dieser Lektion haben Sie Folgendes gelernt:

- Mit `UNION` alle eindeutigen Zeilen zurückgeben
- Mit `UNION ALL` alle Zeilen einschließlich mehrfach vorhandener Zeilen zurückgeben
- Mit `INTERSECT` nur die Zeilen zurückgeben, die in beiden Abfragen gemeinsam vorhanden sind
- Mit `MINUS` alle eindeutigen Zeilen zurückgeben, die von der ersten, aber nicht von der zweiten Abfrage gewählt werden
- `ORDER BY` erst am Ende der Anweisung angeben

ORACLE®

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

- Der Operator `UNION` gibt alle eindeutigen Zeilen zurück, die von den einzelnen Abfragen der zusammengesetzten Abfrage gewählt werden. Mit dem Operator `UNION` können Sie alle Zeilen aus mehreren Tabellen zurückgeben und dabei mehrfach vorhandene Zeilen entfernen.
- Mit dem Operator `UNION ALL` geben Sie alle Zeilen aus mehreren Abfragen zurück. Anders als beim Operator `UNION` werden mehrfach vorhandene Zeilen nicht entfernt, und die Ausgabe wird standardmäßig nicht sortiert.
- Mit dem Operator `INTERSECT` geben Sie nur die Zeilen zurück, die in mehreren Abfragen gemeinsam vorkommen.
- Mit dem Operator `MINUS` geben Sie Zeilen aus der ersten Abfrage zurück, die in der zweiten Abfrage nicht vorkommen.
- Die Klausel `ORDER BY` wird erst am Ende der zusammengesetzten Anweisung angegeben.
- Stellen Sie sicher, dass die entsprechenden Ausdrücke in den `SELECT`-Listen in Anzahl und Datentyp übereinstimmen.

Übungen zu Lektion 9 – Überblick

In dieser Übung erstellen Sie Berichte mit den Operatoren:

- UNION
- INTERSECT
- MINUS

ORACLE

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

In dieser Übung erstellen Sie Abfragen mit Mengenoperatoren.

10

Tabellen mit DML-Anweisungen verwalten

ORACLE®

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

Ziele

Nach Ablauf dieser Lektion haben Sie folgende Ziele erreicht:

- DML-Anweisungen beschreiben
- Transaktionen steuern

ORACLE

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

In dieser Lektion lernen Sie, wie Sie mithilfe von DML-(Data Manipulation Language)-Anweisungen Zeilen in Tabellen einfügen, aktualisieren und löschen. Darüber hinaus wird erläutert, wie Transaktionen mit den Anweisungen `COMMIT`, `SAVEPOINT` und `ROLLBACK` gesteuert werden.

Lektionsagenda

- Tabellen neue Zeilen hinzufügen
 - INSERT-Anweisungen
- Daten in Tabellen ändern
 - UPDATE-Anweisungen
- Zeilen aus Tabellen entfernen
 - DELETE-Anweisungen
 - TRUNCATE-Anweisungen
- Datenbanktransaktionen mit COMMIT, ROLLBACK und SAVEPOINT steuern
- Lesekonsistenz
- Klausel FOR UPDATE in SELECT-Anweisungen

ORACLE

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

Data Manipulation Language (DML)

- DML-Anweisungen werden ausgeführt, wenn Sie:
 - neue Zeilen in Tabellen einfügen
 - vorhandene Zeilen in Tabellen bearbeiten
 - vorhandene Zeilen aus Tabellen löschen
- Eine *Transaktion* besteht aus einer Zusammenstellung von DML-Anweisungen, die eine logische Arbeitseinheit bilden.

ORACLE

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

Die Data Manipulation Language (DML) ist eine Hauptkomponente von SQL. Wenn Sie Datenbankdaten hinzufügen, aktualisieren oder löschen möchten, führen Sie eine DML-Anweisung aus. Eine Zusammenstellung von DML-Anweisungen, die eine logische Arbeitseinheit bilden, wird als *Transaktion* bezeichnet.

Stellen Sie sich die Datenbank einer Bank vor. Wenn ein Bankkunde Geld von seinem Sparkonto auf sein Girokonto überweist, kann die Transaktion aus drei separaten Vorgängen bestehen: Betrag vom Sparkonto abbuchen, Betrag dem Girokonto gutschreiben und Transaktion im Transaktionsjournal aufzeichnen. Der Oracle-Server muss sicherstellen, dass alle drei SQL-Anweisungen ausgeführt werden, damit der Saldo der Konten korrekt ist. Wenn die Ausführung einer der Anweisungen der Transaktion verhindert wird, müssen die anderen Anweisungen der Transaktion rückgängig gemacht werden.

Hinweise

- Für die meisten DML-Anweisungen in dieser Lektion gilt die Voraussetzung, dass keine Tabellen-Constraints verletzt werden dürfen. Constraints werden im weiteren Verlauf dieses Kurses behandelt.
- Um die DML-Anweisungen auszuführen, klicken Sie in SQL Developer auf das Symbol **Run Script** oder drücken [F5]. Die Rückmeldungen werden in der Registerkarte **Script Output** angezeigt.

Tabellen neue Zeilen hinzufügen

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10 Administration	200	1700
2	20 Marketing	201	1800
3	50 Shipping	124	1500
4	60 IT	103	1400
5	80 Sales	149	2500
6	90 Executive	100	1700
7	110 Accounting	205	1700
8	190 Contracting	(null)	1700

70 Public Relations	100	1700
---------------------	-----	------

Neue Zeile

Neue Zeile in die Tabelle DEPARTMENTS einfügen

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	70 Public Relations	100	1700
2	10 Administration	200	1700
3	20 Marketing	201	1800
4	50 Shipping	124	1500
5	60 IT	103	1400
6	80 Sales	149	2500
7	90 Executive	100	1700
8	110 Accounting	205	1700
9	190 Contracting	(null)	1700

ORACLE

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

Die Grafik auf der Folie zeigt, wie der Tabelle DEPARTMENTS eine neue Abteilung hinzugefügt wird.

INSERT-Anweisungen – Syntax

- Tabellen mit der Anweisung `INSERT` neue Zeilen hinzufügen:

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- Mit dieser Syntax wird nur jeweils eine Zeile eingefügt.

ORACLE®

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

Mit der Anweisung `INSERT` fügen Sie einer Tabelle neue Zeilen hinzu.

Für die Syntax gilt:

<i>table</i>	Steht für den Namen der Tabelle
<i>column</i>	Name der Tabellenspalte, die gefüllt werden soll
<i>value</i>	Entsprechender Wert für die Spalte

Hinweis: Wenn Sie diese Anweisung mit der Klausel `VALUES` verwenden, wird nur jeweils eine Zeile in die Tabelle eingefügt.

Neue Zeilen einfügen

- Neue Zeile mit Werten für jede Spalte einfügen
- Werte in der Standardreihenfolge der Spalten in der Tabelle angeben
- Optional die Spalten in der Klausel `INSERT` auflisten

```
INSERT INTO departments(department_id,  
                        department_name, manager_id, location_id)  
VALUES (70, 'Public Relations', 100, 1700);  
1 rows inserted
```

- Zeichen- und Datumswerte in Hochkommas setzen

ORACLE

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

Da die neuen Zeilen, die Sie in die Tabelle einfügen, Werte für die einzelnen Spalten enthalten können, ist die Spaltenliste in der Klausel `INSERT` nicht erforderlich. Sie müssen in diesem Fall jedoch für jede Spalte einen Wert gemäß der Standardreihenfolge der Spalten in der Tabelle angeben.

```
DESCRIBE departments
```

Aus Gründen der Übersichtlichkeit empfiehlt es sich, eine Spaltenliste in die Klausel `INSERT` aufzunehmen. Setzen Sie Zeichen- und Datumswerte in Hochkommas. Numerische Werte sollten nicht in Hochkommas gesetzt werden.

Zeilen mit Nullwerten einfügen

- Implizite Methode: Spalte aus der Spaltenliste entfernen

```
INSERT INTO departments (department_id,  
                          department_name)  
VALUES (30, 'Purchasing');  
1 rows inserted
```

- Explizite Methode: In der Klausel VALUES das Schlüsselwort NULL angeben

```
INSERT INTO departments  
VALUES (100, 'Finance', NULL, NULL);  
1 rows inserted
```

ORACLE

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

Methode	Beschreibung
Implizit	Spalte aus der Spaltenliste entfernen
Explizit	Das Schlüsselwort <code>NULL</code> in der <code>VALUES</code> -Liste angeben; die leere Zeichenfolge (<code>' '</code>) in der <code>VALUES</code> -Liste für Zeichenfolgen und Datumsangaben angeben.

Vergewissern Sie sich, dass Sie in der Zielspalte Nullwerte verwenden können. Prüfen Sie dazu den Status `Null` mit dem Befehl `DESCRIBE`.

Der Oracle-Server setzt automatisch alle Constraints in Bezug auf Datentypen, Datenbereiche und Datenintegrität durch. Für alle Spalten, die nicht explizit aufgelistet sind, wird ein Nullwert in die neue Zeile eingetragen, es sei denn, für die fehlenden Spalten sind Standardwerte vorhanden.

Die Prüfung auf gängige Fehler bei der Benutzereingabe erfolgt in der folgenden Reihenfolge:

- Für eine `NOT NULL`-Spalte fehlt ein erforderlicher Wert.
- Doppelter Wert verletzt ein `UNIQUE`- oder `PRIMARY KEY`-Constraint.
- Ein Wert verletzt ein `CHECK`-Constraint.
- Für das `FOREIGN KEY`-Constraint wird eine referenzielle Integrität aufrechterhalten.
- Datentyp stimmt nicht überein, oder Werte sind für die Spalte zu lang.

Hinweis: Die Verwendung einer Spaltenliste wird empfohlen, da sie `INSERT`-Anweisungen übersichtlicher, zuverlässiger und weniger fehleranfällig macht.

Spezielle Werte einfügen

Die Funktion `SYSDATE` zeichnet das aktuelle Datum und die aktuelle Uhrzeit auf.

```
INSERT INTO employees (employee_id,
                        first_name, last_name,
                        email, phone_number,
                        hire_date, job_id, salary,
                        commission_pct, manager_id,
                        department_id)
VALUES (113,
        'Louis', 'Popp',
        'LPOPP', '515.124.4567',
        CURRENT_DATE, 'AC_ACCOUNT', 6900,
        NULL, 205, 110);
```

1 rows inserted

ORACLE

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

Mithilfe von Funktionen können Sie spezielle Werte in eine Tabelle eingeben.

Im Beispiel auf der Folie werden Informationen zum Mitarbeiter Popp in die Tabelle `EMPLOYEES` aufgenommen. In der Spalte `HIRE_DATE` werden das aktuelle Datum und die aktuelle Uhrzeit angegeben. Die Funktion `CURRENT_DATE` ruft das aktuelle Datum in der Zeitzone der Session ab. Mit der Funktion `USER` wird beim Einfügen von Zeilen in Tabellen der aktuelle Benutzername aufgezeichnet.

Hinzufügungen zur Tabelle prüfen

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

Spezielle Datums- und Uhrzeitwerte einfügen

- Neuen Mitarbeiter hinzufügen

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 2003', 'MON DD, YYYY'),
             'SA_REP', 11000, 0.2, 100, 60);
```

1 rows inserted

- Prüfen, ob der Mitarbeiter hinzugefügt wurde

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-03	SA_REP	11000	0.2	100

ORACLE

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

Zum Einfügen eines Datumswertes wird in der Regel das Format `DD-MON-RR` verwendet. Mit dem Format `RR` gibt das System automatisch das richtige Jahrhundert zurück.

Sie können den Datumswert auch im Format `DD-MON-YYYY` angeben. Dieses Format wird empfohlen, weil es das Jahrhundert eindeutig angibt und nicht von der internen Formatlogik `RR` zur Angabe des richtigen Jahrhunderts abhängt.

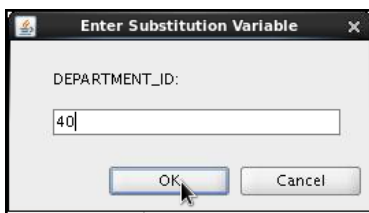
Verwenden Sie die Funktion `TO_DATE`, wenn Sie ein Datum in einem Format eingeben müssen, das nicht dem Standardformat entspricht (beispielsweise ein anderes Jahrhundert- oder ein bestimmtes Uhrzeitformat).

Im Beispiel auf der Folie werden Informationen zum Mitarbeiter Raphealy in der Tabelle `EMPLOYEES` aufgezeichnet. Die Spalte `HIRE_DATE` wird auf den 3. Februar 2003 eingestellt.

Skripte erstellen

- In SQL-Anweisungen mit der Substitutionsvariablen & zur Eingabe von Werten auffordern
- & ist ein Platzhalter für den Variablenwert.

```
INSERT INTO departments
      (department_id, department_name, location_id)
VALUES (&department_id, '&department_name', &location);
```

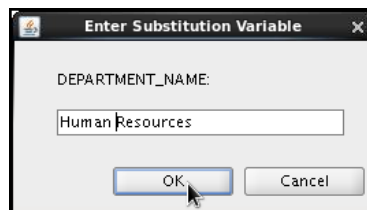


Enter Substitution Variable

DEPARTMENT_ID:

40

OK Cancel

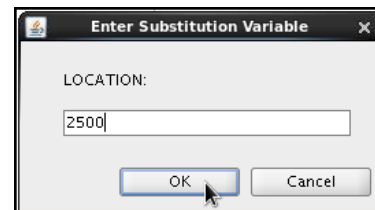


Enter Substitution Variable

DEPARTMENT_NAME:

Human Resources

OK Cancel



Enter Substitution Variable

LOCATION:

2500

OK Cancel

ORACLE

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

Sie können Befehle mit Substitutionsvariablen in einer Datei speichern und die Befehle der Datei ausführen. Im Beispiel auf der Folie werden Informationen zu einer Abteilung in der Tabelle `DEPARTMENTS` aufgezeichnet.

Wenn Sie die Skriptdatei ausführen, werden Sie aufgefordert, für jede Substitutionsvariable (&) einen Wert einzugeben. Klicken Sie nach Eingabe des Wertes für die Substitutionsvariable auf **OK**. Die eingegebenen Werte ersetzen die Substitutionsvariablen in der Anweisung. Dadurch können Sie dieselbe Skriptdatei mehrfach verwenden und bei jeder Ausführung eine andere Wertemenge angeben.

Zeilen aus anderen Tabellen kopieren

- Anweisung INSERT mit einer Unterabfrage erstellen:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM employees
WHERE job_id LIKE '%REP%';
```

5 rows inserted.

- Keine VALUES-Klauseln verwenden
- Die Anzahl der Spalten in der Klausel INSERT muss mit der Anzahl der Spalten in der Unterabfrage übereinstimmen.
- Alle von der Unterabfrage zurückgegebenen Zeilen werden in die Tabelle sales_reps eingefügt.

ORACLE

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

Mit der Anweisung INSERT können Sie einer Tabelle Zeilen hinzufügen, indem Werte aus vorhandenen Tabellen abgeleitet werden. Damit die auf der Folie gezeigte Anweisung INSERT INTO funktioniert, müssen Sie die Tabelle sales_reps bereits mit der Anweisung CREATE TABLE erstellt haben. Die Anweisung CREATE TABLE wird in der Lektion "Data Definition Language – Einführung" erläutert.

Anstelle der Klausel VALUES verwenden Sie eine Unterabfrage.

Syntax

```
INSERT INTO table [ column (, column) ] subquery;
```

Für die Syntax gilt:

<i>table</i>	Steht für den Namen der Tabelle
<i>column</i>	Name der Tabellenspalte, die gefüllt werden soll
<i>subquery</i>	Unterfrage, die Zeilen an die Tabelle zurückgibt

Die Anzahl und die Datentypen der Spalten in der Spaltenliste der Klausel INSERT müssen mit der Anzahl und den Datentypen der Werte in der Unterabfrage übereinstimmen. Je nach Anzahl der von der Unterabfrage zurückgegebenen Zeilen werden keine, eine oder mehrere Zeilen hinzugefügt. Um eine Kopie der Zeilen einer Tabelle zu erstellen, verwenden Sie in der Unterabfrage die Anweisung SELECT*:

```
INSERT INTO copy_emp
SELECT *
FROM employees;
```


Lektionsagenda

- Tabellen neue Zeilen hinzufügen
 - INSERT-Anweisungen
- **Daten in Tabellen ändern**
 - UPDATE-Anweisungen
- Zeilen aus Tabellen entfernen
 - DELETE-Anweisungen
 - TRUNCATE-Anweisungen
- Datenbanktransaktionen mit COMMIT, ROLLBACK und SAVEPOINT steuern
- Lesekonsistenz
- Klausel FOR UPDATE in SELECT-Anweisungen


ORACLE

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

Daten in Tabellen ändern

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

Zeilen in der Tabelle EMPLOYEES aktualisieren: 

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

ORACLE

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

Auf der Folie wird gezeigt, wie die Abteilungsnummer für Mitarbeiter von 60 in 80 geändert wird.

UPDATE-Anweisungen – Syntax

- In einer Tabelle vorhandene Werte mit der Anweisung UPDATE ändern:

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- Falls erforderlich, können mehrere Zeilen gleichzeitig aktualisiert werden.

ORACLE®

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

Sie können die in einer Tabelle vorhandenen Werte mit der Anweisung UPDATE ändern.

Für die Syntax gilt:

<i>table</i>	Steht für den Namen der Tabelle
<i>column</i>	Name der Tabellenspalte, die gefüllt werden soll
<i>value</i>	Entsprechender Wert oder Unterabfrage für die Spalte
<i>condition</i>	Gibt die zu aktualisierenden Zeilen an und besteht aus Spaltennamen, Ausdrücken, Konstanten, Unterabfragen und Vergleichsoperatoren

Prüfen Sie den Aktualisierungsvorgang, indem Sie die aktualisierten Zeilen über eine Tabellenabfrage anzeigen.

Weitere Informationen finden Sie in der *Oracle Database SQL Language Reference* für Oracle Database 12c im Abschnitt "UPDATE".

Hinweis: In der Regel identifizieren Sie eine einzelne zu aktualisierende Zeile mit der Primärschlüsselspalte in der Klausel WHERE. Wenn Sie andere Spalten verwenden, kann es passieren, dass ungewollt mehrere Zeilen aktualisiert werden. Beispielsweise kann eine einzelne, anhand des Namens identifizierte Zeile in der Tabelle EMPLOYEES mehrere Mitarbeiter mit demselben Namen zurückgeben.

Zeilen in Tabellen aktualisieren

- Durch Angabe der Klausel `WHERE` Werte spezifischer Zeilen ändern:

```
UPDATE employees
SET department_id = 50
WHERE employee_id = 113;
```

1 rows updated

- Ohne `WHERE`-Klausel die Werte aller Zeilen der Tabelle ändern:

```
UPDATE copy_emp
SET department_id = 110;
```

22 rows updated

- Um einen Spaltenwert mit `NULL` zu aktualisieren, geben Sie `SET column_name= NULL` an.

ORACLE

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

Mit der Klausel `WHERE` in der Anweisung `UPDATE` können die Werte bestimmter Zeilen geändert werden. Im Beispiel auf der Folie wird der Mitarbeiter 113 (Popp) in die Abteilung 50 übertragen. Wenn Sie die Klausel `WHERE` weglassen, werden die Werte aller Tabellenzeilen geändert. Prüfen Sie die aktualisierten Zeilen in der Tabelle `COPY_EMP`.

```
SELECT last_name, department_id
FROM copy_emp;
```

Beispiel: Die Tätigkeits-ID eines Mitarbeiters hat sich aufgrund eines Wechsels seines Aufgabebereichs von `SA_REP` in `IT_PROG` geändert. Daher muss die `JOB_ID` aktualisiert und das Feld für die Provision auf `NULL` eingestellt werden.

```
UPDATE employees
SET job_id = 'IT_PROG', commission_pct = NULL
WHERE employee_id = 114;
```

Hinweis: Die Tabelle `COPY_EMP` enthält dieselben Daten wie die Tabelle `EMPLOYEES`.

Zwei Spalten mithilfe einer Unterabfrage aktualisieren

Tätigkeit und Gehalt des Mitarbeiters 103 aktualisieren, damit sie mit den Angaben für Mitarbeiter 205 übereinstimmen

```
UPDATE employees
SET (job_id,salary) = (SELECT job_id,salary
                       FROM employees
                       WHERE employee_id = 205)
WHERE employee_id = 103;
```

1 rows updated

ORACLE

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

Sie können mehrere Spalten in der Klausel SET einer UPDATE-Anweisung aktualisieren, indem Sie mehrere Unterabfragen erstellen. Die Syntax lautet wie folgt:

```
UPDATE table
SET column =
    (SELECT column
     FROM table
     WHERE condition)
[ ,
  column =
    (SELECT column
     FROM table
     WHERE condition)]
[WHERE condition] ;
```

Zeilen auf Basis einer anderen Tabelle aktualisieren

Mit Unterabfragen in UPDATE-Anweisungen Zeilenwerte einer Tabelle basierend auf den Werten einer anderen Tabelle aktualisieren:

```
UPDATE copy_emp
SET    department_id = (SELECT department_id
                        FROM employees
                        WHERE employee_id = 100)
WHERE  job_id = (SELECT job_id
                 FROM employees
                 WHERE employee_id = 200);
```

1 rows updated

ORACLE

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

Mithilfe von Unterabfragen in UPDATE-Anweisungen können Sie Werte in einer Tabelle aktualisieren. Im Beispiel auf der Folie wird die Tabelle `COPY_EMP` basierend auf den Werten der Tabelle `EMPLOYEES` aktualisiert. Im Einzelnen wird die Abteilungsnummer aller Mitarbeiter, die die gleiche Tätigkeits-ID wie der Mitarbeiter 200 haben, in die aktuelle Abteilungsnummer des Mitarbeiters 100 geändert.

Lektionsagenda

- Tabellen neue Zeilen hinzufügen
 - INSERT-Anweisungen
- Daten in Tabellen ändern
 - UPDATE-Anweisungen
- Zeilen aus Tabellen entfernen
 - DELETE-Anweisungen
 - TRUNCATE-Anweisungen
- Datenbanktransaktionen mit COMMIT, ROLLBACK und SAVEPOINT steuern
- Lesekonsistenz
- Klausel FOR UPDATE in SELECT-Anweisungen

ORACLE

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

Zeilen aus Tabellen entfernen

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Zeilen aus der Tabelle DEPARTMENTS löschen:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700

ORACLE

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

Im Beispiel auf der Folie wird die Abteilung Contracting aus der Tabelle DEPARTMENTS gelöscht, sofern keine für die Tabelle gültigen Constraints verletzt werden.

DELETE-Anweisungen

Mit der Anweisung `DELETE` vorhandene Zeilen aus einer Tabelle löschen:

```
DELETE [FROM]   table
[WHERE          condition];
```

ORACLE

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

Mit der Anweisung `DELETE` können Sie vorhandene Zeilen aus einer Tabelle löschen.

Für die Syntax gilt:

<i>table</i>	Steht für den Namen der Tabelle
<i>condition</i>	Gibt die zu löschenden Zeilen an und besteht aus Spaltennamen, Ausdrücken, Konstanten, Unterabfragen und Vergleichsoperatoren

Hinweis: Wenn keine Zeilen gelöscht werden, wird die Meldung "0 rows deleted" zurückgegeben (in der Registerkarte **Script Output** in SQL Developer).

Weitere Informationen finden Sie in der *Oracle Database SQL Language Reference* für Oracle Database 12c im Abschnitt "DELETE".

Zeilen aus Tabellen löschen

- Durch Angabe der Klausel `WHERE` bestimmte Zeilen löschen:

```
DELETE FROM departments
WHERE department_name = 'Finance';
```

```
1 rows deleted
```

- Ohne `WHERE`-Klausel alle Zeilen der Tabelle löschen:

```
DELETE FROM copy_emp;
```

```
22 rows deleted
```

ORACLE

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

Wenn Sie in der Anweisung `DELETE` die Klausel `WHERE` angeben, können Sie bestimmte Zeilen löschen. Im ersten Beispiel auf der Folie wird die Finanzabteilung aus der Tabelle `DEPARTMENTS` gelöscht. Sie können den Löschvorgang prüfen, indem Sie die gelöschten Zeilen mit der Anweisung `SELECT` anzeigen.

```
SELECT *
FROM departments
WHERE department_name = 'Finance';
```

Wenn Sie die Klausel `WHERE` weglassen, werden alle Zeilen der Tabelle gelöscht. Im zweiten Beispiel auf der Folie werden alle Zeilen der Tabelle `COPY_EMP` gelöscht, da keine `WHERE`-Klausel angegeben wurde.

Beispiel

Sie entfernen die in der Klausel `WHERE` angegebenen Zeilen:

```
DELETE FROM employees WHERE employee_id = 114;
```

```
DELETE FROM departments WHERE department_id IN (30, 40);
```

Zeilen auf Basis einer anderen Tabelle löschen

Mithilfe von Unterabfragen in DELETE-Anweisungen Zeilen einer Tabelle basierend auf den Werten einer anderen Tabelle entfernen:

```
DELETE FROM employees
WHERE department_id IN
    (SELECT department_id
     FROM departments
     WHERE department_name
           LIKE '%Public%');
1 rows deleted
```

ORACLE

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

Mithilfe von Unterabfragen können Sie Zeilen aus einer Tabelle basierend auf den Werten einer anderen Tabelle löschen. Im Beispiel auf der Folie werden alle Mitarbeiter gelöscht, deren Abteilungsname die Zeichenfolge `Public` enthält.

Die Unterabfrage durchsucht die Tabelle `DEPARTMENTS` und ermittelt die Abteilungsnummer, die dem Abteilungsnamen mit der Zeichenfolge `Public` entspricht. Die Unterabfrage übergibt die Abteilungsnummer anschließend an die Hauptabfrage, die auf Basis dieser Abteilungsnummer Datenzeilen aus der Tabelle `EMPLOYEES` löscht.

TRUNCATE-Anweisungen

- Entfernen alle Zeilen aus einer Tabelle, sodass die Tabelle anschließend leer, die Tabellenstruktur jedoch weiterhin intakt ist
- Sind keine DML-, sondern DDL-Anweisungen und können nicht ohne Weiteres rückgängig gemacht werden
- Syntax:

```
TRUNCATE TABLE table_name;
```

- Beispiel:

```
TRUNCATE TABLE copy_emp;
```

ORACLE

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

Die Anweisung `TRUNCATE` ist eine effizientere Methode, Tabellen zu leeren. Sie entfernt Zeilen aus folgenden Gründen schneller aus Tabellen oder Clustern als die Anweisung `DELETE`:

- Die Anweisung `TRUNCATE` ist eine DDL-Anweisung, die keine Rollbackinformationen generiert. Rollbackinformationen werden im weiteren Verlauf dieser Lektion behandelt.

Tabellen, die einem referenziellen Integritäts-Constraint übergeordnet sind, können nicht mit `TRUNCATE` geleert werden. Sie müssen das Constraint deaktivieren, bevor Sie die Anweisung `TRUNCATE` absetzen. Die Deaktivierung von Constraints wird in der Lektion "DDL-Anweisungen – Einführung" behandelt.

Lektionsagenda

- Tabellen neue Zeilen hinzufügen
 - INSERT-Anweisungen
- Daten in Tabellen ändern
 - UPDATE-Anweisungen
- Zeilen aus Tabellen entfernen
 - DELETE-Anweisungen
 - TRUNCATE-Anweisungen
- **Datenbanktransaktionen mit COMMIT, ROLLBACK und SAVEPOINT steuern**
- Lesekonsistenz
- Klausel FOR UPDATE in SELECT-Anweisungen

ORACLE

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

Datenbanktransaktionen

Eine Datenbanktransaktion umfasst einen der folgenden Anweisungstypen:

- DML-Anweisungen, die zu einer konsistenten Änderung der Daten führen
- Eine DDL-Anweisung
- Eine DCL-Anweisung

ORACLE

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

Der Oracle-Server stellt die Datenkonsistenz auf der Basis von Transaktionen sicher. Transaktionen bieten beim Ändern von Daten mehr Flexibilität und Steuerungsmöglichkeiten und stellen im Falle von abgebrochenen Benutzerprozessen oder Systemausfällen die Datenkonsistenz sicher.

Transaktionen bestehen aus DML-Anweisungen, die eine konsistente Änderung der Daten bewirken. Beispiel: Bei einer Überweisung zwischen zwei Konten muss ein Konto mit einem bestimmten Betrag belastet und derselbe Betrag einem anderen Konto gutgeschrieben werden. Beide Aktionen dürfen nur gemeinsam gelingen oder misslingen. Die Gutschrift darf nicht ohne die Lastschrift erfolgen.

Transaktionstypen

Typ	Beschreibung
Data Manipulation Language (DML)	Umfasst beliebig viele DML-Anweisungen, die der Oracle-Server als einzelne Entität oder logische Arbeitseinheit behandelt
Data Definition Language (DDL)	Umfasst nur eine DDL-Anweisung
Data Control Language (DCL)	Umfasst nur eine DCL-Anweisung

Datenbanktransaktionen – Beginn und Ende

- Eine Transaktion beginnt mit der Ausführung der ersten DML-SQL-Anweisung.
- Eine Transaktion endet mit einem der folgenden Ereignisse:
 - Ein `COMMIT` oder `ROLLBACK` wird abgesetzt.
 - Eine DDL- oder DCL-Anweisung wird ausgeführt (und automatisch festgeschrieben).
 - Der Benutzer beendet SQL Developer oder SQL*Plus.
 - Das System stürzt ab.

ORACLE

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

Wann beginnen und enden Datenbanktransaktionen?

Eine Transaktion beginnt mit der ersten DML-Anweisung und endet bei Eintritt eines der folgenden Ereignisse:

- Eine `COMMIT`- oder `ROLLBACK`-Anweisung wird abgesetzt.
- Eine DDL-Anweisung wird abgesetzt, beispielsweise `CREATE`.
- Eine DCL-Anweisung wird abgesetzt.
- Der Benutzer beendet SQL Developer oder SQL*Plus.
- Ein Computer- oder Systemausfall tritt auf.

Nach dem Abschluss einer Transaktion startet die nächste ausführbare SQL-Anweisung automatisch die nächste Transaktion.

DDL- oder DCL-Anweisungen werden automatisch festgeschrieben und beenden deshalb eine Transaktion implizit.

COMMIT und ROLLBACK – Vorteile

Mit COMMIT- und ROLLBACK-Anweisungen können Sie:

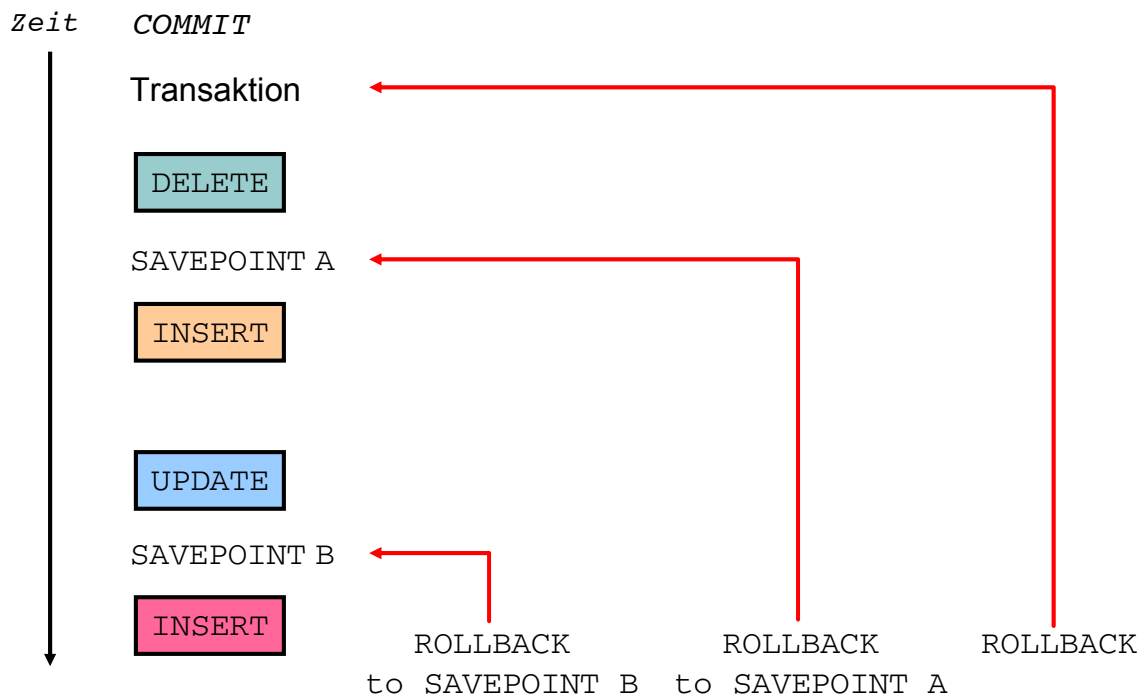
- Datenkonsistenz sicherstellen
- Datenänderungen vor dem dauerhaften Speichern in einer Vorschau anzeigen
- logisch zusammenhängende Vorgänge gruppieren

ORACLE

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

COMMIT- und ROLLBACK-Anweisungen bieten Steuerungsmöglichkeiten in Bezug auf die dauerhafte Speicherung von Datenänderungen.

Explizite Steueranweisungen für Transaktionen



ORACLE

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

Mit den Anweisungen `COMMIT`, `SAVEPOINT` und `ROLLBACK` können Sie die Logik von Transaktionen steuern.

Anweisung	Beschreibung
<code>COMMIT</code>	<code>COMMIT</code> beendet die aktuelle Transaktion, indem alle noch nicht gespeicherten Datenänderungen dauerhaft festgeschrieben werden.
<code>SAVEPOINT name</code>	<code>SAVEPOINT name</code> markiert einen Savepoint innerhalb der aktuellen Transaktion.
<code>ROLLBACK</code>	<code>ROLLBACK</code> beendet die aktuelle Transaktion, indem alle noch nicht gespeicherten Datenänderungen verworfen werden.
<code>ROLLBACK TO SAVEPOINT name</code>	<code>ROLLBACK TO <savepoint></code> rollt die aktuelle Transaktion zum angegebenen Savepoint zurück, wodurch alle Änderungen und/oder Savepoints verworfen werden, die nach dem Savepoint erstellt wurden, zu dem das Rollback erfolgt. Wenn Sie die Klausel <code>TO SAVEPOINT</code> weglassen, rollt die Anweisung <code>ROLLBACK</code> die gesamte Transaktion zurück. Da Savepoints logisch sind, gibt es keine Möglichkeit, die erstellten Savepoints aufzulisten.

Hinweis: Für einen `SAVEPOINT` können Sie keine `COMMIT`-Anweisung ausführen. `SAVEPOINT` entspricht nicht dem ANSI-SQL-Standard.

Änderungen bis zu einer Markierung zurückrollen

- Mit der Anweisung `SAVEPOINT` eine Markierung in der aktuellen Transaktion erstellen
- Mit der Anweisung `ROLLBACK TO SAVEPOINT` zu dieser Markierung zurückrollen

```
UPDATE...  
SAVEPOINT update_done  
SAVEPOINT update_done succeeded.  
INSERT...  
ROLLBACK TO update_done;  
ROLLBACK TO succeeded.
```

ORACLE

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

Mithilfe der Anweisung `SAVEPOINT` können Sie in der aktuellen Transaktion eine Markierung erstellen, die die Transaktion in kleinere Abschnitte aufteilt. Anschließend können Sie mit der Anweisung `ROLLBACK TO SAVEPOINT` noch nicht gespeicherte Änderungen bis zu dieser Markierung verwerfen.

Wenn Sie einen zweiten Savepoint erstellen, der denselben Namen wie ein früherer Savepoint hat, wird der frühere Savepoint gelöscht.

Implizite Transaktionsverarbeitung

- In den folgenden Situationen wird ein automatischer Commit durchgeführt:
 - Eine DDL-Anweisung wurde abgesetzt.
 - Eine DCL-Anweisung wurde abgesetzt.
 - SQL Developer oder SQL*Plus wurde normal ohne explizites Absetzen von COMMIT- oder ROLLBACK-Anweisungen beendet.
- Bei nicht normaler Beendigung von SQL Developer oder SQL*Plus sowie bei einem Systemausfall wird automatisch ein Rollback durchgeführt.

ORACLE

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

Status	Situationen
Automatischer Commit	DDL- oder DCL-Anweisung wurde abgesetzt. SQL Developer oder SQL*Plus wurde normal ohne explizites Absetzen von COMMIT- oder ROLLBACK-Befehlen beendet.
Automatisches Rollback	Nicht normale Beendigung von SQL Developer oder SQL*Plus oder Systemausfall

Hinweis: In SQL*Plus kann der Befehl `AUTOCOMMIT` aktiviert oder deaktiviert werden. Ist der Befehl aktiviert (`ON`), wird jede einzelne DML-Anweisung unmittelbar nach ihrer Ausführung festgeschrieben. Die Änderungen können nicht zurückgerollt werden. Ist der Befehl deaktiviert (`OFF`), kann die Anweisung `COMMIT` weiterhin explizit abgesetzt werden. Die Anweisung `COMMIT` wird auch abgesetzt, wenn Sie eine DDL-Anweisung absetzen oder SQL*Plus beenden. Der Befehl `SET AUTOCOMMIT ON/OFF` wird in SQL Developer übersprungen. Eine DML-Anweisung wird bei normaler Beendigung von SQL Developer nur dann festgeschrieben, wenn die Voreinstellung **Autocommit** aktiviert ist. Um **Autocommit** zu aktivieren, gehen Sie wie folgt vor:

- Wählen Sie im Menü **Tools** die Option **Preferences**. Blenden Sie im Dialogfeld **Preferences** den Eintrag **Database** ein, und wählen Sie **Worksheet Parameters**.
- Aktivieren Sie im rechten Bereich die Option **Autocommit in SQL Worksheet**. Klicken Sie auf **OK**.

Systemausfälle

Bei Unterbrechung einer Transaktion durch einen Systemausfall wird die gesamte Transaktion automatisch zurückgerollt. Dadurch wird verhindert, dass der Fehler unerwünschte Änderungen an den Daten verursacht. Zudem werden die Tabellen in den Zustand zurückgesetzt, in dem sie sich zum Zeitpunkt des letzten Commits befanden. Auf diese Weise schützt der Oracle-Server die Integrität der Tabellen.

Um in SQL Developer die Session normal zu beenden, wählen Sie im Menü **File** die Option **Exit**. SQL*Plus wird normal beendet, wenn Sie in der Eingabeaufforderung den Befehl `EXIT` eingeben. Das Schließen des Fensters wird nicht als normaler Beendigungsvorgang interpretiert.

Zustand der Daten vor COMMIT oder ROLLBACK

- Der vorherige Zustand der Daten lässt sich wiederherstellen.
- Die aktuelle Session kann die Ergebnisse der DML-Vorgänge mit der Anweisung `SELECT` prüfen.
- Andere Sessions können die Ergebnisse der von der aktuellen Session abgesetzten DML-Anweisungen *nicht* anzeigen.
- Die betroffenen Zeilen werden *gesperrt*. Andere Sessions können die Daten in den betroffenen Zeilen nicht ändern.

ORACLE

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

Jede Datenänderung im Verlauf der Transaktion ist temporär, bis die Transaktion festgeschrieben wird.

Der Zustand der Daten vor dem Absetzen der Anweisungen `COMMIT` oder `ROLLBACK` kann wie folgt beschrieben werden:

- Datenmanipulationsvorgänge betreffen hauptsächlich den Datenbankpuffer. Daher kann der vorherige Zustand der Daten wiederhergestellt werden.
- Die aktuelle Session kann die Ergebnisse von Datenmanipulationsvorgängen durch eine Abfrage der Tabellen prüfen.
- Andere Sessions können die Ergebnisse der von der aktuellen Session durchgeführten Datenmanipulationsvorgänge nicht anzeigen. Der Oracle-Server stellt die Lesekonsistenz her, um sicherzustellen, dass die Daten den einzelnen Sessions so angezeigt werden, wie sie zum Zeitpunkt des letzten Commits gültig waren.
- Die betroffenen Zeilen werden gesperrt. Andere Sessions können die Daten in den betroffenen Zeilen nicht ändern.

Zustand der Daten nach COMMIT

- Datenänderungen werden in der Datenbank gespeichert.
- Der vorherige Zustand der Daten wird überschrieben.
- Die Ergebnisse werden allen Sessions angezeigt.
- Sperren der betroffenen Zeilen werden freigegeben. Die Zeilen können von den Sessions wieder bearbeitet werden.
- Alle Savepoints werden gelöscht.

ORACLE

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

Mit der Anweisung `COMMIT` werden alle noch nicht gespeicherten Änderungen dauerhaft festgeschrieben. Der Zustand der Daten nach dem Absetzen der Anweisung `COMMIT` kann wie folgt beschrieben werden:

- Datenänderungen werden in die Datenbank geschrieben.
- Der vorherige Zustand der Daten kann mit normalen SQL-Abfragen nicht wiederhergestellt werden.
- Alle Sessions können die Ergebnisse der Transaktion sehen.
- Die Sperren der betroffenen Zeilen werden freigegeben. Diese Zeilen können anschließend von den Sessions wieder bearbeitet werden.
- Alle Savepoints werden gelöscht.

Daten festschreiben

- Änderungen vornehmen:

```
DELETE FROM EMPLOYEES
WHERE employee_id=113;
1 rows deleted
INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- Änderungen festschreiben:

```
COMMIT;
```

```
committed.
```

ORACLE

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

Im Beispiel auf der Folie wird aus der Tabelle `EMPLOYEES` eine Zeile gelöscht und in die Tabelle `DEPARTMENTS` eine neue Zeile eingefügt. Um die Änderungen zu speichern, wird die Anweisung `COMMIT` abgesetzt.

Beispiel

Sie entfernen die Abteilungen 290 und 300 aus der Tabelle `DEPARTMENTS` und aktualisieren eine Zeile in der Tabelle `EMPLOYEES`. Anschließend speichern Sie die Datenänderung:

```
DELETE FROM departments
WHERE department_id IN (290, 300);

UPDATE employees
SET department_id = 80
WHERE employee_id = 206;

COMMIT;
```

Zustand der Daten nach ROLLBACK

Mit der Anweisung `ROLLBACK` werden alle noch nicht gespeicherten Änderungen verworfen:

- Datenänderungen werden rückgängig gemacht.
- Der vorherige Zustand der Daten wird wiederhergestellt.
- Sperren der betroffenen Zeilen werden freigegeben.

```
DELETE FROM copy_emp;  
ROLLBACK ;
```

ORACLE

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

Mit der Anweisung `ROLLBACK` werden alle noch nicht gespeicherten Änderungen verworfen. Dies hat folgende Auswirkungen:

- Datenänderungen werden rückgängig gemacht.
- Der vorherige Zustand der Daten wird wiederhergestellt.
- Sperren der betroffenen Zeilen werden freigegeben.

Zustand der Daten nach ROLLBACK – Beispiel

```
DELETE FROM test;  
4 rows deleted.  
  
ROLLBACK;  
Rollback complete.  
  
DELETE FROM test WHERE id = 100;  
1 row deleted.  
  
SELECT * FROM test WHERE id = 100;  
No rows selected.  
  
COMMIT;  
Commit complete.
```

ORACLE

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

Sie versuchen, einen Datensatz aus der Tabelle `TEST` zu entfernen. Dabei leeren Sie versehentlich die Tabelle. Sie können den Fehler jedoch korrigieren, eine neue, richtige Anweisung absetzen und die Datenänderung dauerhaft speichern.

Rollbacks auf Anweisungsebene

- Wenn eine einzelne DML-Anweisung nicht erfolgreich ausgeführt werden kann, wird nur diese Anweisung zurückgerollt.
- Der Oracle-Server implementiert einen impliziten Savepoint.
- Alle anderen Änderungen werden beibehalten.
- Transaktionen sollten durch Ausführen einer `COMMIT`- oder `ROLLBACK`-Anweisung explizit beendet werden.

ORACLE

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

Transaktionen können mithilfe eines impliziten Rollbacks teilweise verworfen werden, wenn bei der Ausführung ein Fehler auftritt. Kann eine einzelne DML-Anweisung in einer Transaktion nicht erfolgreich ausgeführt werden, werden die Auswirkungen durch ein Rollback auf Anweisungsebene rückgängig gemacht. Die durch die vorherigen DML-Anweisungen in der Transaktion bewirkten Änderungen werden jedoch nicht verworfen und können vom Benutzer explizit festgeschrieben oder zurückgerollt werden.

Im Falle von DDL-Anweisungen setzt der Oracle-Server vor und nach jeder Anweisung eine implizite Commit-Anweisung ab. Nicht erfolgreich ausgeführte DDL-Anweisungen können daher nicht zurückgerollt werden, da sie bereits festgeschrieben wurden.

Beenden Sie Ihre Transaktionen explizit durch Ausführen einer `COMMIT`- oder `ROLLBACK`-Anweisung.

Lektionsagenda

- Tabellen neue Zeilen hinzufügen
 - INSERT-Anweisungen
- Daten in Tabellen ändern
 - UPDATE-Anweisungen
- Zeilen aus Tabellen entfernen
 - DELETE-Anweisungen
 - TRUNCATE-Anweisungen
- Datenbanktransaktionen mit COMMIT, ROLLBACK und SAVEPOINT steuern
- **Lesekonsistenz**
- Klausel FOR UPDATE in SELECT-Anweisungen

ORACLE

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

Lesekonsistenz

- Die Funktion der Lesekonsistenz stellt eine jederzeit konsistente Datenansicht sicher.
- Änderungen eines Benutzers verursachen keine Konflikte mit den Änderungen eines anderen Benutzers.
- In Bezug auf dieselben Daten stellt die Lesekonsistenz Folgendes sicher:
 - Lesevorgänge müssen keine Schreibvorgänge abwarten.
 - Schreibvorgänge müssen keine Lesevorgänge abwarten.
 - Schreibvorgänge warten Schreibvorgänge ab.

ORACLE

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

Datenbankbenutzer greifen wie folgt auf die Datenbank zu:

- Mit Lesevorgängen (`SELECT`-Anweisungen)
- Mit Schreibvorgängen (`INSERT`-, `UPDATE`- und `DELETE`-Anweisungen)

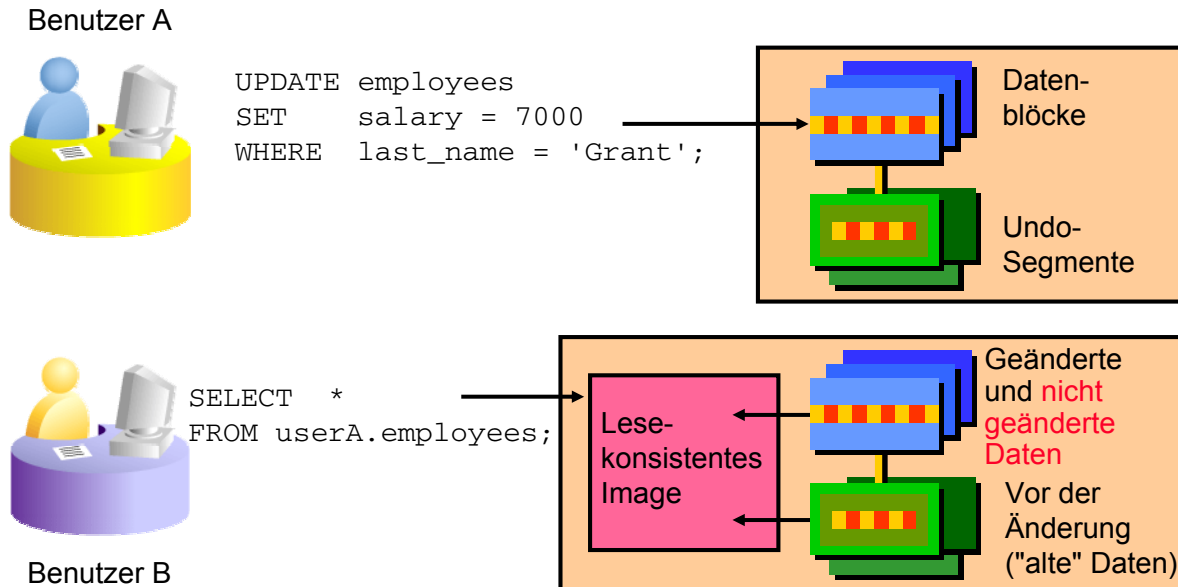
Lesekonsistenz ist erforderlich, um folgendes Verhalten zu gewährleisten:

- Für Lese- und Schreibvorgänge in Datenbanken wird eine konsistente Datensicht sichergestellt.
- Lesevorgängen werden keine Daten angezeigt, die gerade geändert werden.
- Bei Schreibvorgängen ist gewährleistet, dass Änderungen an der Datenbank konsistent erfolgen.
- Änderungen durch einen Schreibvorgang verursachen keine Konflikte mit Änderungen durch einen anderen Schreibvorgang.

Mit der Lesekonsistenz soll sichergestellt werden, dass allen Benutzern der Datenzustand zum Zeitpunkt des letzten Commits angezeigt wird, bevor ein DML-Vorgang gestartet wurde.

Hinweis: Der gleiche Benutzer kann sich bei verschiedenen Sessions anmelden. Jede Session hält die Lesekonsistenz in der oben beschriebenen Weise aufrecht, selbst wenn es sich um die gleichen Benutzer handelt.

Lesekonsistenz implementieren



ORACLE

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

Lesekonsistenz wird automatisch implementiert. Dabei wird eine Teilkopie der Datenbank in den Undo-Segmenten gespeichert. Das lesekonsistente Image besteht aus festgeschriebenen Daten der Tabelle sowie aus alten Daten aus dem Undo-Segment, die geändert, aber noch nicht festgeschrieben wurden.

Wenn Daten in der Datenbank eingefügt, aktualisiert oder gelöscht werden, erstellt der Oracle-Server vor der Änderung eine Kopie der Daten und schreibt sie in ein *Undo-Segment*.

Alle Leseberechtigten (außer dem Benutzer, der die Änderungsanweisung abgesetzt hat) sehen die Datenbank in ihrem Zustand vor der Änderung. Sie sehen einen "Snapshot" der Daten aus dem Undo-Segment.

Solange die Änderungen noch nicht in der Datenbank festgeschrieben sind, werden sie nur von dem Benutzer gesehen, der sie selbst durchführt. Alle anderen Benutzer sehen den Snapshot aus dem Undo-Segment. Auf diese Weise wird garantiert, dass die gelesenen Daten konsistent sind und aktuell nicht geändert werden.

Sobald eine DML-Anweisung festgeschrieben wurde, sind die Änderungen für alle Benutzer sichtbar, die *nach* dem Commit eine *SELECT*-Anweisung absetzen. Der von den *alten* Daten in der Undo-Segmentdatei belegte Speicherplatz wird zur Wiederverwendung freigegeben.

Wird die Transaktion zurückgerollt, werden die Änderungen rückgängig gemacht:

- Die ursprüngliche, ältere Version der Daten im Undo-Segment wird zurück in die Tabelle geschrieben.
- Benutzer sehen die Datenbank im Zustand vor Beginn der Transaktion.

Lektionsagenda

- Tabellen neue Zeilen hinzufügen
 - INSERT-Anweisungen
- Daten in Tabellen ändern
 - UPDATE-Anweisungen
- Zeilen aus Tabellen entfernen
 - DELETE-Anweisungen
 - TRUNCATE-Anweisungen
- Datenbanktransaktionen mit COMMIT, ROLLBACK und SAVEPOINT steuern
- Lesekonsistenz
- **Klausel FOR UPDATE in SELECT-Anweisungen**

ORACLE

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

Klausel FOR UPDATE in SELECT-Anweisungen

- Sperrt die Zeilen in der Tabelle `EMPLOYEES`, in denen `job_id` gleich `SA_REP` ist

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE
ORDER BY employee_id;
```

- Die Sperre wird erst aufgehoben, wenn eine `ROLLBACK`- oder `COMMIT`-Anweisung abgesetzt wird.
- Versucht eine `SELECT`-Anweisung, eine bereits durch einen anderen Benutzer gesperrte Zeile zu sperren, wartet die Datenbank, bis die Zeile verfügbar ist, und gibt dann die Ergebnisse zurück.

ORACLE

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

In der Regel werden keine Zeilen gesperrt, die durch eine `SELECT`-Anweisung in der Datenbank gewählt werden, da die Anzahl gesperrter Datensätze (laut Standard) stets auf ein absolutes Minimum begrenzt sein soll. Gesperrt werden nur die Datensätze, die geändert, aber noch nicht festgeschrieben wurden. Auch dann können die Datensätze von anderen Benutzern in ihrem Zustand vor der Änderung gelesen werden (das "Before Image" der Daten). Es kann jedoch Situationen geben, in denen Sie eine Gruppe von Datensätzen sperren möchten, noch bevor sie im Programm geändert werden. Oracle stellt für diesen Fall die Klausel `FOR UPDATE` der Anweisung `SELECT` bereit.

Wenn Sie eine `SELECT . . . FOR UPDATE`-Anweisung absetzen, ruft das relationale Datenbankmanagementsystem (RDBMS) automatisch exklusive Zeilensperren für alle in der Anweisung `SELECT` angegebenen Zeilen ab und stellt nur Ihnen die Datensätze zum Ändern bereit. Kein anderer Benutzer kann die Datensätze ändern, bis Sie ein `ROLLBACK` oder `COMMIT` ausführen.

Sie können die Klausel `FOR UPDATE` durch das optionale Schlüsselwort `NOWAIT` ergänzen. Damit weisen Sie den Oracle-Server an, nicht auf die Freigabe der Sperre durch einen anderen Benutzer zu warten. In diesem Fall wird die Kontrolle sofort an Ihr Programm oder Ihre SQL Developer-Umgebung zurückgegeben, und Sie können andere Aufgaben ausführen oder den Vorgang nach einer gewissen Wartezeit wiederholen. Ohne die Klausel `NOWAIT` wird Ihr Prozess so lange blockiert, bis die Tabelle wieder verfügbar ist, weil die Sperren vom anderen Benutzer durch Absetzen eines `COMMIT`- oder `ROLLBACK`-Befehls aufgehoben wurden.

Klausel FOR UPDATE – Beispiele

- Die Klausel FOR UPDATE kann in einer SELECT-Anweisung für mehrere Tabellen verwendet werden.

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK'
AND location_id = 1500
FOR UPDATE
ORDER BY e.employee_id;
```

- Sowohl Zeilen aus der Tabelle EMPLOYEES als auch aus der Tabelle DEPARTMENTS werden gesperrt.
- Ist die zu ändernde Spalte mit FOR UPDATE OF *column_name* gekennzeichnet, werden nur die Zeilen aus dieser Tabelle gesperrt.

ORACLE

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

Im Beispiel auf der Folie sperrt die Anweisung alle Zeilen in der Tabelle EMPLOYEES, bei denen JOB_ID auf ST_CLERK und LOCATION_ID auf 1500 eingestellt ist, sowie alle Zeilen in der Tabelle DEPARTMENTS, bei denen die LOCATION_ID auf 1500 eingestellt ist.

Mit FOR UPDATE OF *column_name* können Sie die zu ändernde Spalte kennzeichnen. Die OF-Liste der Klausel FOR UPDATE beschränkt Sie nicht auf die Spalten der gewählten Zeilen. Es bleiben weiterhin alle Zeilen gesperrt. Wenn Sie in der Abfrage nur FOR UPDATE angeben und nach dem Schlüsselwort OF keine Spalten einfügen, sperrt die Datenbank alle identifizierten Zeilen in allen Tabellen, die in der Klausel FROM aufgelistet sind.

Die folgende Anweisung sperrt nur die Zeilen in der Tabelle EMPLOYEES, bei denen sich ST_CLERK am Standort mit der LOCATION_ID 1500 befindet. In der Tabelle DEPARTMENTS werden keine Zeilen gesperrt:

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK' AND location_id = 1500
FOR UPDATE OF e.salary
ORDER BY e.employee_id;
```


Im folgenden Beispiel wird die Datenbank angewiesen, fünf Sekunden auf die Verfügbarkeit der Zeile zu warten und dann die Kontrolle an Sie zurückzugeben.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE WAIT 5
ORDER BY employee_id;
```

Quiz

Die folgenden Anweisungen liefern dasselbe Ergebnis:

```
DELETE FROM copy_emp;
```

```
TRUNCATE TABLE copy_emp;
```

- a. Richtig
- b. Falsch

ORACLE

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

Richtige Antwort: b

Zusammenfassung

In dieser Lektion haben Sie die Verwendung der folgenden Anweisungen kennengelernt:

Funktion	Beschreibung
INSERT	Fügt der Tabelle neue Zeilen hinzu
UPDATE	Ändert vorhandene Zeilen in der Tabelle
DELETE	Entfernt vorhandene Zeilen aus der Tabelle
TRUNCATE	Entfernt alle Zeilen aus der Tabelle
COMMIT	Schreibt alle noch nicht gespeicherten Änderungen
SAVEPOINT	Wird für ein Rollback zur Savepoint-Markierung verwendet
ROLLBACK	Verwirft alle noch nicht gespeicherten Datenänderungen
Klausel FOR UPDATE in SELECT	Sperrt die von der Abfrage SELECT identifizierten Zeilen

ORACLE

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

In dieser Lektion haben Sie gelernt, wie Sie Daten in der Oracle-Datenbank mit den Anweisungen INSERT, UPDATE, DELETE und TRUNCATE bearbeiten und Datenänderungen mit den Anweisungen COMMIT, SAVEPOINT und ROLLBACK steuern. Darüber hinaus wurde erläutert, wie Sie mit der Klausel FOR UPDATE der Anweisung SELECT Zeilen sperren, damit nur Sie in dieser Zeit Änderungen vornehmen können.

Der Oracle-Server stellt eine jederzeit konsistente Ansicht der Daten sicher.

Übungen zu Lektion 10 – Überblick

Diese Übung behandelt folgende Themen:

- Zeilen in Tabellen einfügen
- Zeilen einer Tabelle aktualisieren und löschen
- Transaktionen steuern

ORACLE

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

In dieser Übung fügen Sie der Tabelle `MY_EMPLOYEE` Zeilen hinzu. Außerdem aktualisieren und löschen Sie Tabellendaten und steuern Ihre Transaktionen. Um die Tabelle `MY_EMPLOYEE` zu erstellen, führen Sie ein Skript aus.

11

Data Definition Language – Einführung

ORACLE®

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

Ziele

Nach Ablauf dieser Lektion haben Sie folgende Ziele erreicht:

- Wichtigste Datenbankobjekte kategorisieren
- Tabellenstrukturen prüfen
- Für Spalten verfügbare Datentypen auflisten
- Einfache Tabellen erstellen
- Erstellung von Constraints im Zuge der Tabellenerstellung beschreiben
- Funktionsweise von Schemaobjekten beschreiben

ORACLE

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

In dieser Lektion lernen Sie die Anweisungen der Data Definition Language (DDL) kennen. Sie erhalten alle grundlegenden Informationen zum Erstellen, Ändern und Entfernen einfacher Tabellen. Außerdem werden die in DDL verfügbaren Datentypen beschrieben und Schema-konzepte vorgestellt. Darüber hinaus behandelt diese Lektion Constraints und geht auf die bei der Verletzung von Constraints in DML-Vorgängen generierten Ausnahmemeldungen ein.

Lektionsagenda

- **Datenbankobjekte**
 - Benennungsregeln
- CREATE TABLE-Anweisungen
- Datentypen
- Constraints im Überblick: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Tabellen mithilfe von Unterabfragen erstellen
- ALTER TABLE-Anweisungen
- DROP TABLE-Anweisungen

ORACLE

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

Datenbankobjekte

Objekt	Beschreibung
Tabelle	Grundlegende Speichereinheit, die aus Zeilen besteht
View	Stellt Teilmengen von Daten aus einzelnen oder mehreren Tabellen logisch dar
Sequence	Generiert numerische Werte
Index	Verbessert die Performance einiger Abfragen
Synonym	Gibt Objekten einen alternativen Namen

ORACLE

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

Die Oracle-Datenbank kann mehrere Datenstrukturen enthalten. Jede Struktur muss im Datenbankdesign dargestellt werden, damit sie während der Erstellungsphase der Datenbankentwicklung erzeugt werden kann.

- **Tabelle:** Speichert Daten
- **View:** Stellt eine Teilmenge von Daten aus einer oder mehreren Tabellen dar
- **Sequence:** Generiert numerische Werte
- **Index:** Verbessert die Performance einiger Abfragen
- **Synonym:** Gibt Objekten einen alternativen Namen

Oracle-Tabellenstrukturen

- Tabellen können jederzeit erstellt werden, auch wenn andere Benutzer die Datenbank gerade verwenden.
- Die Größe der Tabelle müssen Sie nicht angeben. Sie wird letztendlich durch den Speicherplatz definiert, den Sie der Datenbank als Ganzes zuweisen. Wichtig ist jedoch die Einschätzung, wie viel Speicherplatz eine Tabelle mit der Zeit belegen wird.
- Die Tabellenstruktur kann online geändert werden.

Hinweis: Es sind noch weitere Datenbankobjekte verfügbar, die jedoch in diesem Kurs nicht behandelt werden.

Benennungsregeln

Tabellennamen und Spaltennamen:

- müssen mit einem Buchstaben beginnen
- dürfen 1-30 Zeichen enthalten
- dürfen nur die folgenden Zeichen enthalten: A-Z, a-z, 0-9, _, \$, #
- dürfen nicht mit dem Namen eines anderen Objekts desselben Benutzers übereinstimmen
- dürfen nicht mit einem für Oracle reservierten Wort übereinstimmen

ORACLE

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

Benennen Sie Datenbanktabellen und -spalten gemäß den Standardbenennungsregeln für Oracle-Datenbankobjekte:

- Tabellen- und Spaltennamen müssen mit einem Buchstaben beginnen und dürfen 1-30 Zeichen umfassen.
- Namen dürfen nur die folgenden Zeichen enthalten: A-Z, a-z, 0-9, _ (Unterstrich), \$ und # (gültige Zeichen, deren Verwendung jedoch nicht empfohlen wird).
- Namen dürfen nicht mit dem Namen eines anderen Objekts übereinstimmen, das demselben Oracle-Benutzer gehört.
- Es dürfen keine für Oracle reservierten Wörter als Namen verwendet werden.
 - Sie können den Namen eines Objekts auch mit in Anführungszeichen gesetzten IDs darstellen. Eine solche ID beginnt und endet mit doppelten Anführungszeichen (" "). Wenn Sie ein Schemaobjekt mit einer in Anführungszeichen gesetzten ID benennen, müssen die doppelten Anführungszeichen bei jeder Referenz auf das Objekt angegeben werden. Auch reservierte Wörter können als ID in Anführungszeichen verwendet werden. Von dieser Vorgehensweise wird jedoch abgeraten.

Benennungsrichtlinien

Wählen Sie für Tabellen und andere Datenbankobjekte aussagekräftige Namen.

Hinweis: Bei den Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

EMPLOYEES wird als gleichbedeutend mit Schreibweisen wie eMPloyees und eMpLOYEES interpretiert. Bei den in Anführungszeichen gesetzten IDs muss die Groß-/Kleinschreibung jedoch beachtet werden.

Weitere Informationen finden Sie in der *Oracle Database SQL Language Reference* für Oracle Database 12g im Abschnitt "Schema Object Names and Qualifiers".

Lektionsagenda

- Datenbankobjekte
 - Benennungsregeln
- CREATE TABLE-Anweisungen
- Datentypen
- Constraints im Überblick: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Tabellen mithilfe von Unterabfragen erstellen
- ALTER TABLE-Anweisungen
- DROP TABLE-Anweisungen

ORACLE

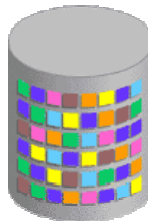
Copyright © 2014, Oracle und/oder verbundene Unternehmen. All rights reserved. Alle Rechte vorbehalten.

CREATE TABLE-Anweisungen

- Erforderlich sind:
 - Berechtigung CREATE TABLE
 - Ein Speicherbereich

```
CREATE TABLE [schema.]table  
    (column datatype [DEFAULT expr] [, ...]);
```

- Anzugeben sind:
 - Tabellenname
 - Name, Datentyp und Größe der Spalte



ORACLE

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

Mit der Anweisung `CREATE TABLE` erstellen Sie Tabellen, in die Daten gespeichert werden können. Diese Anweisung gehört zu den DDL-Anweisungen aus der Teilmenge von SQL-Anweisungen, mit denen Oracle-Datenbankstrukturen erstellt, geändert oder entfernt werden. Diese Anweisungen wirken sich unmittelbar auf die Datenbank aus und zeichnen zudem Informationen im Data Dictionary auf.

Um eine Tabelle zu erstellen, muss ein Benutzer über die Berechtigung `CREATE TABLE` und einen Speicherbereich verfügen, in dem die Objekte erstellt werden sollen. Der Datenbankadministrator (DBA) weist Benutzern mithilfe von DCL-(Data Control Language-)Anweisungen Berechtigungen zu.

Für die Syntax gilt:

<i>schema</i>	Entspricht dem Namen des Eigentümers
<i>table</i>	Steht für den Namen der Tabelle
DEFAULT <i>expr</i>	Gibt einen Standardwert an, wenn in der Anweisung INSERT kein Wert angegeben wurde
<i>column</i>	Name der Spalte
<i>datatype</i>	Datentyp und Länge der Spalte

Hinweis: Um Tabellen in Schemas zu erstellen, die nicht das Schema des Benutzers sind, benötigen Sie die Berechtigung `CREATE ANY TABLE`.

Tabellen erstellen

- Tabelle erstellen:

```
CREATE TABLE dept
  (deptno      NUMBER(2),
   dname       VARCHAR2(14),
   loc         VARCHAR2(13),
   create_date DATE DEFAULT SYSDATE);
```

table DEPT created.

- Prüfen, ob die Tabelle erstellt wurde:

```
DESCRIBE dept
```

```
DESCRIBE dept
Name          Null Type
-----
DEPTNO        NUMBER(2)
DNAME         VARCHAR2(14)
LOC           VARCHAR2(13)
CREATE_DATE    DATE
```

ORACLE

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

Im Beispiel auf der Folie wird die Tabelle DEPT mit vier Spalten erstellt: DEPTNO, DNAME, LOC und CREATE_DATE. Die Spalte CREATE_DATE hat einen Standardwert. Wenn Sie für eine INSERT-Anweisung keinen Wert bereitstellen, wird automatisch das Systemdatum eingefügt.

Um zu prüfen, ob die Tabelle erstellt wurde, führen Sie den Befehl DESCRIBE aus.

Da eine Tabelle mit einer DDL-Anweisung erstellt wird, wird nach Ausführung dieser Anweisung ein automatischer Commit durchgeführt.

Hinweis: Sie können die Liste Ihrer eigenen Tabellen anzeigen, indem Sie das Data Dictionary abfragen. Beispiel:

```
select table_name from user_tables;
```

Mithilfe von Data Dictionary Views können Sie auch Informationen über andere Datenbankobjekte wie Views, Indizes und so weiter suchen. Data Dictionarys werden im Kurs *Oracle Database: SQL Workshop II* ausführlich behandelt.

Lektionsagenda

- Datenbankobjekte
 - Benennungsregeln
- CREATE TABLE-Anweisungen
- **Datentypen**
- Constraints im Überblick: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Tabellen mithilfe von Unterabfragen erstellen
- ALTER TABLE-Anweisungen
- DROP TABLE-Anweisungen

ORACLE®

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

Datentypen

Datentyp	Beschreibung
VARCHAR2(<i>size</i>)	Zeichendaten variabler Länge
CHAR(<i>size</i>)	Zeichendaten fester Länge
NUMBER(<i>p</i> , <i>s</i>)	Numerische Daten variabler Länge
DATE	Datums- und Uhrzeitwerte
LONG	Zeichendaten variabler Länge (bis zu 2 GB)
CLOB	Maximale Größe (4 GB - 1) * (DB_BLOCK_SIZE)
RAW and LONG RAW	RAW-Binärdaten
BLOB	Maximale Größe (4 GB - 1) * (Initialisierungsparameter DB_BLOCK_SIZE (8 TB bis 128 TB))
BFILE	In einer externen Datei gespeicherte Binärdaten (bis zu 4 GB)
ROWID	Base-64-Code, der die eindeutige Adresse einer Zeile in einer Tabelle darstellt

ORACLE

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

Beim Angeben einer Tabellenspalte müssen Sie einen Datentyp zuweisen. Verschiedene Datentypen sind verfügbar:

Datentyp	Beschreibung
VARCHAR2(<i>size</i>)	Zeichendaten variabler Länge (Für <i>size</i> muss eine Maximalgröße angegeben werden: Die <i>Mindestgröße</i> lautet 1.) Die Maximalgröße lautet: <ul style="list-style-type: none"> 32767 Byte, wenn MAX_SQL_STRING_SIZE = EXTENDED 4000 Byte, wenn MAX_SQL_STRING_SIZE = LEGACY
CHAR [(<i>size</i>)]	Zeichendaten fester Länge mit der Größe <i>size</i> in Byte (Standard- und Mindestgröße 1 Byte, <i>Maximalgröße</i> 2.000 Byte)
NUMBER [(<i>p</i> , <i>s</i>)]	Wert mit der Gesamtstellenzahl <i>p</i> und der Anzahl der Nachkommastellen <i>s</i> (Die Gesamtstellenzahl darf 1 bis 38 und die Anzahl der Nachkommastellen -84 bis 127 betragen.)
DATE	Datums- und Uhrzeitwerte bis auf die nächste Sekunde vom 1. Januar 4712 v. Chr. bis zum 31. Dezember 9999 n. Chr.

Datentyp	Beschreibung
LONG	Zeichendaten variabler Länge (bis zu 2 GB)
CLOB	Character Large Object (großes Objekt vom Typ CHAR), das Einzel- oder Mehr-Byte-Zeichen enthält. Maximale Größe: $(4 \text{ GB} - 1) * (\text{DB_BLOCK_SIZE})$; speichert Daten des länderspezifischen Zeichensatzes
NCLOB	Großes Objekt vom Typ CHAR, das Unicode-Zeichen enthält. Es werden Zeichensätze mit fester und variabler Breite unterstützt, beide verwenden den nationalen Zeichensatz der Datenbank. Maximale Größe: $(4 \text{ GB} - 1) * (\text{Datenbankblockgröße})$; speichert Daten des länderspezifischen Zeichensatzes
RAW(size)	RAW-Binärdaten mit der Größe <i>size</i> in Byte. Für RAW-Werte muss eine Größe (<i>size</i>) angegeben werden: Maximalgröße: 32767 Byte, wenn <code>MAX_SQL_STRING_SIZE = EXTENDED</code> 4000 Byte, wenn <code>MAX_SQL_STRING_SIZE = LEGACY</code>
LONG RAW	RAW-Binärdaten variabler Länge (bis zu 2 GB)
BLOB	Großes Binärobjekt (Binary Large Object). Maximale Größe $(4 \text{ GB} - 1) * (\text{Initialisierungsparameter DB_BLOCK_SIZE})$ (8 TB bis 128 TB))
BFILE	In einer externen Datei gespeicherte Binärdaten (bis zu 4 GB)
ROWID	Base 64-Zeichenfolge für die eindeutige Adresse einer Zeile in der Tabelle. Wird in erster Linie für Werte verwendet, die von der Pseudospalte ROWID zurückgegeben werden.

Richtlinien

- Spalten vom Typ LONG werden nicht kopiert, wenn Tabellen mithilfe von Unterabfragen erstellt werden.
- Spalten vom Typ LONG dürfen nicht in eine GROUP BY- oder eine ORDER BY-Klausel aufgenommen werden.
- Pro Tabelle darf nur eine Spalte vom Typ LONG verwendet werden.
- Für Spalten vom Typ LONG können keine Constraints definiert werden.
- Anstelle einer Spalte vom Typ LONG können Sie eine Spalte vom Typ CLOB verwenden.

Datetime-Datentypen

Sie können verschiedene Datetime-Datentypen verwenden:

Datentyp	Beschreibung
TIMESTAMP	Datum mit Sekundenbruchteilen
INTERVAL YEAR TO MONTH	Wird als Intervall von Jahren und Monaten gespeichert
INTERVAL DAY TO SECOND	Wird als Intervall von Tagen, Stunden, Minuten und Sekunden gespeichert



ORACLE

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

Datentyp	Beschreibung
TIMESTAMP	Ermöglicht das Speichern der Zeit als Datum mit Sekundenbruchteilen. Speichert das Jahr, den Monat, den Tag, die Minute und den Sekundenwert des DATE-Datentyps sowie den Wert des Sekundenbruchteils. Es sind zahlreiche Variationen dieses Datentyps wie WITH TIMEZONE und WITH LOCALTIMEZONE verfügbar.
INTERVAL YEAR TO MONTH	Ermöglicht das Speichern der Zeit als Intervall von Jahren und Monaten. Wird verwendet, um den Unterschied zwischen zwei Datetime-Werten darzustellen, bei denen nur das Jahr und der Monat von Bedeutung sind.
INTERVAL DAY TO SECOND	Ermöglicht das Speichern der Zeit als Intervall von Tagen, Stunden, Minuten und Sekunden. Wird verwendet, um den Unterschied zwischen zwei Datetime-Werten präzise darzustellen.

Hinweis: Diese Datetime-Datentypen sind in allen Releases ab Oracle9i verfügbar. Die Datetime-Datentypen werden im Kurs *Oracle Database: SQL Workshop II* in der Lektion "Daten in verschiedenen Zeitzonen verwalten" ausführlich behandelt.

Weitere Informationen zu den Datetime-Datentypen finden Sie auch in der *Oracle Database SQL Language Reference* für Oracle Database 12c in den Abschnitten "TIMESTAMP Datatype", "INTERVAL YEAR TO MONTH Datatype" und "INTERVAL DAY TO SECOND Datatype".

Option DEFAULT

- Beim Erstellen der Tabelle mit `CREATE` einen Standardwert angeben

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Zulässige Werte sind Literalwerte, Ausdrücke und SQL-Funktionen.
- Unzulässige Werte sind Namen anderer Spalten und Pseudospalten.
- Der Standarddatentyp muss dem Spaltentyp entsprechen.

```
CREATE TABLE hire_dates  
  (id          NUMBER(8),  
   hire_date DATE DEFAULT SYSDATE);  
table HIRE_DATES created.
```

ORACLE

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

Beim Definieren einer Tabelle können Sie mit der Option `DEFAULT` festlegen, dass eine Spalte einen Standardwert erhalten soll. Diese Option verhindert das Einlesen von Nullwerten, wenn Zeilen eingefügt werden, deren Spalten keinen Wert enthalten. Der Standardwert kann ein Literal, ein Ausdruck oder eine SQL-Funktion wie `SYSDATE` oder `USER` sein. Er darf jedoch nicht der Name einer anderen Spalte oder eine Pseudospalte wie `NEXTVAL` oder `CURRVAL` sein. Der Standardausdruck muss dem Datentyp der Spalte entsprechen.

Hierzu zwei Beispiele:

```
INSERT INTO hire_dates values(45, NULL);
```

Die vorstehende Anweisung fügt anstelle des Standardwertes den Nullwert ein.

```
INSERT INTO hire_dates(id) values(35);
```

Die vorstehende Anweisung fügt `SYSDATE` für die Spalte `HIRE_DATE` ein.

Hinweis: Um die DDL-Anweisungen auszuführen, klicken Sie in SQL Developer auf das Symbol **Run Script** oder drücken F5. Die Rückmeldungen werden in der Registerkarte **Script Output** angezeigt.

Lektionsagenda

- Datenbankobjekte
 - Benennungsregeln
- CREATE TABLE-Anweisungen
- Datentypen
- **Constraints im Überblick: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK**
- Tabellen mithilfe von Unterabfragen erstellen
- ALTER TABLE-Anweisungen
- DROP TABLE-Anweisungen

ORACLE®

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

Constraints hinzufügen

- Constraints erzwingen Regeln auf Tabellenebene.
- Constraints stellen die Konsistenz und Integrität der Datenbank sicher.
- Die folgenden Constraint-Typen sind gültig:
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK



ORACLE®

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

Der Oracle-Server verwendet Constraints, um die Eingabe ungültiger Daten in Tabellen zu verhindern.

Constraints dienen folgenden Zwecken:

- Beim Einfügen, Aktualisieren oder Löschen von Tabellenzeilen Regeln für die Tabellendaten durchsetzen. Das Constraint muss erfüllt werden, damit der Vorgang erfolgreich durchgeführt werden kann.
- Löschen von Tabelle verhindern, wenn Abhängigkeiten mit anderen Tabellen bestehen

Datenintegritäts-Constraints

Constraint	Beschreibung
NOT NULL	Gibt an, dass die Spalte keinen Nullwert enthalten darf
UNIQUE	Gibt eine Spalte oder Spaltenkombination an, deren Werte in allen Zeilen der Tabelle eindeutig sein müssen
PRIMARY KEY	Identifiziert jede Zeile einer Tabelle eindeutig
FOREIGN KEY	Erstellt und erzwingt eine referenzielle Integrität zwischen der Spalte und einer Spalte der referenzierten Tabelle, bei der die Werte in einer Tabelle mit den Werten in einer anderen Tabelle übereinstimmen müssen
CHECK	Legt eine Bedingung fest, die erfüllt sein muss

Constraints – Richtlinien

- Sie können ein Constraint benennen. Andernfalls generiert der Oracle-Server einen Namen im Format `SYS_Cn`.
- Constraints erstellen:
 - Während der Erstellung der Tabelle
 - Nach der Erstellung der Tabelle
- Constraints auf Spalten- oder Tabellenebene definieren
- Constraints im Data Dictionary anzeigen

ORACLE

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

Alle Constraints werden im Data Dictionary gespeichert. Constraints lassen sich leichter referenzieren, wenn Sie ihnen aussagekräftige Namen geben. Constraint-Namen müssen den Standardbenennungsregeln für Objekte entsprechen. Der Name darf allerdings nicht mit dem Namen eines anderen Objekts identisch sein, das dem gleichen Benutzer gehört. Wenn Sie das Constraint nicht benennen, erzeugt der Oracle-Server einen Namen im Format `SYS_Cn`. Dabei ist *n* eine Ganzzahl, damit der Constraint-Name eindeutig ist.

Constraints können zusammen mit der Tabelle erstellt oder anschließend definiert werden. Sie können Constraints auf Spalten- oder Tabellenebene definieren. Ein Constraint auf Tabellenebene ist mit einem Constraint auf Spaltenebene funktionell identisch.

Weitere Informationen finden Sie in der *Oracle Database SQL Language Reference* für Oracle Database 12c im Abschnitt "Constraints".

Constraints definieren

- Syntax:

```
CREATE TABLE [schema.]table
  (column datatype [DEFAULT expr]
   [column_constraint],
   ...
   [table_constraint][, ...]);
```

- Syntax eines Constraints auf Spaltenebene:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Syntax eines Constraints auf Tabellenebene:

```
column, ...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

ORACLE

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

Die Folie zeigt die Syntax für die Definition von Constraints beim Erstellen einer Tabelle. Sie können Constraints auf Spalten- oder Tabellenebene definieren. Auf Spaltenebene definierte Constraints werden beim Definieren der Spalte hinzugefügt. Auf Tabellenebene definierte Constraints werden am Ende der Tabellendefinition hinzugefügt und müssen in Klammern die Spalten referenzieren, für die sie gelten sollen. Der Unterschied besteht hauptsächlich in der Syntax. Ansonsten sind Constraints auf Spalten- und Tabellenebene funktionell identisch.

Constraints vom Typ NOT NULL können ausschließlich auf Spaltenebene definiert werden.

Constraints, die mehrere Spalten betreffen, müssen auf Tabellenebene definiert werden.

Für die Syntax gilt:

<i>schema</i>	Entspricht dem Namen des Eigentümers
<i>table</i>	Steht für den Namen der Tabelle
DEFAULT <i>expr</i>	Gibt einen zu verwendenden Standardwert an, wenn in der Anweisung INSERT kein Wert angegeben wurde
<i>column</i>	Name der Spalte
<i>datatype</i>	Datentyp und Länge der Spalte
<i>column_constraint</i>	Integritäts-Constraint als Teil der Spaltendefinition
<i>table_constraint</i>	Integritäts-Constraint als Teil der Tabellendefinition

Constraints definieren

- Beispiel für ein Constraint auf Spaltenebene:

```
CREATE TABLE employees(  
  employee_id  NUMBER(6)  
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,  
  first_name   VARCHAR2(20),  
  ...);
```

1

- Beispiel für ein Constraint auf Tabellenebene:

```
CREATE TABLE employees(  
  employee_id  NUMBER(6),  
  first_name   VARCHAR2(20),  
  ...  
  job_id       VARCHAR2(10) NOT NULL,  
  CONSTRAINT emp_emp_id_pk  
    PRIMARY KEY (EMPLOYEE_ID));
```

2

ORACLE

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

Constraints werden in der Regel zusammen mit der Tabelle erstellt. Sie können aber ebenso nach dem Erstellen einer Tabelle hinzugefügt und auch vorübergehend deaktiviert werden.

In beiden Beispielen auf der Folie wird ein Constraint vom Typ `PRIMARY KEY` für die Spalte `EMPLOYEE_ID` der Tabelle `EMPLOYEES` erstellt.

1. Im ersten Beispiel wird die Syntax für die Spaltenebene verwendet, um das Constraint zu definieren.
2. Im zweiten Beispiel wird die Syntax für die Tabellenebene verwendet, um das Constraint zu definieren.

Weitere Informationen zu Constraints vom Typ `PRIMARY KEY` erhalten Sie im weiteren Verlauf dieser Lektion.

Constraint NOT NULL

Stellt sicher, dass für die Spalte keine Nullwerte akzeptiert werden:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	2600	(null)	50	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	2500	(null)	50	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	10500	0.2	80	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	11000	0.3	80	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	8600	0.2	80	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	7000	0.15	(null)	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	4400	(null)	10	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	13000	(null)	20	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	6000	(null)	20	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	12000	(null)	110	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	8300	(null)	110	WGIEZT	515.123.8181	07-JUN-94

↑
Constraint NOT NULL
(Primärschlüssel erzwingt
Constraint NOT NULL.)

↑
NOT NULL-
Constraint

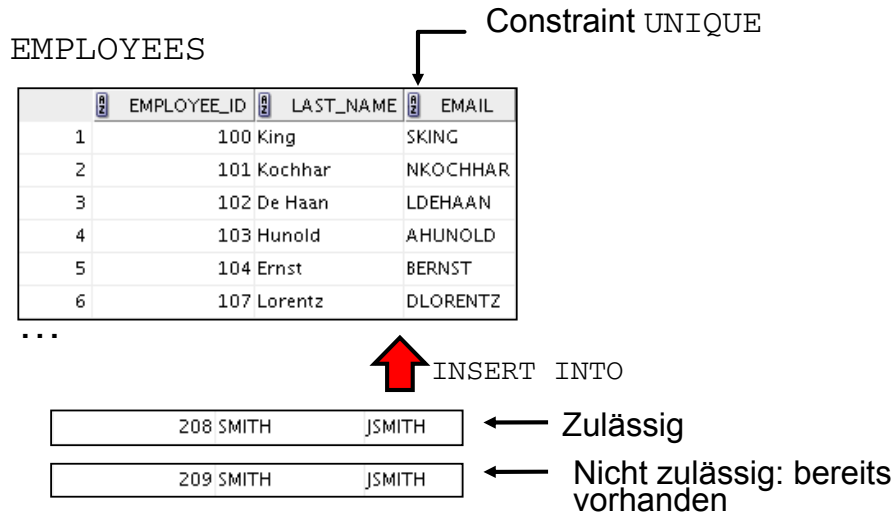
↑
Kein NOT NULL-Constraint (Jede
Zeile darf in dieser Spalte einen
Nullwert enthalten.)

ORACLE

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

Das Constraint NOT NULL stellt sicher, dass die Spalte keine Nullwerte enthält. Spalten ohne NOT NULL-Constraint können standardmäßig Nullwerte enthalten. Constraints vom Typ NOT NULL müssen auf Spaltenebene definiert werden. In der Tabelle EMPLOYEES erbt die Spalte EMPLOYEE_ID ein NOT NULL-Constraint, da sie als Primärschlüssel definiert ist. Daneben wird das Constraint NOT NULL für die Spalten LAST_NAME, EMAIL, HIRE_DATE und JOB_ID erzwungen.

Constraint UNIQUE



ORACLE

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

Bei Schlüsselintegritäts-Constraints vom Typ `UNIQUE` muss jeder Wert in einer Spalte oder Spaltengruppe (Schlüssel) eindeutig sein. Das bedeutet, dass die angegebenen Spalten oder Spaltengruppen zweier Tabellenzeilen keine doppelten Werte enthalten dürfen. Die Spalte (oder Spaltengruppe), die in der Definition des Constraints `UNIQUE` enthalten ist, wird als *eindeutiger Schlüssel* bezeichnet. Wenn das Constraint `UNIQUE` mehrere Spalten umfasst, wird diese Gruppe als *zusammengesetzter eindeutiger Schlüssel* bezeichnet.

Constraints vom Typ `UNIQUE` erlauben die Eingabe von Nullwerten, sofern keine Constraints vom Typ `NOT NULL` für dieselben Spalten definiert sind. Tatsächlich können beliebig viele Zeilen Nullwerte in Spalten ohne `NOT NULL`-Constraints enthalten, da Nullwerte keinen anderen Werten entsprechen und daher stets das für eine Spalte (oder alle Spalten eines zusammengesetzten eindeutigen Schlüssels) festgelegte Constraint `UNIQUE` erfüllen.

Hinweis: Aufgrund des für `UNIQUE`-Constraints für mehrere Spalten gültigen Suchmechanismus dürfen die Werte in den Nicht-Null-Spalten eines zusammengesetzten `UNIQUE`-Schlüssel-Constraints, das auch Nullspalten enthält, nicht identisch sein.

Constraint UNIQUE

Auf Tabellen- oder Spaltenebene definieren:

```
CREATE TABLE employees (  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

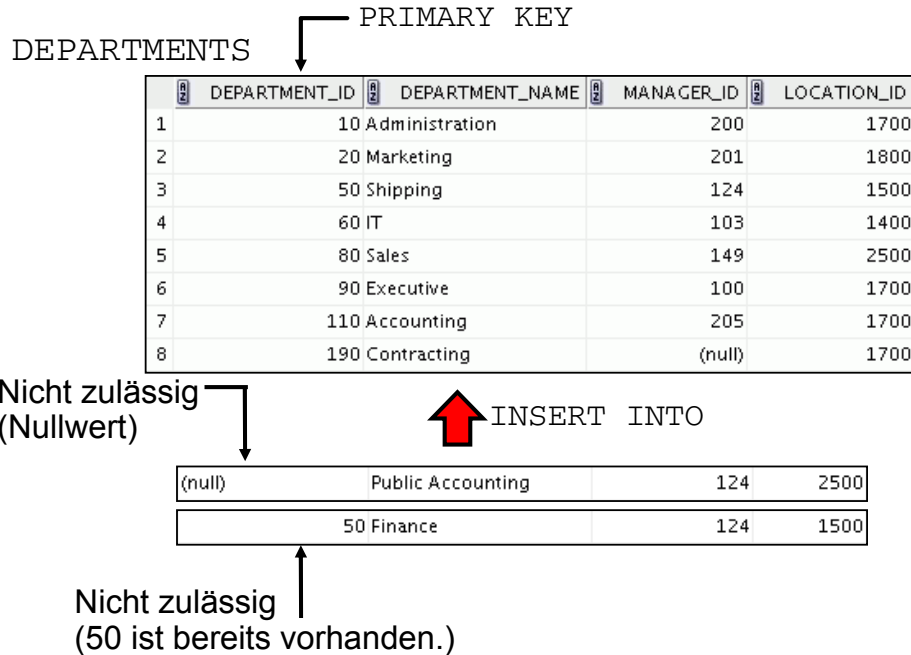
Copyright © 2014, Oracle und/oder verbundene Unternehmen. All rights reserved. Alle Rechte vorbehalten.

Constraints vom Typ `UNIQUE` können auf Spalten- oder Tabellenebene definiert werden. Wenn Sie einen zusammengesetzten eindeutigen Schlüssel erstellen möchten, definieren Sie das Constraint auf Tabellenebene. Ein zusammengesetzter Schlüssel wird definiert, wenn kein einzelnes Attribut vorhanden ist, das eine Zeile eindeutig identifizieren kann. In diesem Fall kann sich der eindeutige Schlüssel aus zwei oder mehr Spalten zusammensetzen. Der kombinierte Wert ist dabei stets eindeutig und kann Zeilen identifizieren.

Im Beispiel auf der Folie wird das Constraint `UNIQUE` auf die Spalte `EMAIL` der Tabelle `EMPLOYEES` angewendet. Der Name des Constraints ist `EMP_EMAIL_UK`.

Hinweis: Der Oracle-Server setzt das Constraint `UNIQUE` durch, indem für die eindeutigen Schlüsselspalten implizit ein eindeutiger Index erstellt wird.

Constraint PRIMARY KEY



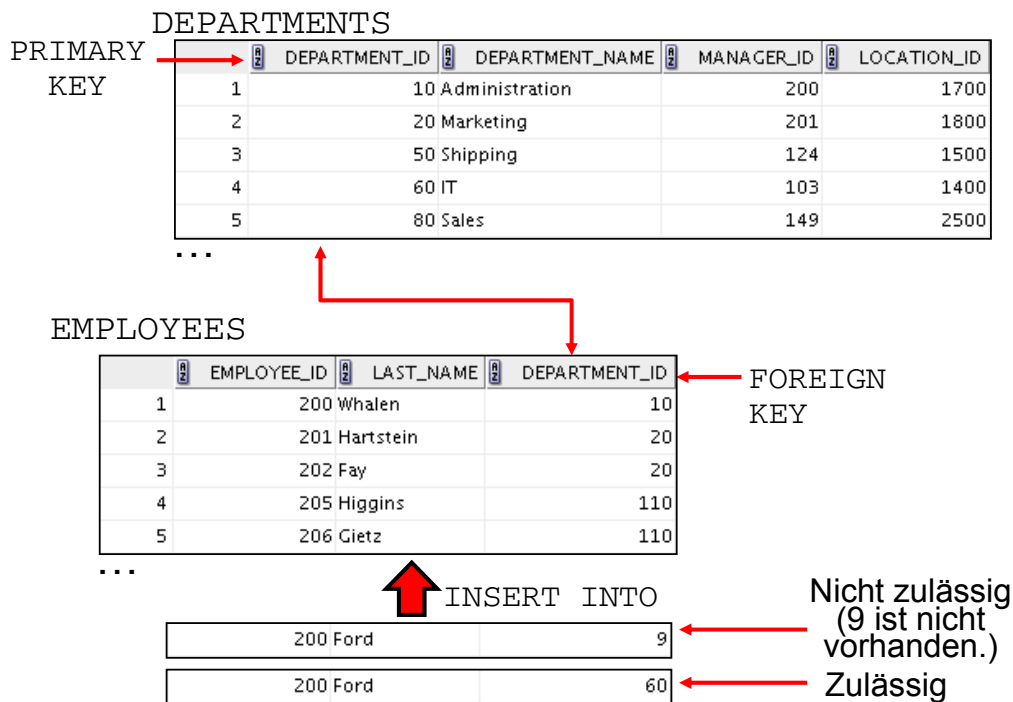
ORACLE

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

Constraints vom Typ `PRIMARY KEY` erstellen für die Tabelle einen Primärschlüssel. Für jede Tabelle kann nur ein Primärschlüssel erstellt werden. Das Constraint `PRIMARY KEY` ist eine Spalte oder Spaltengruppe, die jede Zeile einer Tabelle eindeutig identifiziert. Dieses Constraint setzt die Eindeutigkeit der Spalte oder Spaltenkombination durch und stellt sicher, dass keine Spalte, die zu einem Primärschlüssel gehört, einen Nullwert enthält.

Hinweis: Da die Eindeutigkeit Teil der Definition des Constraints `PRIMARY KEY` ist, setzt der Oracle-Server die Eindeutigkeit durch, indem für die `PRIMARY KEY`-Spalten implizit ein eindeutiger Index erstellt wird.

Constraint FOREIGN KEY



ORACLE

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

Constraints vom Typ `FOREIGN KEY` (oder referenzielle Integritäts-Constraints) bestimmen eine Spalte oder Spaltenkombination als Fremdschlüssel und richten eine Beziehung mit einem Primär- oder einem eindeutigen Schlüssel in derselben oder einer anderen Tabelle ein.

Im Beispiel auf der Folie wurde `DEPARTMENT_ID` als Fremdschlüssel in der (abhängigen oder untergeordneten) Tabelle `EMPLOYEES` definiert. Dieser referenziert die Spalte `DEPARTMENT_ID` der (referenzierten oder übergeordneten) Tabelle `DEPARTMENTS`.

Richtlinien

- Ein Fremdschlüsselwert muss einem vorhandenen Wert in der übergeordneten Tabelle entsprechen oder `NULL` sein.
- Fremdschlüssel basieren auf Datenwerten und sind rein logische (nicht physische) Zeiger.

Constraint FOREIGN KEY

Auf Tabellen- oder Spaltenebene definieren:

```
CREATE TABLE employees(  
    employee_id    NUMBER(6),  
    last_name      VARCHAR2(25) NOT NULL,  
    email          VARCHAR2(25),  
    salary         NUMBER(8,2),  
    commission_pct NUMBER(2,2),  
    hire_date      DATE NOT NULL,  
    ...  
    department_id  NUMBER(4),  
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)  
        REFERENCES departments(department_id),  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

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

Constraints vom Typ `FOREIGN KEY` können auf Spalten- oder Tabellenebene definiert werden. Ein zusammengesetzter Fremdschlüssel muss mithilfe der Definition auf Tabellenebene erstellt werden.

Im Beispiel auf der Folie wird mithilfe der Syntax für die Tabellenebene ein `FOREIGN KEY`-Constraint für die Spalte `DEPARTMENT_ID` der Tabelle `EMPLOYEES` definiert. Der Name des Constraints ist `EMP_DEPT_FK`.

Der Fremdschlüssel kann auch auf Spaltenebene definiert werden. Dazu muss das Constraint auf einer einzelnen Spalte basieren. Die Syntax unterscheidet sich dahingehend, dass das Schlüsselwort `FOREIGN KEY` nicht vorhanden ist. Beispiel:

```
CREATE TABLE employees  
(  
    ...  
    department_id NUMBER(4) CONSTRAINT emp_deptid_fk  
        REFERENCES departments(department_id),  
    ...  
)
```

Constraint FOREIGN KEY – Schlüsselwörter

- FOREIGN KEY: Definiert die Spalte der untergeordneten Tabelle auf Tabellen-Constraint-Ebene
- REFERENCES: Identifiziert die Tabelle und Spalte in der übergeordneten Tabelle
- ON DELETE CASCADE: Löscht die abhängigen Zeilen aus der untergeordneten Tabelle, wenn eine Zeile in der übergeordneten Tabelle gelöscht wird
- ON DELETE SET NULL: Konvertiert abhängige Fremdschlüsselwerte in Nullwerte

ORACLE

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

Der Fremdschlüssel wird in der untergeordneten Tabelle definiert. Die Tabelle mit der referenzierten Spalte ist die übergeordnete Tabelle. Der Fremdschlüssel wird mithilfe einer Kombination der folgenden Schlüsselwörter definiert:

- FOREIGN KEY definiert die Spalte in der untergeordneten Tabelle auf Tabellen-Constraint-Ebene.
- REFERENCES identifiziert die Tabelle und Spalte in der übergeordneten Tabelle.
- ON DELETE CASCADE gibt an, dass beim Löschen einer Zeile aus der übergeordneten Tabelle die abhängigen Zeilen aus der untergeordneten Tabelle ebenfalls gelöscht werden.
- ON DELETE SET NULL gibt an, dass beim Löschen einer Zeile aus der übergeordneten Tabelle die Fremdschlüsselwerte auf NULL eingestellt werden.

Das Standardverhalten wird *Beschränkungsregel* genannt und lässt das Aktualisieren oder Löschen referenzierter Daten nicht zu.

Ohne die Optionen ON DELETE CASCADE und ON DELETE SET NULL kann die Zeile in der übergeordneten Tabelle nicht gelöscht werden, wenn sie in der untergeordneten Tabelle referenziert wird. Die Schlüsselwörter können nicht in einer Syntax für die Spaltenebene verwendet werden.

Constraint CHECK

- Definiert eine Bedingung, die jede Zeile erfüllen muss
- Kann keine Spalten aus anderen Tabellen referenzieren

```
..., salary NUMBER(2)  
    CONSTRAINT emp_salary_min  
        CHECK (salary > 0),...
```

ORACLE

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

Das Constraint `CHECK` definiert eine Bedingung, die jede Zeile erfüllen muss. Dies ist dann der Fall, wenn die Bedingung für jede Zeile in der Tabelle entweder `TRUE` lautet oder unbekannt ist (aufgrund eines Nullwertes).

Die Bedingung kann dieselben Konstrukte verwenden wie Abfragebedingungen. Ausgenommen hiervon sind Abfragen, die andere Werte in anderen Zeilen referenzieren.

Eine einzelne Spalte darf über mehrere Constraints vom Typ `CHECK` verfügen, die in ihrer Definition auf die Spalte verweisen. Sie können für eine Spalte beliebig viele Constraints vom Typ `CHECK` definieren.

Constraints vom Typ `CHECK` können auf Spalten- oder Tabellenebene definiert werden.

```
CREATE TABLE employees  
(  
    ...  
    salary NUMBER(8,2) CONSTRAINT emp_salary_min  
        CHECK (salary > 0),  
    ...  
)
```


CREATE TABLE – Beispiel

```
CREATE TABLE teach_emp (  
    empno      NUMBER(5) PRIMARY KEY,  
    ename      VARCHAR2(15) NOT NULL,  
    job        VARCHAR2(10),  
    mgr        NUMBER(5),  
    hiredate   DATE DEFAULT (sysdate),  
    photo      BLOB,  
    sal        NUMBER(7,2),  
    deptno     NUMBER(3) NOT NULL  
                CONSTRAINT admin_dept_fkey REFERENCES  
                departments(department_id));
```

ORACLE®

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

Das Beispiel auf der Folie zeigt die Anweisung, mit der die Tabelle TEACH_EMP erstellt wird.

Constraints verletzen

```
UPDATE employees
SET   department_id = 55
WHERE department_id = 110;
```

Error starting at line 1 in command: UPDATE employees SET department_id = 55 WHERE department_id = 110	
Error report: SQL Error: ORA-02291: integrity constraint (ORA1.EMP_DEPT_FK) violated - parent key not found 02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found" *Cause: A foreign key value has no matching primary key value. *Action: Delete the foreign key or add a matching primary key.	

Abteilung 55 ist nicht vorhanden.

ORACLE

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

Wenn Constraints für Spalten vorliegen, wird beim Versuch einer Verletzung gegen die Constraint-Regel ein Fehler zurückgegeben. Beispiel: Wenn Sie versuchen, einen Datensatz mit einem Wert zu aktualisieren, der an ein Integritäts-Constraint gebunden ist, wird ein Fehler zurückgegeben.

Im Beispiel auf der Folie fehlt die Abteilung 55 in der übergeordneten Tabelle `DEPARTMENTS`. Deshalb wird der Fehler `ORA-02291` ("parent key not found") angezeigt.

Constraints verletzen

Zeilen mit einem Primärschlüssel, der in einer anderen Tabelle als Fremdschlüssel dient, können nicht gelöscht werden.

```
DELETE FROM departments
WHERE department_id = 60;
```

Error starting at line 1 in command: DELETE FROM departments WHERE department_id = 60	
Error report: SQL Error: ORA-02292: integrity constraint (ORA1.JHIST_DEPT_FK) violated - child record found 02292. 00000 - "integrity constraint (%s.%s) violated - child record found" *Cause: attempted to delete a parent key value that had a foreign dependency. *Action: delete dependencies first then parent or disable constraint.	

ORACLE

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

Wenn Sie versuchen, einen Datensatz mit einem Wert zu löschen, der an ein Integritäts-Constraint gebunden ist, wird eine Fehlermeldung zurückgegeben.

Im Beispiel auf der Folie wird versucht, die Abteilung 60 aus der Tabelle `DEPARTMENTS` zu löschen. Dies führt jedoch zu einem Fehler, da die Abteilungsnummer in der Tabelle `EMPLOYEES` als Fremdschlüssel dient. Wenn der übergeordnete Datensatz, den Sie zu löschen versuchen, untergeordnete Datensätze enthält, wird die Fehlermeldung `ORA-02292` ("child record found") angezeigt.

Die folgende Anweisung ist gültig, da es in Abteilung 70 keine Mitarbeiter gibt:

```
DELETE FROM departments
WHERE department_id = 70;
```

```
1 rows deleted
```

Lektionsagenda

- Datenbankobjekte
 - Benennungsregeln
- Datentypen
- CREATE TABLE-Anweisungen
- Constraints im Überblick: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- **Tabellen mithilfe von Unterabfragen erstellen**
- ALTER TABLE-Anweisungen
- DROP TABLE-Anweisungen

ORACLE®

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

Tabellen mithilfe von Unterabfragen erstellen

- Um eine Tabelle zu erstellen und Zeilen einzufügen, die Anweisung `CREATE TABLE` mit der Option `AS subquery` kombinieren

```
CREATE TABLE table
      [(column, column...)]
AS subquery;
```

- Anzahl der angegebenen Spalten mit der Anzahl der Spalten in der Unterabfrage abgleichen
- Spalten mit Spaltennamen und Standardwerten definieren

ORACLE®

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

Eine zweite Methode zur Erstellung einer Tabelle bietet die Klausel `AS subquery`, die sowohl die Tabelle erstellt als auch die von der Unterabfrage zurückgegebenen Zeilen einfügt.

Für die Syntax gilt:

<i>table</i>	Steht für den Namen der Tabelle
<i>column</i>	Spaltenname, Standardwert und Integritäts-Constraint
<i>subquery</i>	Anweisung <code>SELECT</code> zur Definition der in die neue Tabelle einzufügenden Zeilenmenge

Richtlinien

- Die Tabelle wird mit den angegebenen Spaltennamen erstellt. Die von der Anweisung `SELECT` abgerufenen Zeilen werden in die Tabelle eingefügt.
- Die Spaltendefinition darf nur den Spaltennamen und den Standardwert enthalten.
- Sind Spaltenspezifikationen angegeben, muss die Anzahl der Spalten der Anzahl der Spalten in der `SELECT`-Liste der Unterabfrage entsprechen.
- Sind keine Spaltenspezifikationen angegeben, entsprechen die Spaltennamen der Tabelle den Spaltennamen in der Unterabfrage.
- Die Definitionen des Spaltentyps und das Constraint `NOT NULL` werden an die neue Tabelle übergeben. Nur das explizite Constraint `NOT NULL` wird vererbt. Die Spalte vom Typ `PRIMARY KEY` übergibt das Feature `NOT NULL` nicht an die neue Spalte. Auch andere Constraint-Regeln werden nicht an die neue Tabelle übergeben. Sie können jedoch in der Spaltendefinition Constraints hinzufügen.

Tabellen mithilfe von Unterabfragen erstellen

```
CREATE TABLE dept80
AS
SELECT  employee_id, last_name,
        salary*12 ANNSAL,
        hire_date
FROM    employees
WHERE   department_id = 80;
```

table DEPT80 created.

```
DESCRIBE dept80
```

Name	Null	Type
-----	-----	-----
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

ORACLE

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

Im Beispiel auf der Folie wird die Tabelle `DEPT80` erstellt, die Details zu allen Mitarbeitern in Abteilung 80 enthält. Die Daten für die Tabelle `DEPT80` stammen aus der Tabelle `EMPLOYEES`.

Mit dem Befehl `DESCRIBE` können Sie das Vorliegen einer Datenbanktabelle sowie die Spaltendefinitionen prüfen.

Geben Sie jedoch bei der Auswahl eines Ausdrucks einen Spaltenalias an. Der Ausdruck `SALARY*12` erhält den Alias `ANNSAL`. Ohne die Aliasangabe wird der folgende Fehler generiert:

```
Error starting at line 1 in command:
CREATE TABLE dept80
AS
  SELECT  employee_id, last_name,
          salary*12,
          hire_date
  FROM    employees
  WHERE   department_id = 80
Error at Command Line:4 Column:18
Error report:
SQL Error: ORA-00998: must name this expression with a column alias
00998. 00000 - "must name this expression with a column alias"
*Cause:
*Action:
```

Lektionsagenda

- Datenbankobjekte
 - Benennungsregeln
- Datentypen
- CREATE TABLE-Anweisungen
- Constraints im Überblick: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Tabellen mithilfe von Unterabfragen erstellen
- ALTER TABLE-Anweisungen
- DROP TABLE-Anweisungen

ORACLE

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

ALTER TABLE-Anweisungen

Mit der Anweisung `ALTER TABLE` können Sie:

- neue Spalten hinzufügen
- vorhandene Spaltendefinitionen ändern
- Standardwerte für neue Spalten definieren
- Spalten löschen
- Spalten umbenennen
- Schreibschutz für eine Tabelle aktivieren

ORACLE

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

In den folgenden Fällen müssen Sie die Struktur einer bereits erstellten Tabelle im Nachhinein ändern:

- Sie haben eine Spalte vergessen.
- Sie möchten Definition oder Name der Spalte ändern.
- Sie möchten Spalten entfernen.
- Sie möchten die Tabelle in den schreibgeschützten Modus setzen.

Diese Aufgaben können Sie mit der Anweisung `ALTER TABLE` ausführen.

ALTER TABLE-Anweisungen

Mit der Anweisung ALTER TABLE Spalten hinzufügen, ändern und löschen:

```
ALTER TABLE table
ADD          (column datatype [DEFAULT expr]
              [, column datatype]...);
```

```
ALTER TABLE table
MODIFY       (column datatype [DEFAULT expr]
              [, column datatype]...);
```

```
ALTER TABLE table
DROP (column [, column] ...);
```

ORACLE®

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

Mit der Anweisung ALTER TABLE können Sie Tabellenspalten hinzufügen, ändern und löschen.

Für die Syntax gilt:

<i>table</i>	Steht für den Namen der Tabelle
ADD MODIFY DROP	Typ der Änderung
<i>column</i>	Name der Spalte
<i>datatype</i>	Datentyp und Länge der Spalte
DEFAULT <i>expr</i>	Standardwert für eine Spalte

Spalten hinzufügen

- Mit der Klausel `ADD` Spalten hinzufügen:

```
ALTER TABLE dept80  
ADD      (job_id VARCHAR2(9));
```

```
table DEPT80 altered.
```

- Die neue Spalte wird als letzte Spalte hinzugefügt:

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
1	149	Zlotkey	10500	29-JAN-08	(null)
2	174	Abe1	11000	11-MAY-04	(null)
3	176	Taylor	8600	24-MAR-06	(null)

ORACLE

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

Richtlinien für das Hinzufügen von Spalten

- Sie können Spalten hinzufügen oder ändern.
- Sie können nicht angeben, wo die neue Spalte eingefügt werden soll. Sie wird automatisch als letzte Spalte eingefügt.

Im Beispiel auf der Folie wird der Tabelle `DEPT80` die Spalte `JOB_ID` hinzugefügt. Die Spalte `JOB_ID` ist dann die letzte Spalte der Tabelle.

Hinweis: Wenn die Tabelle beim Hinzufügen einer Spalte bereits Zeilen enthält, nehmen die Zeilen der neuen Spalte zunächst den Wert `NULL` oder den Standardwert an. Tabellen, die in den anderen Spalten bereits Daten enthalten, kann nur dann eine `NOT NULL`-Pflichtspalte hinzugefügt werden, wenn Sie einen Standardwert festlegen. Wenn Sie einer leeren Tabelle eine `NOT NULL`-Spalte hinzufügen, benötigen Sie keinen Standardwert.

Spalten ändern

- Sie können Datentyp, Größe und Standardwert einer Spalte ändern.

```
ALTER TABLE dept80  
MODIFY      (last_name VARCHAR2(30));
```

```
table DEPT80 altered.
```

- Die Änderung des Standardwertes wirkt sich nur auf nachfolgende Einfügungen in die Tabelle aus.

ORACLE

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

Spaltendefinitionen können mit der Anweisung `ALTER TABLE` und der Klausel `MODIFY` geändert werden. Sie können Datentyp, Größe und Standardwert einer Spalte ändern.

Richtlinien

- Sie können die Breite oder Gesamtstellenzahl einer numerischen Spalte erhöhen.
- Sie können die Breite von Zeichenspalten erhöhen.
- Sie können die Breite einer Spalte verringern, wenn:
 - die Spalte nur Nullwerte enthält
 - die Tabelle keine Zeilen enthält
 - durch die Verkleinerung der Spaltenbreite die vorhandenen Werte in dieser Spalte nicht unterschritten werden
- Sie können den Datentyp ändern, wenn die Spalte nur Nullwerte enthält. Hiervon ausgenommen sind Konvertierungen von `CHAR` in `VARCHAR2`, die auch durchgeführt werden können, wenn die Spalten Daten enthalten.
- Die Konvertierung von `CHAR`-Spalten in `VARCHAR2`-Spalten oder umgekehrt ist nur möglich, wenn die Spalten Nullwerte enthalten oder die Größe nicht geändert wird.
- Eine Änderung des Standardwertes einer Spalte wirkt sich nur auf nachfolgende Einfügungen in die Tabelle aus.

Spalten löschen

Mit der Klausel `DROP COLUMN` nicht mehr benötigte Spalten aus der Tabelle löschen:

```
ALTER TABLE dept80  
DROP (job_id);
```

```
table DEPT80 altered.
```

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Zlotkey	10500	29-JAN-08
2	174	Abe1	11000	11-MAY-04
3	176	Taylor	8600	24-MAR-06

ORACLE

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

Mit der Anweisung `ALTER TABLE` und der Klausel `DROP COLUMN` können Sie eine Spalte aus einer Tabelle löschen.

Richtlinien

- Die Spalte kann Daten enthalten oder leer sein.
- Mit der Anweisung `ALTER TABLE DROP COLUMN` kann nur jeweils eine Spalte gelöscht werden.
- Nach der Änderung muss die Tabelle noch mindestens eine Spalte enthalten.
- Nach dem Löschen kann die Spalte nicht wiederhergestellt werden.
- Ein Primärschlüssel, der von einer anderen Spalte referenziert wird, kann nur gelöscht werden, wenn die Option `CASCADE` hinzugefügt wird.
- Wenn die Spalte viele Werte enthält, kann der Löschvorgang einige Zeit dauern. In diesem Fall empfiehlt es sich, die Spalte auf `UNUSED` einzustellen und sie später zu löschen, wenn weniger Benutzer im System angemeldet sind. Auf diese Weise können Sie längere Sperren vermeiden.

Hinweis: Bestimmte Spalten können nie gelöscht werden. Hierzu gehören Spalten, die Teil des Partitionierungsschlüssels einer partitionierten Tabelle sind, oder Spalten, die Teil des `PRIMARY KEY` einer indexorganisierten Tabelle sind. Weitere Informationen über indexorganisierte und partitionierte Tabellen finden Sie im *Oracle Database Concepts* and *Oracle Database Administrator's Guide*.

Option SET UNUSED

- Mit der Option `SET UNUSED` markieren Sie eine oder mehrere Spalten als nicht verwendet.
- Mit der Option `DROP UNUSED COLUMNS` entfernen Sie Spalten, die als nicht verwendet markiert sind.
- Mit dem Schlüsselwort `ONLINE` geben Sie an, dass DML-Vorgänge für die Tabelle zulässig sind, während die Spalten als `UNUSED` markiert sind.

```
ALTER TABLE <table_name>  
SET UNUSED(<column_name> [ , <column_name>] );  
OR  
ALTER TABLE <table_name>  
SET UNUSED COLUMN <column_name> [ , <column_name>];  
  
ALTER TABLE <table_name>  
DROP UNUSED COLUMNS;
```

ORACLE

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

Die Option `SET UNUSED` markiert eine oder mehrere Spalten als nicht verwendete Spalten. Diese Spalten können zu einem späteren Zeitpunkt gelöscht werden, wenn der Bedarf an Systemressourcen niedriger ist. Die Zielspalten werden nicht wirklich aus den Tabellenzeilen entfernt, das heißt, der von den Spalten belegte Plattenspeicherplatz wird nicht freigegeben. Dadurch ist die Antwortzeit schneller als bei Ausführung der Klausel `DROP`. Nicht verwendete Spalten werden wie gelöschte Spalten behandelt, wenngleich die Spaltendaten in den Tabellenzeilen erhalten bleiben. Auf Spalten, die als nicht verwendet markiert wurden, ist kein Zugriff möglich. `SELECT *`-Abfragen rufen keine Daten aus nicht verwendeten Spalten ab. Name und Typ der als nicht verwendet markierten Spalten werden in `DESCRIBE`-Anweisungen nicht angezeigt, und Sie können der Tabelle neue Spalten mit denselben Namen wie die nicht verwendeten Spalten hinzufügen. Die `SET UNUSED`-Informationen werden in der Dictionary View `USER_UNUSED_COL_TABS` gespeichert.

Mit dem Schlüsselwort `ONLINE` geben Sie an, dass DML-Vorgänge für die Tabelle zulässig sind, während die Spalten als `UNUSED` markiert sind. Das Codebeispiel auf der Folie zeigt die Verwendung von `SET UNUSED COLUMN`. Mit dem Schlüsselwort `ONLINE` wird eine Spalte dauerhaft auf "nicht verwendet" festgelegt.

```
ALTER TABLE dept80 SET UNUSED(hire_date) ONLINE;
```

Hinweis: Für die Markierung von Spalten als `UNUSED` gelten ähnliche Richtlinien wie für das Löschen von Spalten.

Option DROP UNUSED COLUMNS

Mit `DROP UNUSED COLUMNS` werden alle Spalten aus der Tabelle entfernt, die aktuell als nicht verwendet markiert sind. Mit dieser Anweisung können Sie den Speicherplatz freigeben, der von den nicht verwendeten Spalten in der Tabelle belegt wird. Enthält die Tabelle keine nicht verwendeten Spalten, wird die Anweisung ohne Fehler zurückgegeben.

```
ALTER TABLE dept80  
SET UNUSED (last_name);
```

```
ALTER TABLE dept80  
DROP UNUSED COLUMNS;
```

Hinweis: Mit einer anschließenden `DROP UNUSED COLUMNS`-Anweisung werden ähnlich wie bei der Anweisung `DROP COLUMN` alle nicht verwendeten Spalten physisch aus der Tabelle entfernt.

Schreibgeschützte Tabellen

Mit der Syntax `ALTER TABLE` können Sie:

- Tabellen in den schreibgeschützten Modus setzen und so DDL- oder DML-Änderungen während der Tabellenwartung verhindern
- Tabellen in den Schreibzugriffsmodus zurücksetzen

```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```

ORACLE

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

Mit `READ ONLY` können Sie Tabellen in den schreibgeschützten Modus setzen. In diesem Modus können keine DML-Anweisungen oder `SELECT . . . FOR UPDATE`-Anweisungen für die Tabelle ausgeführt werden. DDL-Anweisungen können abgesetzt werden, wenn sie keine Daten in der Tabelle ändern. Vorgänge, die sich auf die zur Tabelle gehörenden Indizes beziehen, sind im Modus `READ ONLY` zulässig.

Um eine schreibgeschützte Tabelle in den Schreibzugriffsmodus zurückzusetzen, geben Sie `READ/WRITE` an.

Hinweis: Tabellen im Modus `READ ONLY` können gelöscht werden. Der Befehl `DROP` wird nur im Data Dictionary ausgeführt, sodass kein Zugriff auf den Tabelleninhalt erforderlich ist. Der von der Tabelle belegte Speicherplatz wird erst freigegeben, wenn die Tabelle in den Schreibzugriffsmodus zurückgesetzt wird. Erst dann können die erforderlichen Änderungen an den Blocksegmentheadern und anderen Komponenten vorgenommen werden.

Weitere Informationen zur Anweisung `ALTER TABLE` erhalten Sie im Kurs *Oracle Database 12c: SQL Workshop II*.

Lektionsagenda

- Datenbankobjekte
 - Benennungsregeln
- Datentypen
- CREATE TABLE-Anweisungen
- Constraints im Überblick: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Tabellen mithilfe von Unterabfragen erstellen
- ALTER TABLE-Anweisungen
- DROP TABLE-Anweisungen

ORACLE®

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

Tabellen löschen

- Tabellen in den Papierkorb verschieben
- Mit der Klausel `PURGE` Tabellen und alle enthaltenen Daten vollständig entfernen
- Abhängige Objekte invalidieren und Objektberechtigungen für die Tabelle entfernen

```
DROP TABLE dept80;  
table DEPT80 dropped.
```

ORACLE

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

Mit der Anweisung `DROP TABLE` werden Tabellen in den Papierkorb verschoben oder mit allen enthaltenen Daten vollständig aus der Datenbank gelöscht. Ohne die Klausel `PURGE` gibt `DROP TABLE` keinen Speicherplatz an den Tablespace frei, der von anderen Objekten verwendet werden kann. Der Speicherplatz wird weiterhin der Speicherplatz-Quota des Benutzers zugerechnet. Beim Löschen von Tabellen werden die abhängigen Objekte invalidiert und die Objektberechtigungen für die Tabelle entfernt.

Mit `PURGE` werden alle Daten in der Tabelle und die mit ihr verknüpften Indizes aus der Datenbank gelöscht.

Syntax

```
DROP TABLE table [PURGE]
```

In der Syntax steht *table* für den Namen der Tabelle.

.Richtlinien

- Alle Daten werden aus der Tabelle gelöscht.
- Views und Synonyme bleiben erhalten, sind jedoch ungültig.
- Alle noch nicht gespeicherten Transaktionen werden festgeschrieben.
- Nur der Ersteller oder ein Benutzer mit der Berechtigung `DROP ANY TABLE` kann Tabellen entfernen.

Hinweis: Mit der Anweisung `FLASHBACK TABLE` stellen Sie eine gelöschte Tabelle aus dem Papierkorb wieder her. Dieser Vorgang wird im Kurs *Oracle Database 12c: SQL Workshop II* ausführlich behandelt.

Quiz

Für welche Zwecke werden Constraints verwendet? Wählen Sie drei Antworten.

- a. Beim Einfügen, Aktualisieren oder Löschen von Zeilen Regeln für Tabellendaten durchsetzen
- b. Löschen von Tabellen verhindern
- c. Erstellung von Tabellen verhindern
- d. Erstellung von Daten in Tabellen verhindern

ORACLE

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

Richtige Antworten: a, b, d

Zusammenfassung

In dieser Lektion haben Sie gelernt, wie Sie mit den Anweisungen `CREATE TABLE`, `ALTER TABLE` und `DROP TABLE` Tabellen erstellen, Tabellen und Spalten bearbeiten und Constraints hinzufügen.

- Wichtige Datenbankobjekte kategorisieren
- Tabellenstrukturen prüfen
- Für Spalten verfügbare Datentypen auflisten
- Einfache Tabellen erstellen
- Erstellung von Constraints im Zuge der Tabellenerstellung beschreiben
- Funktionsweise von Schemaobjekten beschreiben

ORACLE

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

In dieser Lektion haben Sie Folgendes gelernt:

`CREATE TABLE`

- Mit der Anweisung `CREATE TABLE` Tabellen erstellen und Constraints hinzufügen
- Mithilfe von Unterabfragen Tabellen basierend auf anderen Tabellen erstellen

`DROP TABLE`

- Zeilen und Tabellenstruktur entfernen
- Diese Anweisung kann nach ihrer Ausführung nicht zurückgerollt werden.

Übungen zu Lektion 11 – Überblick

Diese Übung behandelt folgende Themen:

- Neue Tabellen erstellen
- Neue Tabellen mithilfe der Syntax `CREATE TABLE AS` erstellen
- Vorhandensein von Tabellen prüfen
- Tabellen ändern
- Spalten hinzufügen
- Spalten löschen
- Tabellen in den schreibgeschützten Modus setzen
- Tabellen löschen

ORACLE

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

Sie erstellen mit der Anweisung `CREATE TABLE` neue Tabellen und prüfen, ob sie der Datenbank hinzugefügt wurden. Darüber hinaus lernen Sie, Tabellen in den schreibgeschützten Modus (`READ ONLY`) zu setzen und anschließend in den Schreibzugriffsmodus (`READ/WRITE`) zurückzuführen.

Hinweis: Um die Abfragen in SQL Developer auszuführen, klicken Sie für alle DDL- und DML-Anweisungen auf das Symbol **Run Script** (oder drücken F5). Auf diese Weise werden die Rückmeldungen in der Registerkarte **Script Output** angezeigt. Für `SELECT`-Abfragen klicken Sie weiterhin auf das Symbol **Execute Statement** oder drücken F9, um die formatierte Ausgabe in der Registerkarte **Results** anzuzeigen.

Tabellenbeschreibungen



ORACLE®

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

Schemabeschreibung

Allgemeine Beschreibung

Das in den Beispielschemas von Oracle Database dargestellte Beispielunternehmen arbeitet weltweit und liefert viele unterschiedliche Produkte aus. Das Unternehmen hat drei Abteilungen:

- **Human Resources:** Überwacht Informationen zu Mitarbeitern und Einrichtungen
- **Order Entry:** Überwacht den Bestand und den Verkauf der Produkte über verschiedene Kanäle
- **Sales History:** Überwacht Unternehmensstatistiken, um Geschäftsentscheidungen zu vereinfachen

Jede dieser Abteilungen wird durch ein Schema dargestellt. In diesem Kurs hat der Benutzer Zugriff auf die Objekte in sämtlichen Schemas. Der Schwerpunkt in den Beispielen, Demonstrationen und Übungen liegt jedoch auf dem Schema `Human Resources (HR)`.

Alle für die Erstellung der Beispielschemas erforderlichen Skripte befinden sich im Ordner `$ORACLE_HOME/demo/schema/`.

Human Resources (HR)

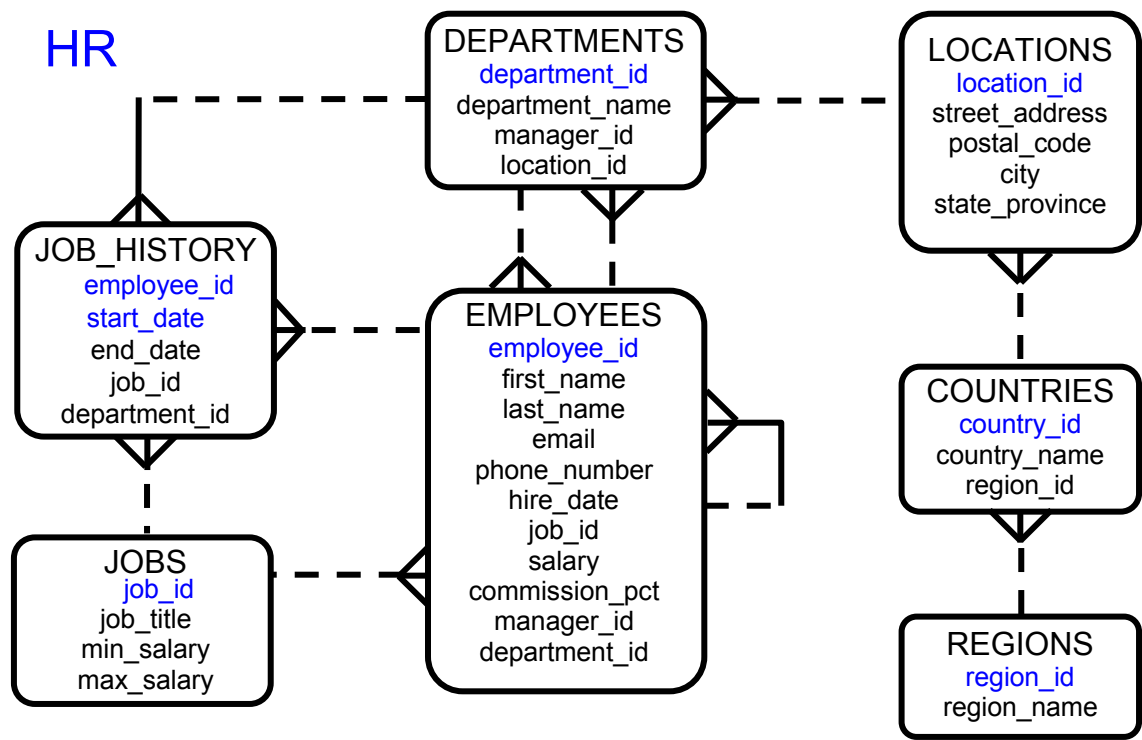
Dieses Schema wird im aktuellen Kurs verwendet. In den Datensätzen des Schemas `Human Resources (HR)` ist jeder Mitarbeiter mit einer ID-Nummer, einer E-Mail-Adresse, einem Tätigkeits-ID-Code, einem Gehalt und einem Manager verzeichnet. Neben ihrem Gehalt bekommen einige Mitarbeiter Provisionen.

Das Unternehmen zeichnet auch Informationen über die Tätigkeiten innerhalb des Unternehmens auf. Jede Tätigkeit ist durch eine ID und eine Bezeichnung gekennzeichnet. Außerdem werden Angaben zum Mindest- und Höchstgehalt für die Tätigkeit aufgezeichnet. Einige Mitarbeiter sind schon sehr lange in der Firma und hatten verschiedene Positionen inne. Wenn ein Mitarbeiter ausscheidet, werden seine Beschäftigungsdauer, die Tätigkeits-ID und die Abteilung aufgezeichnet.

Die Beispielfirma ist an verschiedenen Standorten tätig. Daher werden die Lagerstandorte und die Abteilungsstandorte überwacht. Jeder Mitarbeiter ist einer Abteilung zugewiesen. Jede Abteilung wird entweder anhand einer eindeutigen Abteilungsnummer oder anhand einer Kurzbezeichnung identifiziert. Jede Abteilung ist einem Standort zugewiesen. Jeder Standort besitzt eine vollständige Adresse mit Straßennamen, Postleitzahl, Ort, Bundesland, Bundesstaat, Provinz oder Kanton sowie Länderkennung.

An den Abteilungs- und Lagerstandorten zeichnet das Unternehmen Details wie den Namen des Landes, das Währungssymbol, den Namen der Währung sowie die Region auf, in der das Land geografisch angesiedelt ist.

HR – Entity Relationship-Diagramm



Human Resources (HR) – Tabellenbeschreibungen

DESCRIBE countries

Name	Null	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

SELECT * FROM countries

COUNTRY_ID	COUNTRY_NAME	REGION_ID
1 CA	Canada	2
2 DE	Germany	1
3 UK	United Kingdom	1
4 US	United States of America	2

DESCRIBE departments

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

SELECT * FROM departments

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

DESCRIBE employees

Name	Null	Type
-----	-----	-----
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

SELECT * FROM employees

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	(null)	(null)	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	(null)	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	(null)	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	(null)	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	(null)	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	(null)	103	60
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-07	ST_MAN	5800	(null)	100	50
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-03	ST_CLERK	3500	(null)	124	50
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-05	ST_CLERK	3100	(null)	124	50
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-06	ST_CLERK	2600	(null)	124	50
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-06	ST_CLERK	2500	(null)	124	50
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-08	SA_MAN	10500	0.2	100	80
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-04	SA_REP	11000	0.3	149	80
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-06	SA_REP	8600	0.2	149	80
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-07	SA_REP	7000	0.15	149	(null)
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-03	AD_ASST	4400	(null)	101	10
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000	(null)	100	20
202	Pat	Fay	PFAY	603.123.6666	17-AUG-05	MK_REP	6000	(null)	201	20
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-02	AC_MGR	12008	(null)	101	110
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-02	AC_ACCOUNT	8300	(null)	205	110

```
DESCRIBE job_history
```

Name	Null	Type
-----	-----	-----
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)





```
SELECT * FROM job_history
```

	A2	EMPLOYEE_ID	A2	START_DATE	A2	END_DATE	A2	JOB_ID	A2	DEPARTMENT_ID
1		102		13-JAN-01		24-JUL-06		IT_PROG		60
2		101		21-SEP-97		27-OCT-01		AC_ACCOUNT		110
3		101		28-OCT-01		15-MAR-05		AC_MGR		110
4		201		17-FEB-04		19-DEC-07		MK_REP		20
5		114		24-MAR-06		31-DEC-07		ST_CLERK		50
6		122		01-JAN-07		31-DEC-07		ST_CLERK		50
7		200		17-SEP-95		17-JUN-01		AD_ASST		90
8		176		24-MAR-06		31-DEC-06		SA_REP		80
9		176		01-JAN-07		31-DEC-07		SA_MAN		80
10		200		01-JUL-02		31-DEC-06		AC_ACCOUNT		90

```
DESCRIBE jobs
```

Name	Null	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

```
SELECT * FROM jobs
```

	 JOB_ID	 JOB_TITLE	 MIN_SALARY	 MAX_SALARY
1	AD_PRES	President	20080	40000
2	AD_VP	Administration Vice President	15000	30000
3	AD_ASST	Administration Assistant	3000	6000
4	AC_MGR	Accounting Manager	8200	16000
5	AC_ACCOUNT	Public Accountant	4200	9000
6	SA_MAN	Sales Manager	10000	20080
7	SA_REP	Sales Representative	6000	12008
8	ST_MAN	Stock Manager	5500	8500
9	ST_CLERK	Stock Clerk	2008	5000
10	IT_PROG	Programmer	4000	10000
11	MK_MAN	Marketing Manager	9000	15000
12	MK_REP	Marketing Representative	4000	9000

```
DESCRIBE locations
```

Name	Null	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)



```
SELECT * FROM locations
```

	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1	1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
2	1500	2011 Interiors Blvd	99236	South San Francisco	California	US
3	1700	2004 Charade Rd	98199	Seattle	Washington	US
4	1800	460 Bloor St. W.	ON M5S 1X8	Toronto	Ontario	CA
5	2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

```
DESCRIBE regions
```

Name	Null	Type
-----	-----	-----
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

```
SELECT * FROM regions
```

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



ORACLE®

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

Ziele

Nach Ablauf dieses Anhangs haben Sie folgende Ziele erreicht:

- Schlüsselfeatures von Oracle SQL Developer auflisten
- Menüoptionen von Oracle SQL Developer angeben
- Datenbankverbindungen erstellen
- Datenbankobjekte verwalten
- SQL Worksheet verwenden
- SQL-Skripte speichern und ausführen
- Berichte erstellen und speichern
- Data Modeler-Optionen in SQL Developer durchsuchen

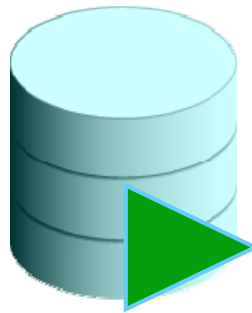
ORACLE

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

In diesem Anhang wird das grafische Tool SQL Developer vorgestellt. Sie lernen, wie Sie SQL Developer für Aufgaben im Bereich der Datenbankentwicklung verwenden. Außerdem wird beschrieben, wie Sie SQL-Anweisungen und SQL-Skripte mit dem SQL Worksheet ausführen.

Was ist Oracle SQL Developer?

- Oracle SQL Developer ist ein grafisches Tool, das die Produktivität erhöht und Aufgaben der Datenbankentwicklung vereinfacht.
- Sie können sich mit der standardmäßigen Oracle-Datenbankauthentifizierung bei jedem Oracle-Zieldatenbankschema anmelden.



SQL Developer

ORACLE

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

Oracle SQL Developer ist ein kostenloses grafisches Tool, das Ihre Produktivität steigert und Routineaufgaben der Datenbankentwicklung vereinfacht. Mit wenigen Mausklicks können Sie problemlos Stored Procedures erstellen und debuggen, SQL-Anweisungen testen und Optimizer-Pläne anzeigen.

Mit SQL Developer, dem visuellen Tool zur Datenbankentwicklung, werden folgende Aufgaben vereinfacht:

- Datenbankobjekte durchsuchen und verwalten
- SQL-Anweisungen und -Skripte ausführen
- PL/SQL-Anweisungen bearbeiten und debuggen
- Berichte erstellen

Mit der standardmäßigen Oracle-Datenbankauthentifizierung können Sie sich bei jedem Oracle-Zieldatenbankschema anmelden. Nach der Anmeldung können Sie Vorgänge für die Objekte in der Datenbank ausführen.

SQL Developer ist die Verwaltungsschnittstelle für den Oracle Application Express Listener. Über die neue Schnittstelle können Sie globale Einstellungen und Einstellungen für mehrere Datenbanken mit unterschiedlichen Datenbankverbindungen für den Application Express Listener festlegen. Objekte lassen sich in SQL Developer nach Tabellen- oder Spaltenname mit Drag & Drop in das Worksheet ziehen. Neben verbesserten DB Diff-Vergleichsoptionen werden auch GRANT-Anweisungen im SQL-Editor und DB Doc-Berichte unterstützt. Zusätzlich unterstützt SQL Developer Funktionen aus Oracle Database 12c.

SQL Developer – Spezifikationen

- Im Lieferumfang von Oracle Database 12c Release 1 enthalten
- In Java entwickelt
- Unterstützt die Plattformen Windows, Linux und Mac OS X
- Standardkonnektivität durch den JDBC-Thin-Treiber
- Anmeldung bei Oracle Database Version 9.2.0.1 oder höher möglich

ORACLE

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

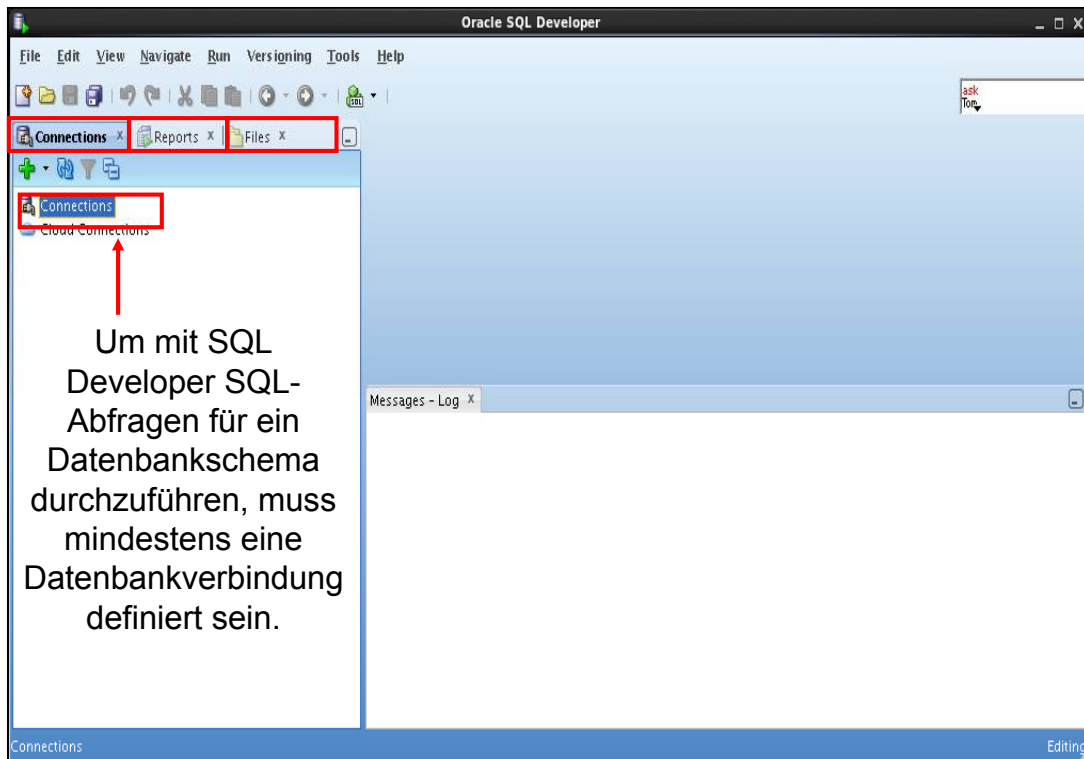
Oracle SQL Developer ist standardmäßig im Lieferumfang von Oracle Database 12c Release 1 enthalten. SQL Developer wurde mithilfe der Oracle JDeveloper-IDE (Integrated Development Environment, integrierte Entwicklungsumgebung) in Java entwickelt und ist damit ein plattformübergreifendes Tool, das unter Windows, Linux und Mac OS X ausgeführt werden kann.

Die Standardkonnektivität zur Datenbank erfolgt über den Java Database Connectivity-(JDBC-)Thin-Treiber, sodass kein Oracle Home erforderlich ist. Für SQL Developer benötigen Sie kein Installationsprogramm. Sie dekomprimieren einfach die heruntergeladene Datei. Mit SQL Developer können sich Benutzer bei Oracle Database 9.2.0.1 oder höher sowie bei allen Oracle-Datenbankeditionen einschließlich Express Edition anmelden.

Hinweis: Für Oracle Database 12c Release 1 müssen Sie SQL Developer herunterladen und installieren. SQL Developer kann kostenlos über folgenden Link heruntergeladen werden:
<http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>

Anweisungen zur Installation von SQL Developer finden Sie auf der Website unter:
<http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html>

SQL Developer 3.2 – Benutzeroberfläche



Um mit SQL Developer SQL-Abfragen für ein Datenbankschema durchzuführen, muss mindestens eine Datenbankverbindung definiert sein.

ORACLE

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

Die Benutzeroberfläche von SQL Developer umfasst (von links nach rechts) drei Hauptregisterkarten zur Navigation:

- **Connections:** In dieser Registerkarte können Sie Datenbankobjekte und Benutzer durchsuchen, auf die Sie Zugriff haben.
- **Berichte:** In dieser durch das Symbol **Reports** dargestellten Registerkarte führen Sie vordefinierte Berichte aus oder erstellen und fügen eigene Berichte hinzu.
- **Files:** In dieser durch das Ordnersymbol **Files** dargestellten Registerkarte können Sie von Ihrem lokalen Rechner aus auf Dateien zugreifen, ohne über das Menü **File > Open** navigieren zu müssen.

Allgemeine Navigation und Verwendung

Bei SQL Developer dient die linke Seite zur Navigation sowie zum Suchen und Wählen von Objekten. Auf der rechten Seite werden Informationen über die gewählten Objekte angezeigt. Über Voreinstellungen können Sie viele Aspekte der Darstellung und des Verhaltens von SQL Developer anpassen.

Hinweis: Sie müssen mindestens eine Verbindung definieren, damit Sie sich bei einem Datenbankschema anmelden und SQL-Abfragen bzw. Prozeduren oder Funktionen ausführen können.

Menüs

Folgende Menüs enthalten Standardeinträge sowie zusätzliche Einträge für spezifische Features von SQL Developer:

- **View:** Enthält Optionen, die bestimmen, was in der Benutzeroberfläche von SQL Developer angezeigt wird
- **Navigate:** Enthält Optionen für die Navigation zu verschiedenen Bereichen und die Ausführung von Unterprogrammen
- **Run:** Enthält die bei Auswahl einer Funktion oder Prozedur relevanten Optionen **Run File** und **Execution Profile** sowie Debugging-Optionen
- **Versioning:** Bietet integrierte Unterstützung für folgende Versionierungs- und Quellkontrollsysteme: Concurrent Versions System (CVS) und Subversion
- **Tools:** Ruft SQL Developer-Tools wie SQL*Plus, Preferences oder SQL Worksheet auf und enthält Optionen für die Migration von Fremddatenbanken zu Oracle

Hinweis: Das Menü **Run** enthält auch Optionen, die relevant sind, wenn Funktionen oder Prozeduren für Debugging-Zwecke gewählt wurden.

Datenbankverbindungen erstellen

- Um SQL Developer verwenden zu können, ist mindestens eine Datenbankverbindung erforderlich.
- Sie können Verbindungen erstellen und testen für:
 - mehrere Datenbanken
 - mehrere Schemas
- SQL Developer importiert automatisch alle Verbindungen, die in der Datei `tnsnames.ora` auf dem System definiert sind.
- Verbindungen können in eine Extensible Markup Language-(XML-)Datei exportiert werden.
- Jede zusätzlich erstellte Datenbankverbindung wird in der Connections Navigator-Hierarchie aufgelistet.

ORACLE

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

Eine Verbindung ist ein SQL Developer-Objekt, das die nötigen Informationen angibt, mit denen Sie sich als bestimmter Benutzer bei einer bestimmten Datenbank anmelden können. Um SQL Developer zu verwenden, muss mindestens eine Datenbankverbindung vorhanden sein bzw. erstellt oder importiert werden.

Sie können Verbindungen für mehrere Datenbanken und mehrere Schemas erstellen und testen.

Standardmäßig befindet sich die Datei `tnsnames.ora` im Verzeichnis

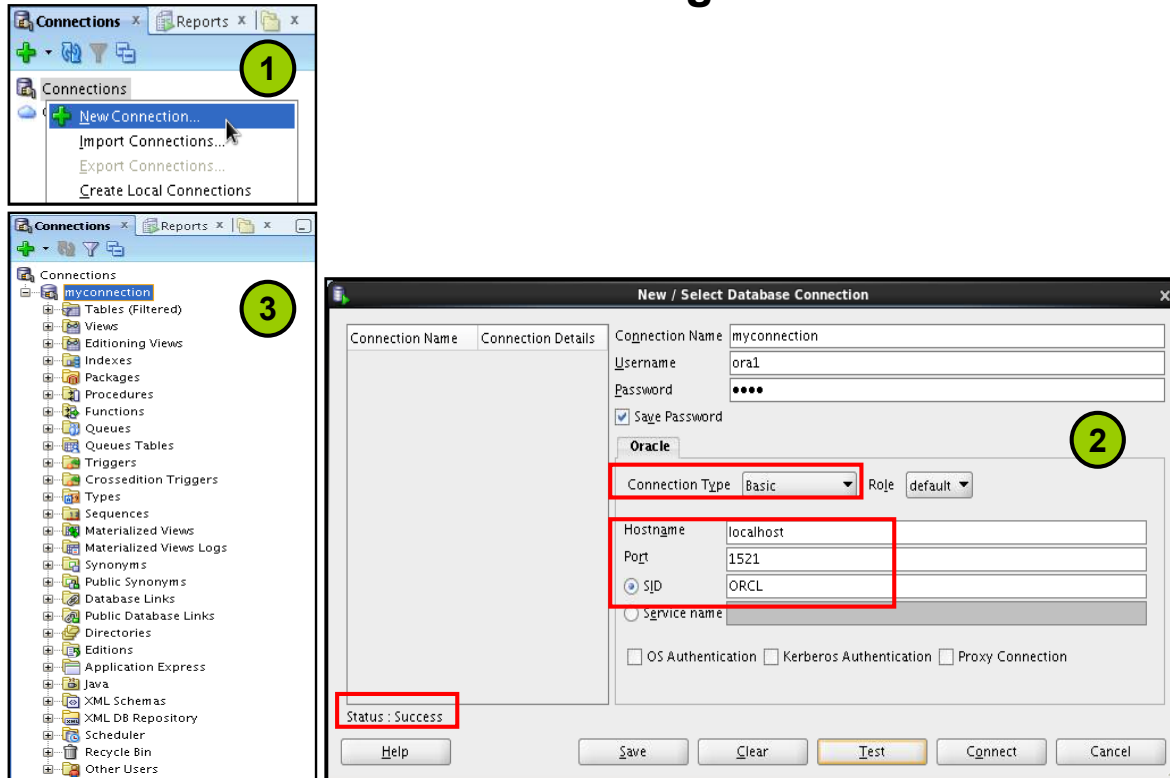
`$ORACLE_HOME/network/admin`. Die Datei kann jedoch auch in einem durch die Umgebungsvariable `TNS_ADMIN` oder durch einen Registrierungswert angegebenen Verzeichnis gespeichert werden. Wenn Sie SQL Developer starten und das Dialogfeld **Database Connections** anzeigen, importiert SQL Developer automatisch alle Verbindungen, die in der Datei `tnsnames.ora` auf dem System definiert sind.

Hinweis: Wenn die in der Datei `tnsnames.ora` definierten Verbindungen unter Windows nicht von SQL Developer verwendet werden, definieren Sie `TNS_ADMIN` als Systemumgebungsvariable.

Sie können Verbindungen zur späteren Wiederverwendung in eine XML-Datei exportieren.

Außerdem haben Sie die Möglichkeit, sich mehrfach als jeweils unterschiedlicher Benutzer bei derselben Datenbank anzumelden oder sich bei verschiedenen Datenbanken anzumelden.

Datenbankverbindungen erstellen



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

Um eine Datenbankverbindung zu erstellen, gehen Sie wie folgt vor:

1. Klicken Sie in der Registerkarte **Connections** mit der rechten Maustaste auf **Connections**, und wählen Sie **New Connection**.
2. Geben Sie im Fenster **New/Select Database Connection** den Verbindungsnamen ein. Geben Sie den Benutzernamen und das Kennwort des Schemas ein, bei dem Sie sich anmelden möchten.
 - a. Wählen Sie in der Dropdown-Liste **Role** entweder *default* oder *SYSDBA*. (*SYSDBA* wird für den Benutzer *sys* oder einen Benutzer mit DBA-Berechtigungen gewählt.)
 - b. Für den Verbindungstyp stehen folgende Optionen zur Wahl:

Basic: Geben Sie für diesen Typ den Hostnamen und die SID für die Datenbank ein, bei der Sie sich anmelden möchten. **Port** ist bereits auf 1521 festgelegt. Sie können aber auch den Servicennamen direkt eingeben, wenn Sie eine Remote-Datenbankverbindung verwenden.

TNS: Sie können jeden Datenbankalias wählen, der aus der Datei `tnsnames.ora` importiert wurde.

LDAP: Sie können Datenbankservices in Oracle Internet Directory, einer Komponente von Oracle Identity Management, nachschlagen.

Advanced: Sie können eine benutzerdefinierte Java Database Connectivity-(JDBC-) URL für die Anmeldung bei der Datenbank definieren.

Local/Bequeath: Befinden sich Client und Datenbank auf demselben Rechner, kann die Clientverbindung direkt an einen dedizierten Serverprozess übergeben werden, ohne dass der Listener einbezogen wird.

- c. Klicken Sie auf **Test**, um sicherzustellen, dass die Verbindung richtig festgelegt wurde.
- d. Klicken Sie auf **Connect**.

Bei Aktivierung des Kontrollkästchens **Save Password** wird das Kennwort in einer XML-Datei gespeichert. Wenn Sie dann die SQL Developer-Verbindung beenden und wieder öffnen, werden Sie nicht mehr aufgefordert, ein Kennwort einzugeben.

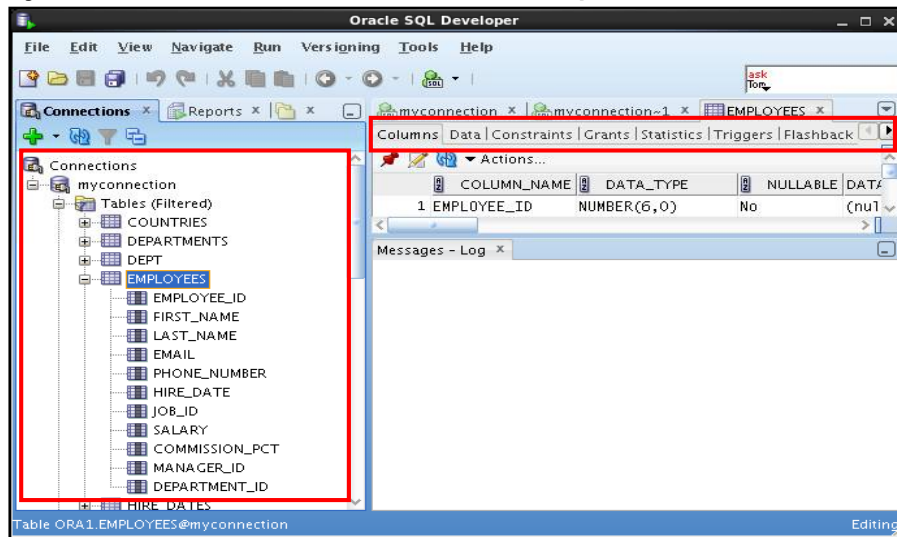
- 3. Die Verbindung wird dem Connections Navigator hinzugefügt. Sie können sie einblenden, um die Datenbankobjekte und Objektdefinitionen anzuzeigen, etwa Abhängigkeiten, Details und Statistiken.

Hinweis: Im Fenster **New/Select Database Connection** können Sie über die Registerkarten **Access**, **MySQL** und **SQL Server** auch Verbindungen zu Fremddatenquellen definieren. Diese Verbindungen sind jedoch Read-Only-Verbindungen, mit denen Sie nur Objekte und Daten in der jeweiligen Datenquelle durchsuchen können.

Datenbankobjekte durchsuchen

Mit dem Connections Navigator können Sie:

- viele Objekte in einem Datenbankschema durchsuchen
- Objektdefinitionen auf einen Blick prüfen



ORACLE

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

Wenn Sie eine Datenbankverbindung erstellt haben, können Sie mit dem Connections Navigator viele Objekte in einem Datenbankschema durchsuchen, darunter Tabellen, Views, Indizes, Packages, Prozeduren, Trigger und Typen.

Bei SQL Developer dient die linke Seite der Navigation, also dem Suchen und Wählen von Objekten. Auf der rechten Seite werden Informationen zu den gewählten Objekten angezeigt. Über Voreinstellungen können Sie zahlreiche Aspekte der Darstellung von SQL Developer anpassen.

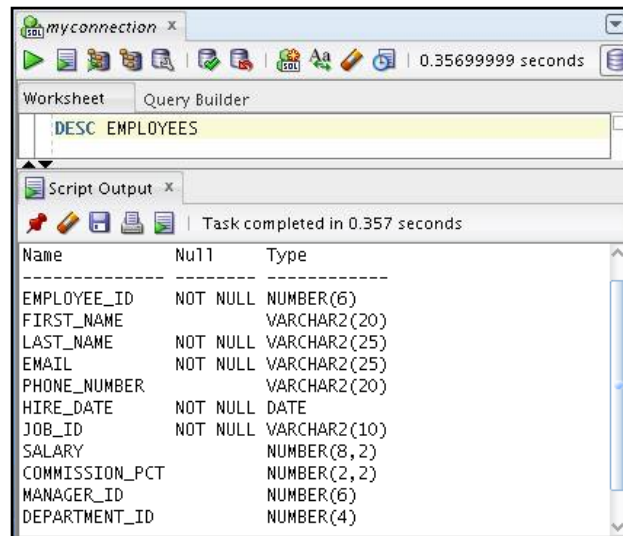
Die Definitionen der Objekte werden in verschiedenen Registerkarten angezeigt, die Informationen aus dem Data Dictionary enthalten. Wenn Sie beispielsweise eine Tabelle im Navigator wählen, werden alle Einzelheiten über Spalten, Constraints, Berechtigungen, Statistiken, Trigger usw. in einem übersichtlichen, in Registerkarten unterteilten Fenster angezeigt.

Um die Definition der Tabelle `EMPLOYEES` anzuzeigen (siehe Folie), gehen Sie wie folgt vor:

1. Blenden Sie im Connections Navigator den Knoten **Connections** ein.
2. Blenden Sie **Tables** ein.
3. Klicken Sie auf `EMPLOYEES`. Standardmäßig ist die Registerkarte **Columns** aktiviert, die die Spaltenbeschreibung der Tabelle enthält. Mit der Registerkarte **Data** können Sie die Tabellendaten anzeigen und außerdem neue Zeilen eingeben, Daten aktualisieren und diese Änderungen in der Datenbank festschreiben.

Tabellenstrukturen anzeigen

Struktur einer Tabelle mit dem Befehl `DESCRIBE` anzeigen:



The screenshot shows the SQL Developer interface. The 'Worksheet' tab is active, displaying the command `DESC EMPLOYEES`. The 'Script Output' tab shows the result of the command, which is a table with three columns: Name, Null, and Type. The table lists the columns of the EMPLOYEES table and their respective data types and nullability.

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

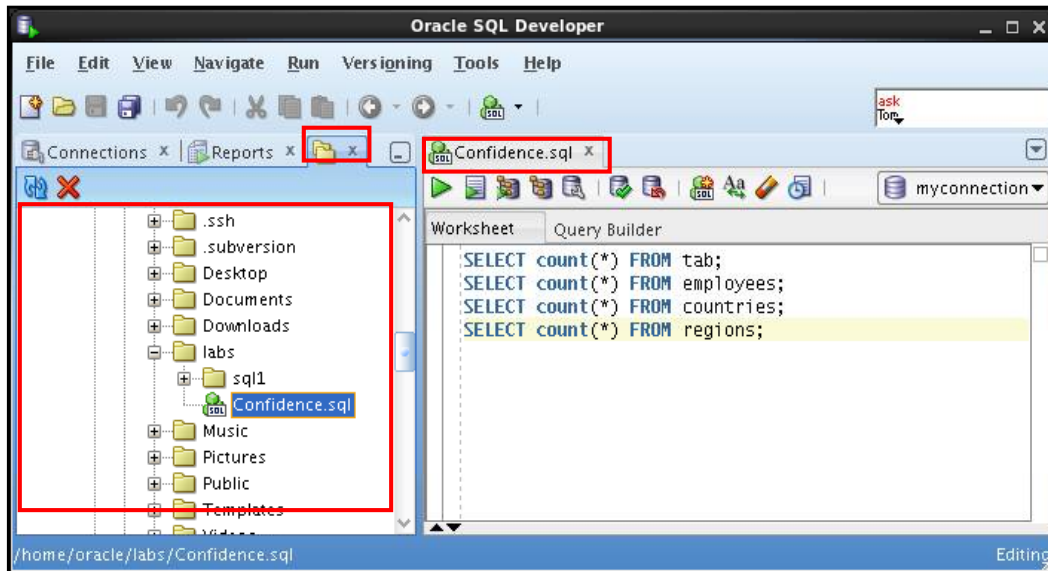
ORACLE

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

In SQL Developer können Sie die Struktur einer Tabelle mit dem Befehl `DESCRIBE` anzeigen. Daraufhin werden die Spaltennamen und Datentypen sowie ein Hinweis darauf angezeigt, ob eine Spalte Daten enthalten muss.

Dateien durchsuchen

Mit dem File Navigator das Dateisystem untersuchen und Systemdateien öffnen



ORACLE

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

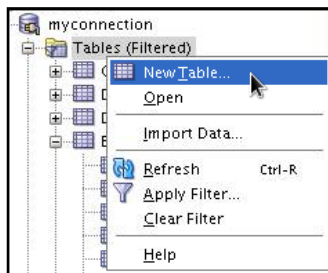
Datenbankobjekte durchsuchen

Mit dem File Navigator können Sie Systemdateien durchsuchen und öffnen.

- Um den File Navigator anzuzeigen, klicken Sie auf die Registerkarte **View** und wählen **Files**. Alternativ können Sie zu **View > Files** navigieren.
- Um den Inhalt einer Datei anzuzeigen, doppelklicken Sie auf einen Dateinamen. Der Inhalt der Datei wird dann im SQL Worksheet-Bereich angezeigt.

Schemaobjekte erstellen

- SQL Developer unterstützt die Erstellung beliebiger Schemaobjekte:
 - durch Ausführung einer SQL-Anweisung im SQL Worksheet
 - mithilfe des Kontextmenüs
- Objekte in einem Bearbeitungsdialogfeld oder über eines der zahlreichen Kontextmenüs bearbeiten
- DDL (Data Definition Language) für Anpassungen wie die Erstellung eines neuen Objekts oder die Bearbeitung eines vorhandenen Schemaobjekts anzeigen



ORACLE

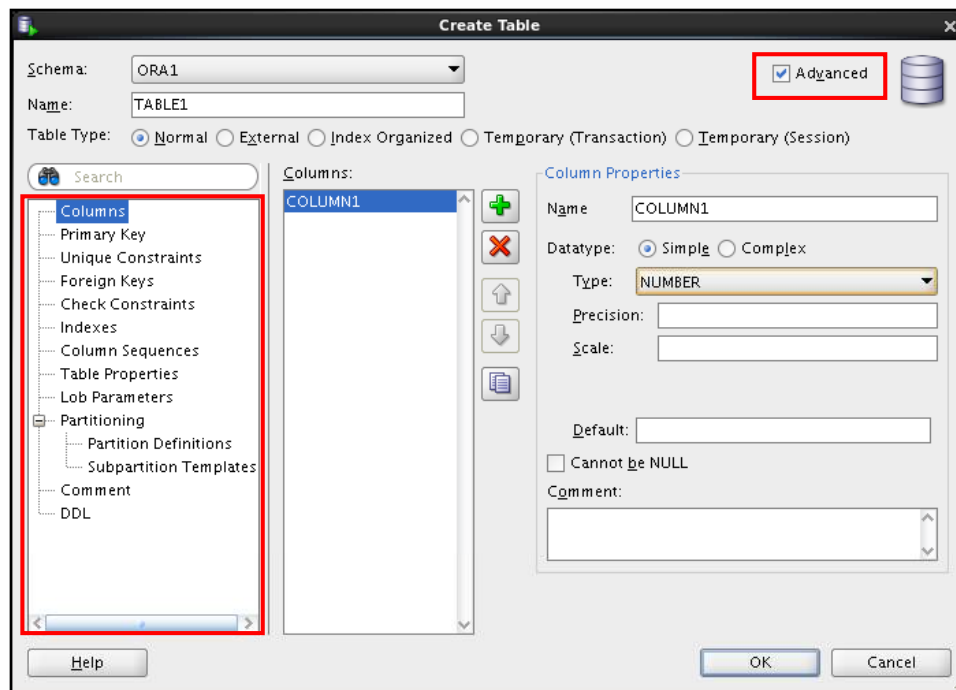
Copyright © 2014, Oracle und/oder verbundene Unternehmen. All rights reserved. Alle Rechte vorbehalten.

SQL Developer unterstützt die Erstellung beliebiger Schemaobjekte durch Ausführung einer SQL-Anweisung im SQL Worksheet. Alternativ können Sie Objekte mithilfe der Kontextmenüs erstellen. Nach ihrer Erstellung können Sie die Objekte in einem Bearbeitungsdialogfeld oder über eines der zahlreichen Kontextmenüs bearbeiten.

Bei der Erstellung neuer Objekte oder der Bearbeitung vorhandener Objekte kann für Prüfzwecke die DDL angezeigt werden. Die Option **Export DDL** ist verfügbar, wenn Sie den vollständigen DDL-Code für ein oder mehrere Objekte im Schema erstellen möchten.

Auf der Folie wird gezeigt, wie Sie eine Tabelle über das Kontextmenü erstellen. Um ein Dialogfeld zur Erstellung einer neuen Tabelle zu öffnen, klicken Sie mit der rechten Maustaste auf **Tables** und wählen **New Table**. Die Dialogfelder zum Erstellen und Bearbeiten von Datenbankobjekten verfügen über mehrere Registerkarten, die jeweils eine logische Zusammenstellung von Eigenschaften für den entsprechenden Objekttyp enthalten.

Neue Tabellen erstellen – Beispiel



ORACLE

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

Wenn Sie im Dialogfeld **Create Table** das Kontrollkästchen **Advanced** deaktivieren, können Sie schnell eine Tabelle erstellen, indem Sie Spalten und einige häufig verwendete Features angeben.

Bei Aktivierung des Kontrollkästchens **Advanced** werden im Dialogfeld **Create Table** mehrere Optionen eingeblendet, mit denen Sie beim Erstellen der Tabelle erweiterte Features festlegen können.

Das Beispiel auf der Folie zeigt, wie die Tabelle `DEPENDENTS` durch Aktivierung des Kontrollkästchens **Advanced** erstellt wird.

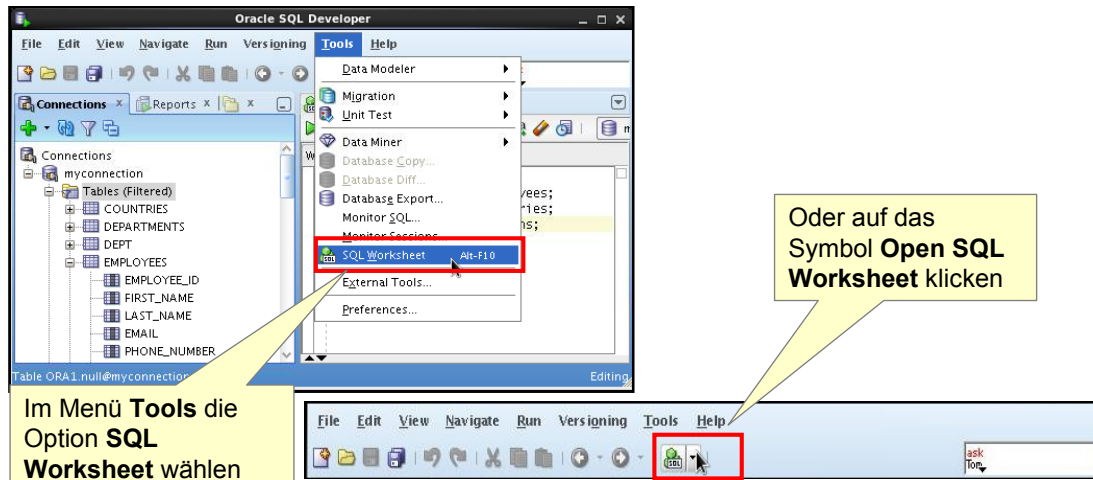
Um eine neue Tabelle zu erstellen, gehen Sie wie folgt vor:

1. Klicken Sie im Connections Navigator mit der rechten Maustaste auf **Tables**, und wählen Sie **Create TABLE**.
2. Aktivieren Sie im Dialogfeld **Create Table** das Kontrollkästchen **Advanced**.
3. Geben Sie Spalteninformationen an.
4. Klicken Sie auf **OK**.

Es empfiehlt sich, darüber hinaus in der Registerkarte **Primary Key** einen Primärschlüssel anzugeben. Um eine erstellte Tabelle zu bearbeiten, klicken Sie im Connections Navigator mit der rechten Maustaste auf die gewünschte Tabelle und wählen **Edit**.

SQL Worksheet

- Mit dem SQL Worksheet SQL-, PL/SQL- und SQL*Plus-Anweisungen eingeben und ausführen
- Aktionen angeben, die von der mit dem Worksheet verknüpften Datenbankverbindung verarbeitet werden können



ORACLE

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

Wenn Sie sich bei einer Datenbank anmelden, wird automatisch ein SQL Worksheet-Fenster für diese Verbindung geöffnet. Im SQL Worksheet können Sie SQL-, PL/SQL- und SQL*Plus-Anweisungen eingeben und ausführen. Das SQL Worksheet unterstützt nicht alle SQL*Plus-Anweisungen. Nicht unterstützte SQL*Plus-Anweisungen werden ignoriert und nicht an die Datenbank übergeben.

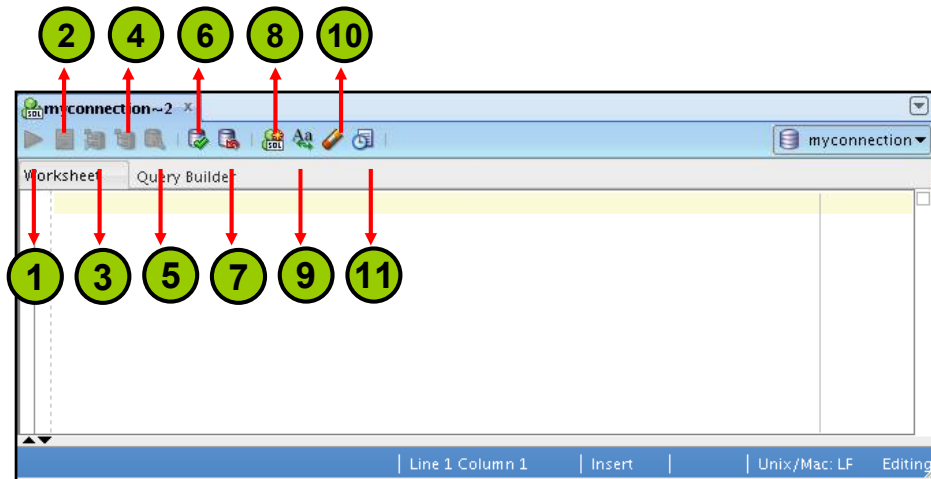
Sie können angeben, welche Aktionen die mit dem Worksheet verknüpfte Datenbankverbindung verarbeiten kann. Beispiele:

- Tabelle erstellen
- Daten einfügen
- Trigger erstellen und bearbeiten
- Daten aus einer Tabelle wählen
- Gewählte Daten in einer Datei speichern

Sie haben folgende Möglichkeiten, ein SQL Worksheet-Fenster anzuzeigen:

- Im Menü **Tools** die Option **SQL Worksheet** wählen
- Auf das Symbol **Open SQL Worksheet** klicken

SQL Worksheet



ORACLE

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

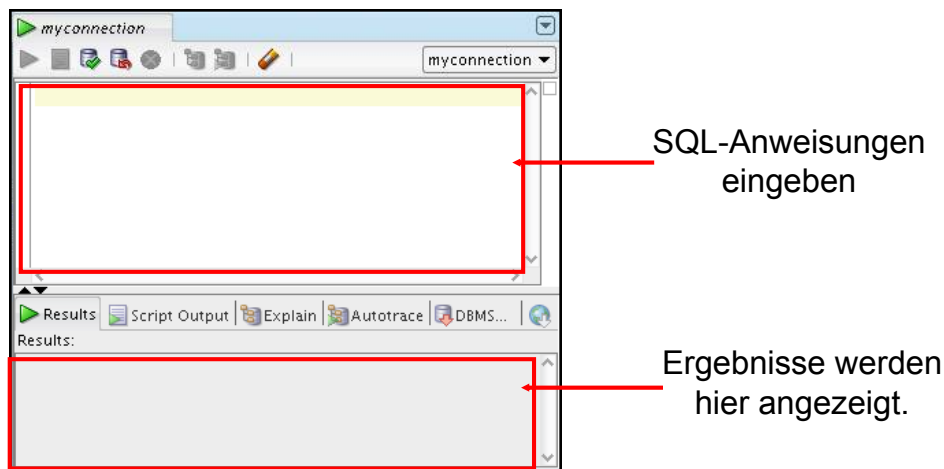
Möglicherweise möchten Sie Tastaturbefehle oder Symbole für bestimmte Aufgaben wie die Ausführung von SQL-Anweisungen und Skripten oder die Anzeige der Historie von bereits ausgeführten SQL-Anweisungen verwenden. Die Symbolleiste des SQL Worksheets enthält folgende Symbole:

1. **Run Statement:** Führt die Anweisung an der Cursorposition im Feld **Enter SQL Statement** aus. Sie können Bind-Variablen, jedoch keine Substitutionsvariablen in den SQL-Anweisungen verwenden.
2. **Run Script:** Führt mithilfe von Script Runner alle Anweisungen im Feld **Enter SQL Statement** aus. Sie können Substitutionsvariablen, jedoch keine Bind-Variablen in den SQL-Anweisungen verwenden.
3. **Autotrace:** Generiert Traceinformationen für die Anweisung
4. **Explain Plan:** Generiert den Ausführungsplan, den Sie in der Registerkarte **Explain** anzeigen können
5. **SQL Tuning Advisory:** Analysiert großvolumige SQL-Anweisungen und bietet Tuning-Empfehlungen
6. **Commit:** Schreibt alle Änderungen in der Datenbank fest und beendet die Transaktion
7. **Rollback:** Verwirft alle Änderungen in der Datenbank, ohne sie festzuschreiben, und beendet die Transaktion

8. **Unshared SQL Worksheet:** Erstellt ein separates, nicht gemeinsam genutztes SQL Worksheet für eine Verbindung
9. **To Upper/Lower/InitCap:** Ändert die Schreibweise des gewählten Textes in Groß- oder Kleinbuchstaben bzw. die Verwendung großer Anfangsbuchstaben
10. **Clear:** Löscht die Anweisungen im Feld **Enter SQL Statement**
11. **SQL History:** Zeigt ein Dialogfeld mit Informationen zu den bereits ausgeführten SQL-Anweisungen an

SQL Worksheet

- Mit dem SQL Worksheet SQL-, PL/SQL- und SQL*Plus-Anweisungen eingeben und ausführen
- Aktionen angeben, die von der mit dem Worksheet verknüpften Datenbankverbindung verarbeitet werden können



ORACLE

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

Wenn Sie sich bei einer Datenbank anmelden, wird automatisch ein SQL Worksheet-Fenster für diese Verbindung geöffnet. Im SQL Worksheet können Sie SQL-, PL/SQL- und SQL*Plus-Anweisungen eingeben und ausführen. Alle SQL- und PL/SQL-Befehle werden unterstützt und direkt vom SQL Worksheet an die Oracle-Datenbank übergeben. In SQL Developer verwendete SQL*Plus-Befehle müssen vor der Übergabe an die Datenbank vom SQL Worksheet interpretiert werden.

Das SQL Worksheet unterstützt derzeit eine Reihe von SQL*Plus-Befehlen. Nicht unterstützte SQL*Plus-Befehle werden ignoriert und nicht an die Oracle-Datenbank gesendet. Mit dem SQL Worksheet lassen sich SQL-Anweisungen und einige SQL*Plus-Befehle ausführen.

SQL-Anweisungen ausführen

Im Feld **Enter SQL Statement** eine oder mehrere SQL-Anweisungen eingeben

The screenshot illustrates the execution of an SQL query in Oracle SQL Developer. The main window, titled 'myconnection', shows the 'Query Builder' tab with the SQL statement: `SELECT employee_id, last_name from employees;`. A red box highlights the 'Execute Statement' button (a green play icon) and the 'Run Script' button (a green document icon). A red arrow points from the 'Execute Statement' button to a green circle labeled 'F9'. Another green circle labeled 'F5' is positioned over the 'Run Script' button. Below the main window, the 'Query Result' window displays the results of the query, showing a table with columns 'EMPLOYEE_ID' and 'LAST_NAME'. The results are as follows:

	EMPLOYEE_ID	LAST_NAME
1	174	Abel
2	142	Davies
3	102	De Haan
4	104	Ernst
5	202	Fay

To the right of the 'Query Result' window, the 'Script Output' window shows the message: 'Task completed in 0.01 seconds'. A green circle labeled 'F5' is positioned above this window.

ORACLE

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

Das Beispiel auf der Folie zeigt die unterschiedliche Ausgabe derselben Abfrage, einmal für die Taste F9 oder das Symbol **Execute Statement** und im Vergleich dazu für die Taste F5 oder das Symbol **Run Script**.

SQL-Skripte speichern

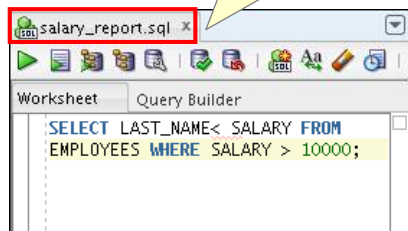
1

Klicken Sie auf das Symbol **Save**, um die SQL-Anweisung in einer Datei zu speichern.



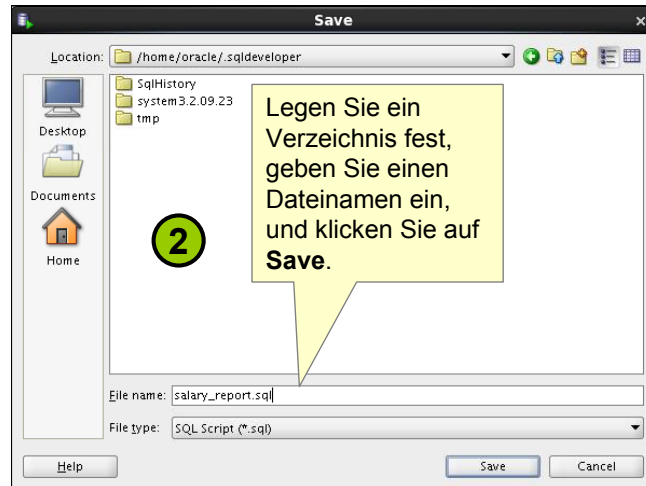
3

Der Inhalt der gespeicherten Datei kann im SQL Worksheet-Fenster angezeigt und bearbeitet werden.



2

Legen Sie ein Verzeichnis fest, geben Sie einen Dateinamen ein, und klicken Sie auf **Save**.



ORACLE

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

Sie können die SQL-Anweisungen im SQL Worksheet in einer Textdatei speichern. Um den Inhalt des Feldes **Enter SQL Statement** zu speichern, gehen Sie wie folgt vor:

1. Klicken Sie auf das Symbol **Save**, oder wählen Sie die Menüoption **File > Save**.
2. Geben Sie im Dialogfeld **Save** einen Dateinamen und das gewünschte Verzeichnis für die Datei ein.
3. Klicken Sie auf **Save**.

Nachdem Sie den Inhalt in einer Datei gespeichert haben, wird im Fenster **Enter SQL Statement** eine Registerkarte mit dem Dateiinhalt angezeigt. Es können mehrere Dateien gleichzeitig geöffnet sein. Jede Datei wird in Form einer Registerkarte angezeigt.

Skriptpfad

Sie können einen Standardpfad für die Suche nach Skripten und deren Speicherung wählen. Geben Sie hierfür unter **Tools > Preferences > Database > Worksheet Parameters** einen Wert im Feld **Select default path to look for scripts** ein.

Gespeicherte Skriptdateien ausführen – Methode 1

1. Bestimmen Sie in der Registerkarte **Files** die zu öffnende Skriptdatei.

2. Doppelklicken Sie auf das Skript, um den Code im SQL Worksheet anzuzeigen.

Klicken Sie zur Ausführung des Codes auf:

- **Run Statement** (F9) oder
- **Run Script** (F5)

Wählen Sie in der Dropdown-Liste eine Verbindung.

```

SELECT count(*) FROM tab;
SELECT count(*) FROM employees;
SELECT count(*) FROM countries;
SELECT count(*) FROM regions;
    
```

ORACLE

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

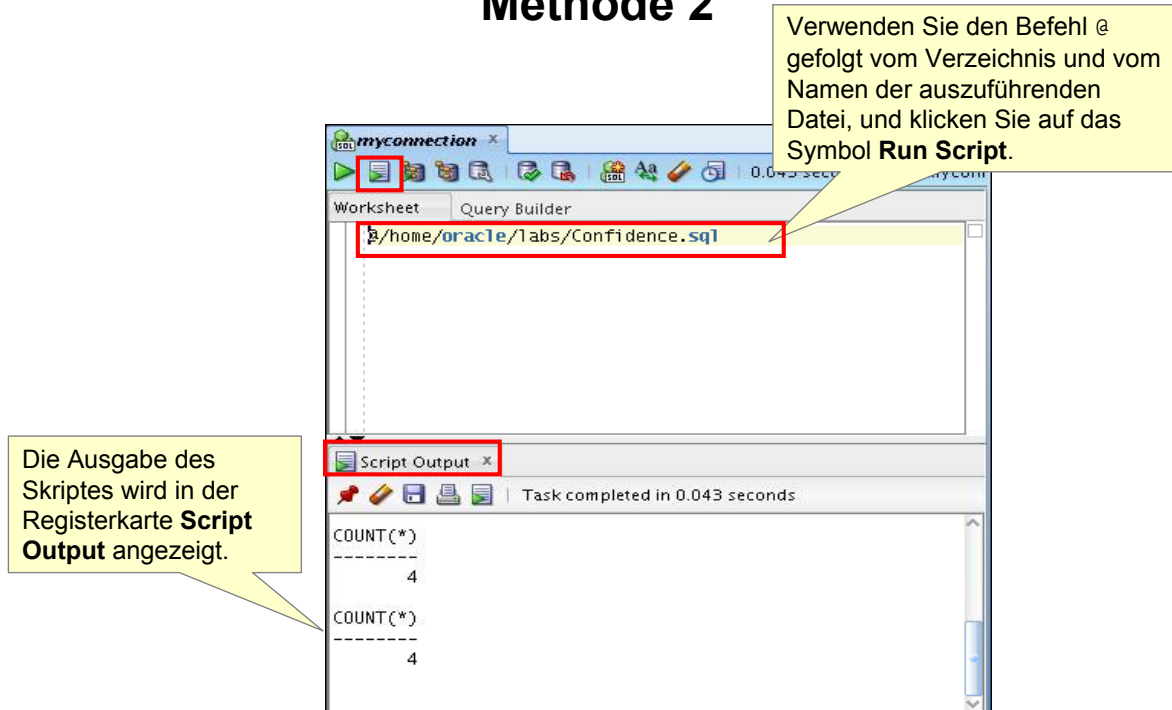
Um eine Skriptdatei zu öffnen und den Code im SQL Worksheet-Bereich anzuzeigen, gehen Sie wie folgt vor:

1. Wählen Sie im File Navigator die zu öffnende Skriptdatei, oder navigieren Sie zur gewünschten Datei.
2. Doppelklicken Sie auf die Datei, um sie zu öffnen. Der Code der Skriptdatei wird im SQL Worksheet-Bereich angezeigt.
3. Wählen Sie in der Dropdown-Liste **Connection** eine Verbindung.
4. Um den Code auszuführen, klicken Sie in der SQL Worksheet-Symbolleiste auf das Symbol **Run Script** (F5). Wenn Sie in der Dropdown-Liste **Connection** keine Verbindung gewählt haben, wird das Dialogfeld **Connection** angezeigt. Wählen Sie die gewünschte Verbindung für die Ausführung des Skriptes.

Alternativ können Sie auch wie folgt vorgehen:

1. Wählen Sie **File > Open**. Das Dialogfeld **Open** wird angezeigt.
2. Wählen Sie im Dialogfeld **Open** die zu öffnende Skriptdatei, oder navigieren Sie zur gewünschten Datei.
3. Klicken Sie auf **Open**. Der Code der Skriptdatei wird im SQL Worksheet-Bereich angezeigt.
4. Wählen Sie in der Dropdown-Liste **Connection** eine Verbindung.
5. Um den Code auszuführen, klicken Sie in der SQL Worksheet-Symbolleiste auf das Symbol **Run Script** (F5). Wenn Sie in der Dropdown-Liste **Connection** keine Verbindung gewählt haben, wird das Dialogfeld **Connection** angezeigt. Wählen Sie die gewünschte Verbindung für die Ausführung des Skriptes.

Gespeicherte Skriptdateien ausführen – Methode 2



ORACLE

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

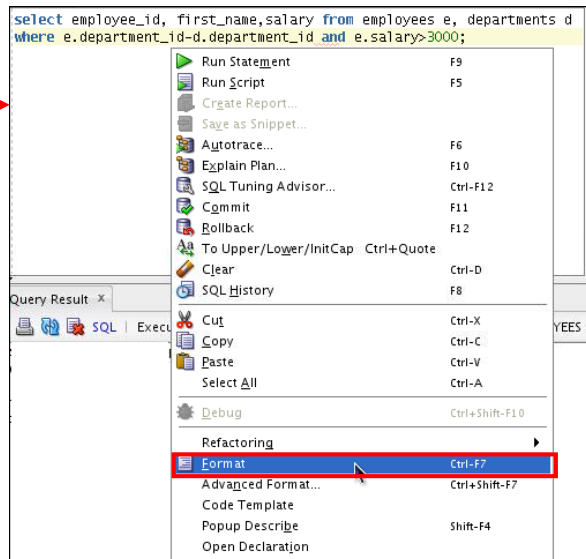
Um ein gespeichertes SQL-Skript auszuführen, gehen Sie wie folgt vor:

1. Verwenden Sie im Fenster **Enter SQL Statement** den Befehl @ gefolgt vom Verzeichnis und Namen der auszuführenden Datei.
2. Klicken Sie auf das Symbol **Run Script**.

Das Ergebnis der Skriptausführung wird in der Registerkarte **Script Output** angezeigt. Sie können die Skriptausgabe auch speichern, indem Sie in der Registerkarte **Script Output** auf das Symbol **Save** klicken. Im angezeigten Dialogfeld **File Save** können Sie Name und Verzeichnis für die Datei angeben.

SQL-Code formatieren

Vor der
Formatierung



Nach der
Formatierung

```
SELECT employee_id,
       first_name,
       salary
FROM employees e,
     departments d
WHERE e.department_id=d.department_id
AND e.salary>3000;
```

ORACLE

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

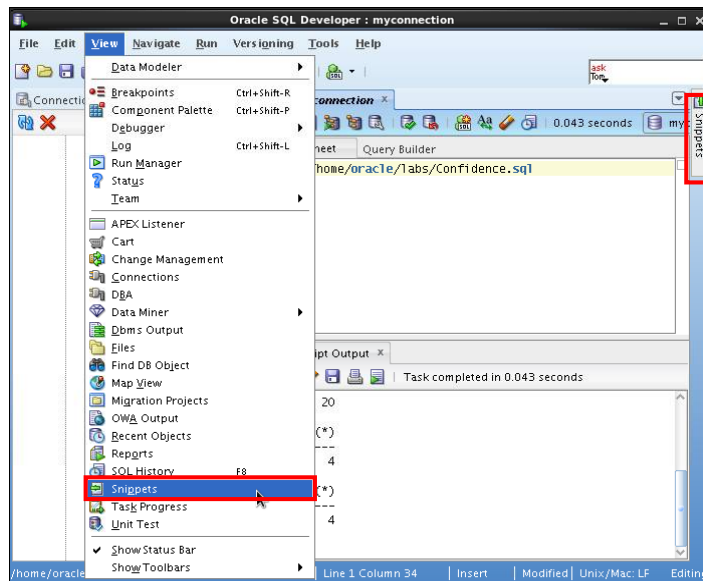
Es kann sinnvoll sein, die Einrückungen, Zeichenabstände, Groß-/Kleinschreibung und Zeilenabstände im SQL-Code übersichtlicher zu gestalten. SQL Developer bietet ein Feature zur Formatierung von SQL-Code.

Um SQL-Code zu formatieren, klicken Sie mit der rechten Maustaste in den Anweisungsbereich und wählen **Format SQL**.

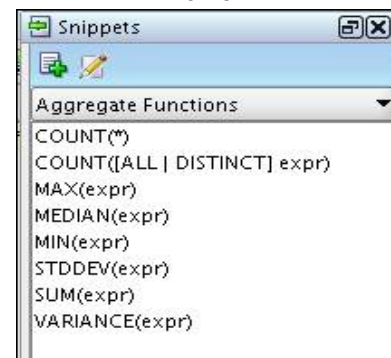
Das Beispiel auf der Folie zeigt, dass die Schlüsselwörter im SQL-Code vor der Formatierung nicht in Großbuchstaben dargestellt werden und die Anweisung nicht die richtigen Einrückungen aufweist. Nach der Formatierung ist der SQL-Code übersichtlicher, da die Schlüsselwörter groß geschrieben sind und die Anweisung mit den richtigen Einrückungen versehen ist.

Snippets

Snippets sind Codefragmente, die unter Umständen nur Syntax oder Beispiele enthalten.



Wenn Sie den Cursor hier platzieren, wird das Fenster **Snippets** angezeigt. Aus der Dropdown-Liste können Sie die gewünschte Funktionskategorie wählen.



ORACLE

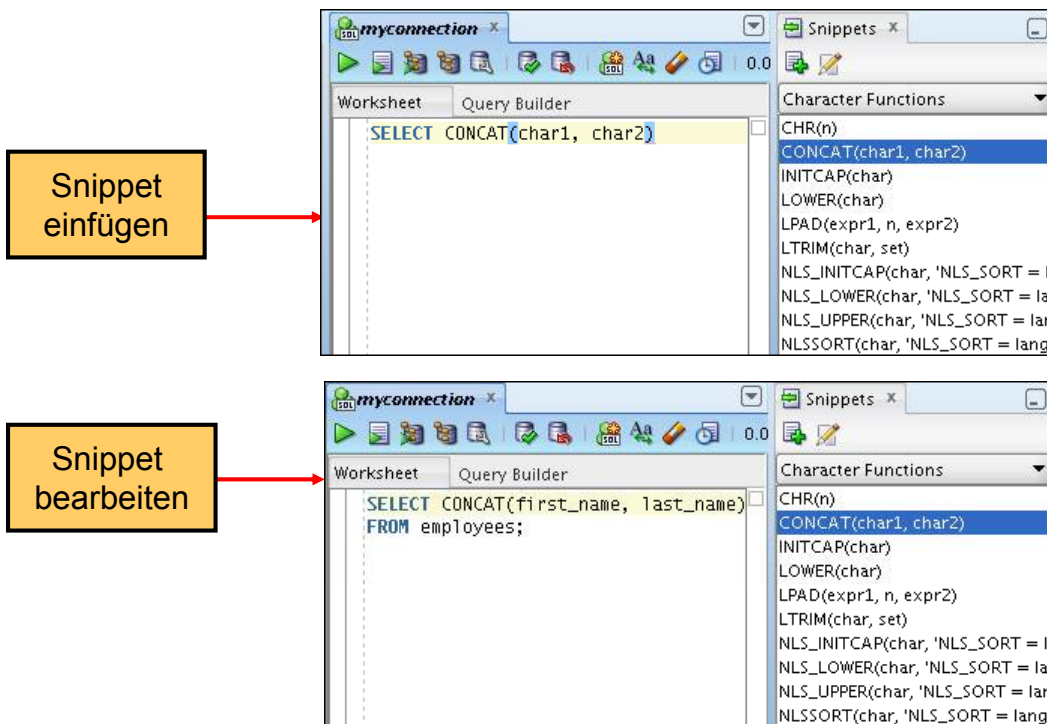
Copyright © 2014, Oracle und/oder verbundene Unternehmen. All rights reserved. Alle Rechte vorbehalten.

Bei der Arbeit mit dem SQL Worksheet oder der Erstellung oder Bearbeitung von PL/SQL-Funktionen oder -Prozeduren möchten Sie eventuell bestimmte Codefragmente verwenden. SQL Developer verfügt hierzu über das Feature Snippets. Snippets sind Codefragmente wie SQL-Funktionen, Optimizer Hints oder verschiedene PL/SQL-Programmiertechniken. Sie können Snippets in das Editorfenster ziehen.

Um Snippets anzuzeigen, wählen Sie **View > Snippets**.

Daraufhin wird rechts das Fenster **Snippets** angezeigt. Wählen Sie eine Gruppe aus der Dropdown-Liste. Über die Schaltfläche **Snippets** am rechten Fensterrand können Sie das ausgeblendete Fenster **Snippets** einblenden.

Snippets – Beispiel



ORACLE

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

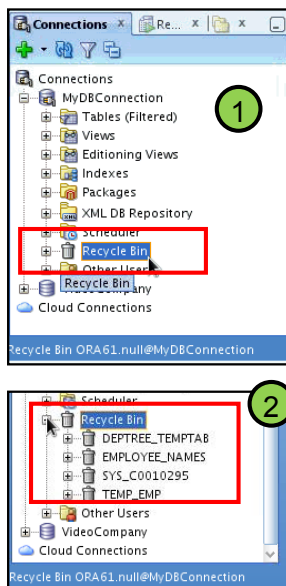
Um ein Snippet in den Code in einem SQL Worksheet oder in eine PL/SQL-Funktion oder -Prozedur einzufügen, ziehen Sie es aus dem Fenster **Snippets** an die gewünschte Stelle im Code. Danach können Sie die Syntax bearbeiten, sodass die SQL-Funktion im aktuellen Kontext gültig ist. Um eine kurze Beschreibung einer SQL-Funktion als QuickInfo anzuzeigen, platzieren Sie den Cursor über dem Funktionsnamen.

Im Beispiel auf der Folie wird `CONCAT(char1, char2)` aus der Gruppe **Character Functions** in das Fenster **Snippets** gezogen. Anschließend wird die Syntax der Funktion `CONCAT` bearbeitet und der restliche Teil der Anweisung wie folgt hinzugefügt:

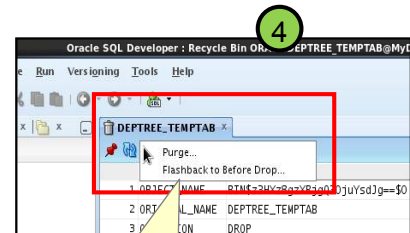
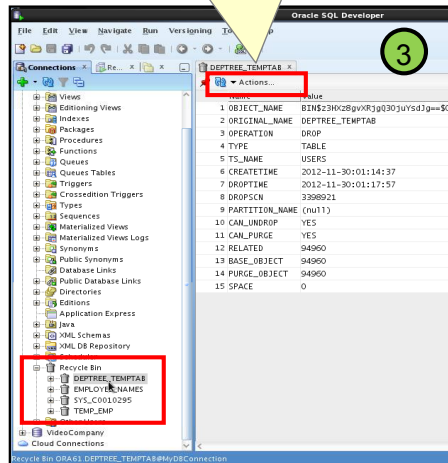
```
SELECT CONCAT(first_name, last_name)
FROM employees;
```

Papierkorb

Im Papierkorb werden gelöschte Objekte aufbewahrt.



Wählen Sie in der Dropdown-Liste **Actions** einen Vorgang.



Purge: Entfernt Objekte aus dem Papierkorb und löscht sie
Flashback to Before Drop: Verschiebt Objekte aus dem Papierkorb zurück an die ursprüngliche Position im Connections Navigator

ORACLE

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

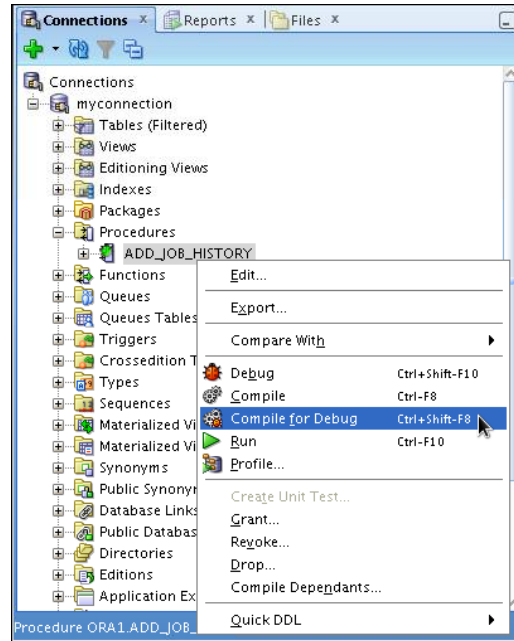
Der Papierkorb ist eine Data Dictionary-Tabelle, die Informationen zu gelöschten Objekten enthält. Gelöschte Tabellen und alle zugehörigen Objekte (wie Indizes, Constraints und Nested Tables) werden nicht entfernt und belegen weiterhin Speicherplatz. Sie werden so lange in den Speicherplatz-Quotas für Benutzer berücksichtigt, bis sie explizit dauerhaft aus dem Papierkorb gelöscht werden oder die unwahrscheinliche Situation eintritt, dass sie von der Datenbank aufgrund von Tablespace-Speicherplatzbeschränkungen dauerhaft gelöscht werden müssen.

Um den Papierkorb zu verwenden, gehen Sie wie folgt vor:

1. Navigieren Sie im Connections Navigator zum Papierkorb.
2. Erweitern Sie den Papierkorb, und klicken Sie auf den Objektnamen. Die Objektdetails werden im SQL Worksheet-Bereich angezeigt.
3. Klicken Sie auf die Dropdown-Liste **Actions**, und wählen Sie den Vorgang, den Sie für das Objekt ausführen möchten.

Prozeduren und Funktionen debuggen

- Mit SQL Developer PL/SQL-Funktionen und -Prozeduren debuggen
- Mit der Option **Compile for Debug** eine PL/SQL-Kompilierung ausführen, sodass Sie die Prozedur debuggen können
- Mit der Menüoption **Debug** Breakpoints setzen und die Schritte **Step into** und **Step over** ausführen



ORACLE

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

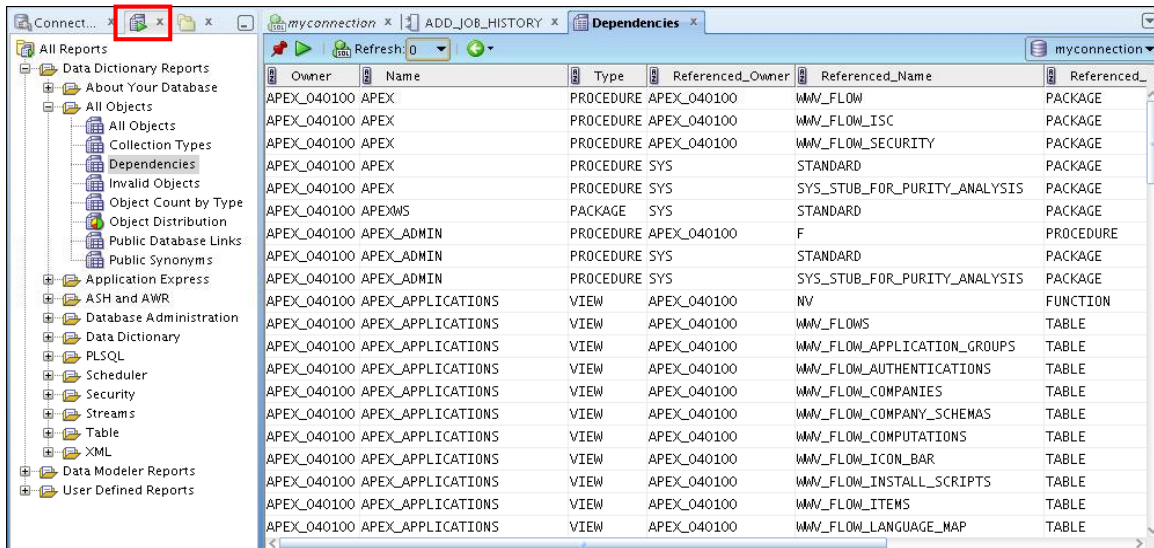
In SQL Developer können Sie PL/SQL-Prozeduren und -Funktionen debuggen. Mit den Menüoptionen zum Debuggen können Sie folgende Debugging-Aufgaben ausführen:

- **Find Execution Point** geht zum nächsten Ausführungspunkt.
- **Resume** setzt die Ausführung fort.
- **Step Over** umgeht die nächste Methode und geht zur nächsten Anweisung nach der Methode.
- **Step Into** geht zur ersten Anweisung in der nächsten Methode.
- **Step Out** verlässt die aktuelle Methode und geht zur nächsten Anweisung.
- **Step to End of Method** geht zur letzten Anweisung der aktuellen Methode.
- **Pause** unterbricht die Ausführung, beendet sie jedoch nicht. Sie können die Ausführung wieder aufnehmen.
- **Terminate** unterbricht die Ausführung und beendet sie. Sie können die Ausführung ab diesem Punkt nicht wieder aufnehmen. Um die Funktion oder Prozedur von Anfang an auszuführen oder zu debuggen, klicken Sie stattdessen in der Symbolleiste der Registerkarte **Source** auf das Symbol **Run** oder **Debug**.
- **Garbage Collection** entfernt ungültige Objekte aus dem Cache, um für gültige Objekte Platz zu machen, auf die häufiger zugegriffen wird.

Diese Optionen stehen auch als Symbole in der Symbolleiste **Debugging** zur Verfügung.

Datenbankberichte

SQL Developer bietet mehrere vordefinierte Berichte zur Datenbank und ihren Objekten.



ORACLE

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

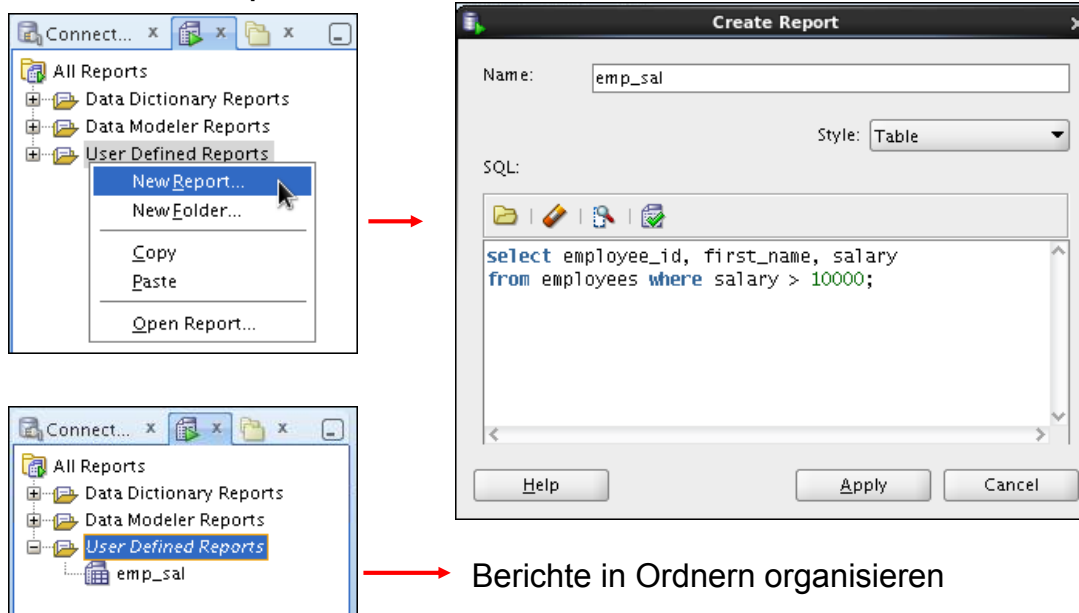
SQL Developer bietet zahlreiche Berichte zur Datenbank und ihren Objekten. Diese Berichte sind in folgende Kategorien unterteilt:

- About Your Database
- Database Administration
- Table
- PLSQL
- Security
- XML
- Jobs
- Streams
- All Objects
- Data Dictionary
- User-Defined Reports

Um Berichte anzuzeigen, klicken Sie links im Fenster auf die Registerkarte **Reports**. Die einzelnen Berichte werden rechts im Fenster im Registerkartenformat angezeigt. Bei jedem Bericht können Sie (in einer Dropdown-Liste) die Datenbankverbindung wählen, für die der Bericht angezeigt werden soll. Bei Berichten zu Objekten werden nur die Objekte angezeigt, die der Datenbankbenutzer mit der gewählten Datenbankverbindung sehen kann. Die Zeilen werden normalerweise nach dem Eigentümer (**Owner**) sortiert. Sie können auch eigene benutzerdefinierte Berichte erstellen.

Benutzerdefinierte Berichte erstellen

Benutzerdefinierte Berichte zur späteren Wiederverwendung erstellen und speichern



ORACLE

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

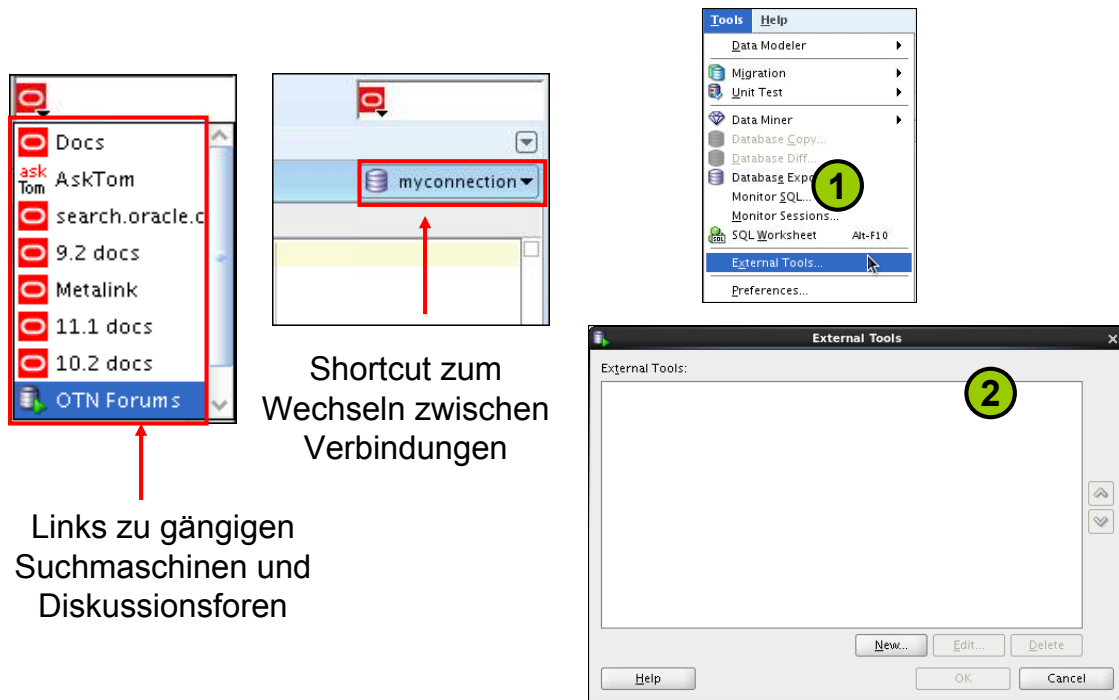
Benutzerdefinierte Berichte sind Berichte, die von SQL Developer-Benutzern erstellt werden. Um einen benutzerdefinierten Bericht zu erstellen, gehen Sie wie folgt vor:

1. Klicken Sie mit der rechten Maustaste unter **All Reports** auf den Knoten **User Defined Reports**, und wählen Sie **Add Report**.
2. Geben Sie im Dialogfeld **Create Report** den Berichtsnamen und die SQL-Abfrage an, um die Informationen für den Bericht abzurufen. Klicken Sie anschließend auf **Apply**.

Im Beispiel auf der Folie lautet der Name des Berichts `emp_sal`. Außerdem wurde eine optionale Beschreibung angegeben, aus der hervorgeht, dass der Bericht Details zu Mitarbeitern mit einem Gehalt von mindestens 10.000 enthält (`salary >= 10000`). Die vollständige SQL-Anweisung zum Abruf der Informationen für den benutzerdefinierten Bericht ist im Feld **SQL** angegeben. Sie können auch eine optionale QuickInfo aufnehmen, die angezeigt wird, wenn der Cursor in der Navigatorsanzeige für **Reports** über dem Berichtsnamen platziert wird.

Sie können benutzerdefinierte Berichte in Ordnern organisieren und eine Hierarchie von Ordnern und Unterordnern erstellen. Um einen Ordner für benutzerdefinierte Berichte zu erstellen, klicken Sie mit der rechten Maustaste auf den Knoten **User Defined Reports** oder einen beliebigen Ordernamen unter diesem Knoten und wählen **Add Folder**. Informationen über benutzerdefinierte Berichte, einschließlich der Ordner für diese Berichte, werden im Verzeichnis für benutzerspezifische Informationen in der Datei `UserReports.xml` gespeichert.

Suchmaschinen und externe Tools



ORACLE

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

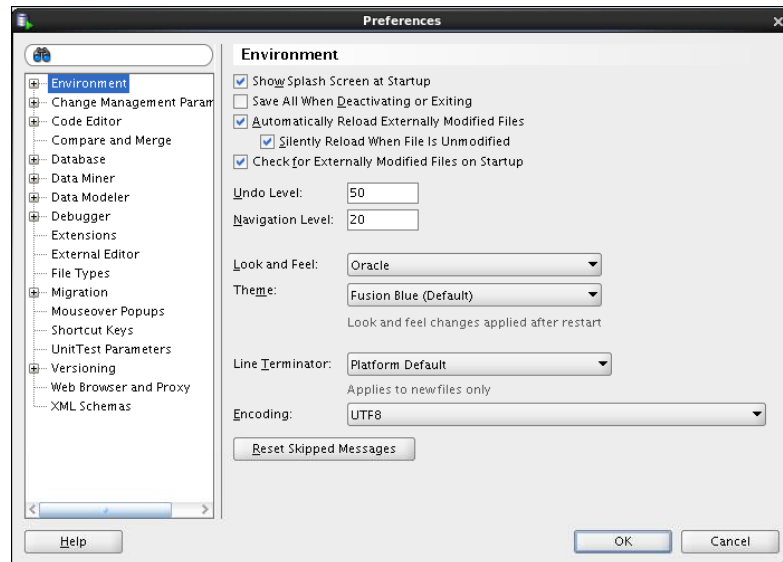
Um die Produktivität von SQL-Entwicklern zu erhöhen, wurde SQL Developer um Quicklinks zu gängigen Suchmaschinen und Diskussionsforen wie AskTom oder Google erweitert. Außerdem stehen Shortcut-Symbole zu häufig verwendeten Tools wie Notepad, Microsoft Word oder Dreamweaver zur Verfügung.

Sie können die vorhandene Liste um externe Tools ergänzen oder auch Shortcuts zu selten verwendeten Tools löschen. Gehen Sie dazu wie folgt vor:

1. Wählen Sie im Menü **Tools** die Option **External Tools**.
2. Um neue Tools hinzuzufügen, wählen Sie im Dialogfeld **External Tools** die Option **New**. Um Tools aus der Liste zu entfernen, wählen Sie **Delete**.

Voreinstellungen festlegen

- Benutzeroberfläche und Umgebung von SQL Developer anpassen
- Im Menü **Tools** die Option **Preferences** wählen



ORACLE

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

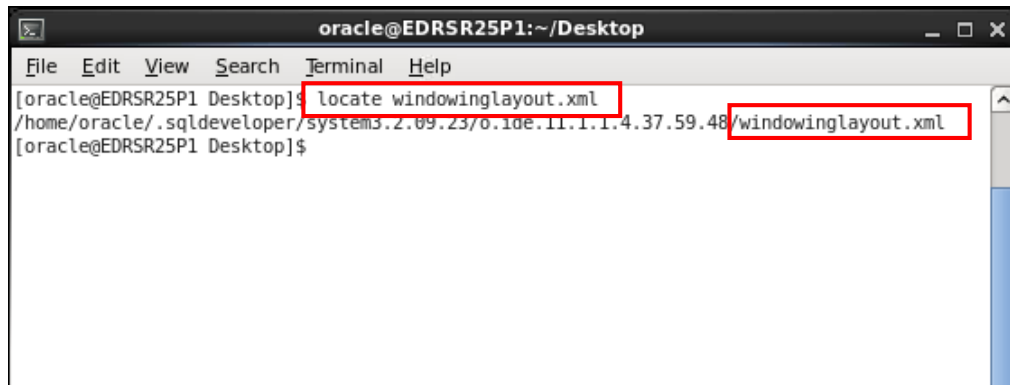
Sie können viele Aspekte der Benutzeroberfläche und der Umgebung von SQL Developer anpassen, indem Sie die SQL Developer-Voreinstellungen entsprechend Ihren Anforderungen ändern. Um SQL Developer-Voreinstellungen zu ändern, wählen Sie **Tools** und anschließend **Preferences**.

Die Voreinstellungen sind in folgende Kategorien eingeteilt:

- Environment
- Change Management Parameter
- Code Editor
- Compare and Merge
- Datenbank
- Data Miner
- Data Modeler
- Debugger
- Extensions
- External Editor
- File Types
- Migration

- Mouseover Popups
- Shortcut Keys
- Unit Test Parameters
- Versioning
- Web Browser and Proxy
- XML Schemas

SQL Developer-Layout zurücksetzen



```
oracle@EDRSR25P1:~/Desktop
File Edit View Search Terminal Help
[oracle@EDRSR25P1 Desktop]$ locate windowinglayout.xml
/home/oracle/.sqldeveloper/system3.2.09.23/6.10e.11.1.1.4.37.59.48/windowinglayout.xml
[oracle@EDRSR25P1 Desktop]$
```

ORACLE

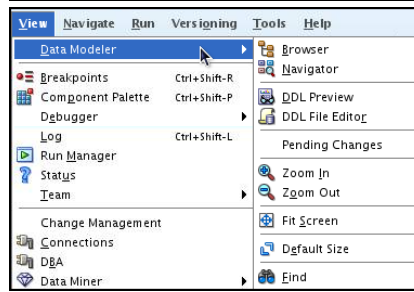
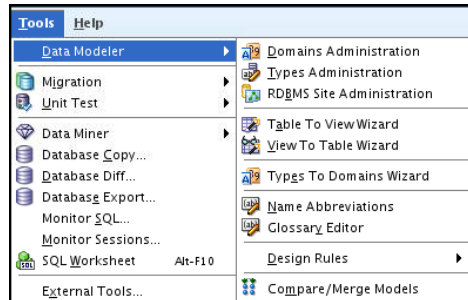
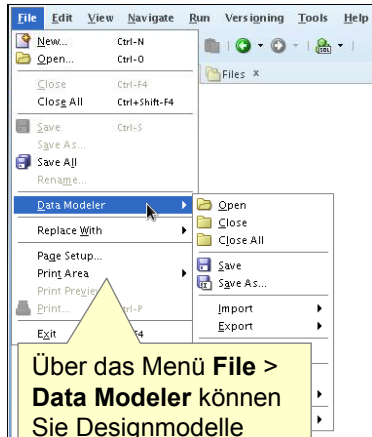
Copyright © 2014, Oracle und/oder verbundene Unternehmen. All rights reserved. Alle Rechte vorbehalten.

Wenn der Connections Navigator während der Arbeit mit SQL Developer ausgeblendet wird oder wenn Sie das Fenster **Log** nicht an seiner ursprünglichen Position verankern können, beheben Sie das Problem wie folgt:

1. Beenden Sie SQL Developer.
2. Öffnen Sie ein Terminalfenster, und verwenden Sie den Befehl `locate`, um den Speicherort von `windowinglayout.xml` zu ermitteln.
3. Navigieren Sie in das Verzeichnis, in dem sich `windowinglayout.xml` befindet, und löschen Sie diese Datei.
4. Starten Sie SQL Developer neu.

Data Modeler in SQL Developer

SQL Developer enthält eine integrierte Version von SQL Developer Data Modeler.



ORACLE

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

Mit der integrierten Version von SQL Developer Data Modeler können Sie:

- Datenbankdesigns erstellen, öffnen, importieren und speichern
- Data Modeler-Objekte erstellen, bearbeiten und löschen

Um Data Modeler in einem Bereich anzuzeigen, klicken Sie auf **Tools** und dann auf **Data Modeler**. Das Untermenü **Data Modeler** im Menü **Tools** enthält zusätzliche Befehle, etwa zur Festlegung von Designregeln und Voreinstellungen.

Zusammenfassung

In diesem Anhang haben Sie gelernt, mit SQL Developer folgende Aufgaben auszuführen:

- Datenbankobjekte durchsuchen, erstellen und bearbeiten
- SQL-Anweisungen und -Skripte im SQL Worksheet ausführen
- Benutzerdefinierte Berichte erstellen und speichern
- Data Modeler-Optionen in SQL Developer durchsuchen

ORACLE

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

Das kostenlose grafische Tool SQL Developer vereinfacht Aufgaben der Datenbankentwicklung. Mit SQL Developer können Sie Datenbankobjekte durchsuchen, erstellen und bearbeiten. Mit dem SQL Worksheet lassen sich SQL-Anweisungen und -Skripte ausführen. Darüber hinaus können Sie mit SQL Developer eigene spezielle Berichte erstellen und speichern, die Sie mehrfach wiederverwenden können.



ORACLE®

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

Ziele

Nach Ablauf dieses Anhangs haben Sie folgende Ziele erreicht:

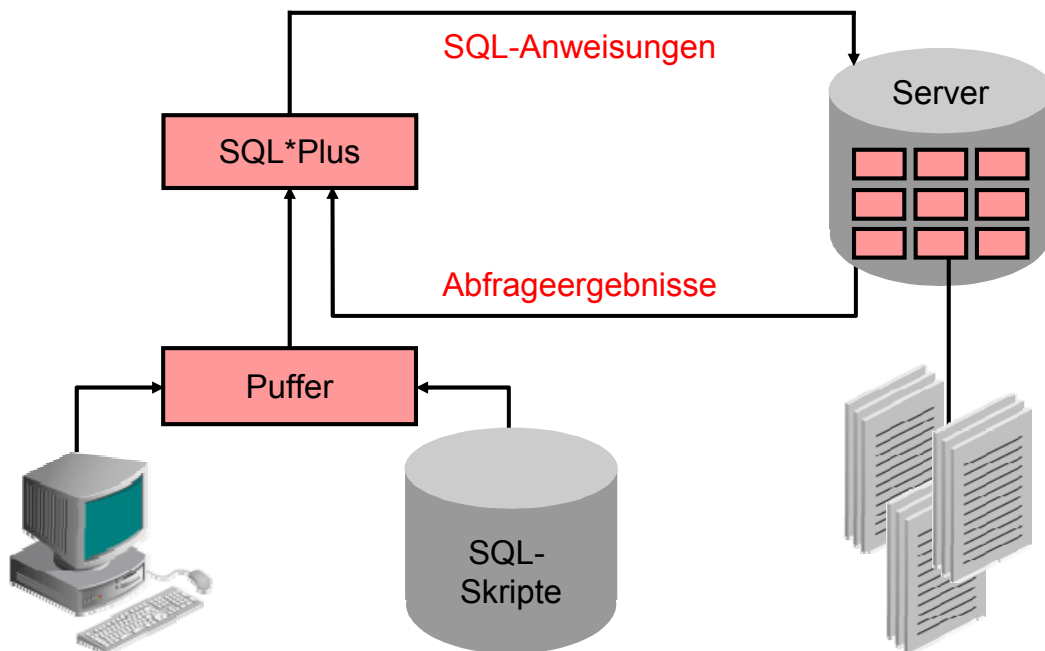
- Bei SQL*Plus anmelden
- SQL-Befehle bearbeiten
- Ausgabe mithilfe von SQL*Plus-Befehlen formatieren
- Mit Skriptdateien interagieren

ORACLE

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

Sie können `SELECT`-Anweisungen erstellen und mehrfach wiederverwenden. In diesem Anhang wird die Ausführung von SQL-Anweisungen mithilfe von SQL*Plus-Befehlen behandelt. Außerdem wird beschrieben, wie Sie die Ausgabe mit SQL*Plus-Befehlen formatieren, SQL-Befehle bearbeiten und Skripte in SQL*Plus speichern.

SQL und SQL*Plus – Interaktion



ORACLE

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

SQL und SQL*Plus

SQL ist eine Befehlssprache, mit deren Hilfe beliebige Tools und Anwendungen mit dem Oracle-Server kommunizieren können. Oracle SQL enthält zahlreiche Erweiterungen. Wenn Sie eine SQL-Anweisung eingeben, wird sie in einem Bereich des Speichers abgelegt, der als *SQL-Puffer* bezeichnet wird. Sie verbleibt dort, bis Sie eine neue SQL-Anweisung eingeben. SQL*Plus ist ein Oracle-Tool, das SQL-Anweisungen erkennt und zur Ausführung an den Oracle 9i Server weiterleitet. Es enthält eine eigene Befehlssprache.

SQL – Features

- Kann von allen Benutzern verwendet werden, auch von Benutzern ohne oder mit geringen Programmiererfahrungen
- Ist eine nicht prozedurale Sprache
- Reduziert den Zeitaufwand für Erstellung und Verwaltung von Systemen
- Ist an das Englische angelehnt

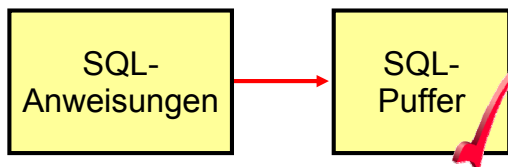
SQL*Plus – Features

- Akzeptiert die Ad-hoc-Eingabe von Anweisungen
- Akzeptiert die SQL-Eingabe aus Dateien
- Bietet einen Zeileneditor zum Ändern von SQL-Anweisungen
- Steuert die Umgebungseinstellungen
- Formatiert Abfrageergebnisse zu Basisberichten
- Greift auf lokale Datenbanken und Remote-Datenbanken zu

SQL-Anweisungen und SQL*Plus-Befehle – Vergleich

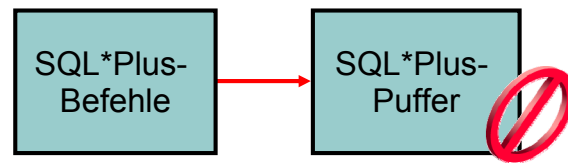
SQL

- Eine Sprache
- ANSI-Standard
- Schlüsselwörter dürfen nicht abgekürzt werden.
- Anweisungen bearbeiten Daten und Tabellen- definitionen in der Datenbank.



SQL*Plus

- Eine Umgebung
- Oracle-Tool
- Schlüsselwörter können abgekürzt werden.
- Befehle erlauben keine Bearbeitung von Werten in der Datenbank.



ORACLE

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

Die folgende Tabelle stellt SQL und SQL*Plus gegenüber:

SQL	SQL*Plus
Eine Sprache für die Kommunikation mit dem Oracle-Server beim Datenzugriff	Erkennt SQL-Anweisungen und sendet sie an den Server
Basiert auf Standard-SQL des American National Standards Institute (ANSI)	Oracle-eigene Schnittstelle zur Ausführung von SQL-Anweisungen
Bearbeitet Daten und Tabellendefinitionen in der Datenbank	Ermöglicht keine Bearbeitung von Werten in der Datenbank
Wird in einer oder mehreren Zeilen des SQL-Puffers eingegeben	Wird zeilenweise eingegeben und nicht im SQL-Puffer gespeichert
Hat kein Fortsetzungszeichen	Verwendet einen Bindestrich (–) als Fortsetzungszeichen, wenn der Befehl länger als eine Zeile ist
Kann nicht abgekürzt werden	Kann abgekürzt werden
Verwendet ein Abschlusszeichen, um Befehle sofort auszuführen	Benötigt kein Abschlusszeichen. Befehle werden sofort ausgeführt.
Verwendet Funktionen zur Ausführung von Formatierungen	Verwendet Befehle zum Formatieren von Daten

SQL*Plus – Überblick

- Bei SQL*Plus anmelden
- Tabellenstruktur beschreiben
- SQL-Anweisungen bearbeiten
- SQL über SQL*Plus ausführen
- SQL-Anweisungen in Dateien speichern und an Dateien anhängen
- Gespeicherte Dateien ausführen
- Befehle aus der Datei zur Bearbeitung in den Puffer laden

ORACLE

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

SQL*Plus

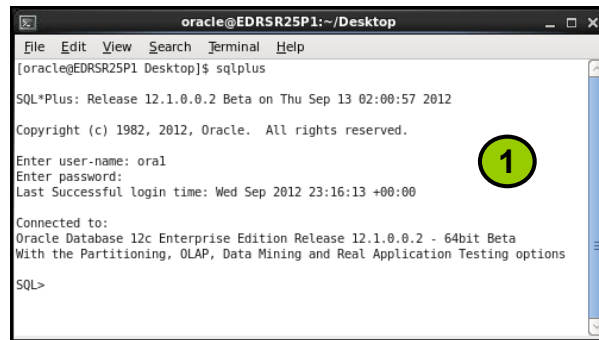
In der SQL*Plus-Umgebung können Sie folgende Aktionen ausführen:

- SQL-Anweisungen ausführen, um Daten aus der Datenbank abzurufen, zu ändern, hinzuzufügen und zu entfernen
- Abfrageergebnisse als Berichte formatieren, speichern und drucken sowie Berechnungen mit den Abfrageergebnissen ausführen
- Skriptdateien erstellen, um SQL-Anweisungen für die spätere Wiederverwendung zu speichern

SQL*Plus-Befehle können in folgende Hauptkategorien unterteilt werden:

Kategorie	Zweck
Umgebung	Allgemeines Verhalten von SQL-Anweisungen für die Session beeinflussen
Format	Abfrageergebnisse formatieren
Dateibearbeitung	Skriptdateien speichern, laden und ausführen
Ausführung	SQL-Anweisungen aus dem SQL-Puffer an den Oracle-Server senden
Bearbeitung	SQL-Anweisungen im Puffer ändern
Interaktion	Variablen erstellen und an SQL-Anweisungen übergeben, Variablenwerte ausgeben und Meldungen auf dem Bildschirm anzeigen
Verschiedenes	Bei der Datenbank anmelden, SQL*Plus-Umgebung bearbeiten und Spaltendefinitionen anzeigen

Bei SQL*Plus anmelden



```
oracle@EDRSR25P1:~/Desktop
[oracle@EDRSR25P1 Desktop]$ sqlplus

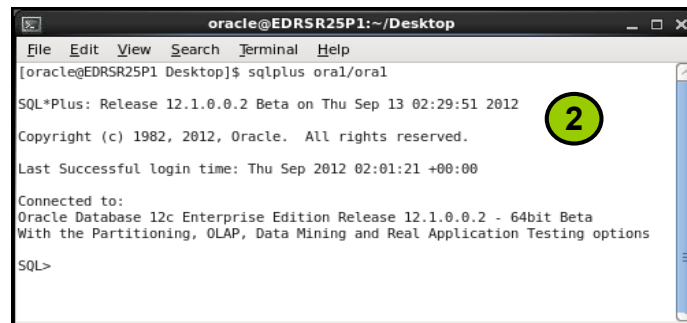
SQL*Plus: Release 12.1.0.0.2 Beta on Thu Sep 13 02:00:57 2012
Copyright (c) 1982, 2012, Oracle. All rights reserved.

Enter user-name: oral
Enter password:
Last Successful login time: Wed Sep 2012 23:16:13 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.0.2 - 64bit Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
```

```
sqlplus [username[/password[@database]]]
```



```
oracle@EDRSR25P1:~/Desktop
[oracle@EDRSR25P1 Desktop]$ sqlplus oral/oral

SQL*Plus: Release 12.1.0.0.2 Beta on Thu Sep 13 02:29:51 2012
Copyright (c) 1982, 2012, Oracle. All rights reserved.

Last Successful login time: Thu Sep 2012 02:01:21 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.0.2 - 64bit Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
```

ORACLE®

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

Wie Sie SQL*Plus aufrufen, richtet sich nach dem Typ des Betriebssystems, auf dem Sie Oracle Database ausführen.

Um sich aus einer Linux-Umgebung anzumelden, gehen Sie wie folgt vor:

1. Klicken Sie mit der rechten Maustaste auf den Linux-Desktop, und wählen Sie **terminal**.
2. Geben Sie den auf der Folie gezeigten Befehl `sqlplus` ein.
3. Geben Sie Benutzernamen, Kennwort und Datenbanknamen ein.

Für die Syntax gilt:

<i>username</i>	Ihr Benutzernamen für die Datenbank
<i>password</i>	Ihr Kennwort für die Datenbank. (Das Kennwort ist hier bei der Eingabe sichtbar.)
<i>@database</i>	Die Verbindungszeichenfolge für die Datenbank

Hinweis: Um die Integrität des Kennwortes zu wahren, dürfen Sie es nicht in der Eingabeaufforderung des Betriebssystems eingeben. Geben Sie stattdessen nur Ihren Benutzernamen ein, und geben Sie das Kennwort in der Kennworteingabeaufforderung ein.

Tabellenstrukturen anzeigen

Struktur einer Tabelle mit dem SQL*Plus-Befehl `DESCRIBE` anzeigen:

```
DESC[RIBE] tablename
```

ORACLE

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

In SQL*Plus können Sie die Struktur einer Tabelle mit dem Befehl `DESCRIBE` anzeigen. Daraufhin werden die Spaltennamen und Datentypen sowie ein Hinweis darauf angezeigt, ob eine Spalte Daten enthalten muss.

Für die Syntax gilt:

tablename Der Name einer beliebigen vorhandenen Tabelle, einer View oder eines Synonyms, auf die bzw. das der Benutzer zugreifen kann

Die Tabelle `DEPARTMENTS` zeigen Sie mit folgendem Befehl an:

```
SQL> DESCRIBE DEPARTMENTS
```

Name	Null	Type
-----	-----	-----
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Tabellenstrukturen anzeigen

```
DESCRIBE departments
```

Name	Null	Type
-----	-----	-----
DEPARTMENT_ID	NOT NULL	NUMBER (4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2 (30)
MANAGER_ID		NUMBER (6)
LOCATION_ID		NUMBER (4)

ORACLE

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

Das Beispiel auf der Folie zeigt Informationen über die Struktur der Tabelle `DEPARTMENTS`. Für das Ergebnis gilt:

Null: Gibt an, ob eine Spalte Daten enthalten muss (`NOT NULL` gibt an, dass eine Spalte Daten enthalten muss.)

Type: Zeigt den Datentyp einer Spalte an

SQL*Plus – Bearbeitungsbefehle

- `A[PPEND] text`
- `C[HANGE] / old / new`
- `C[HANGE] / text /`
- `CL[EAR] BUFF[ER]`
- `DEL`
- `DEL n`
- `DEL m n`

ORACLE

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

SQL*Plus-Befehle werden Zeile für Zeile eingegeben und nicht im SQL-Puffer gespeichert.

Befehl	Beschreibung
<code>A[PPEND] text</code>	Fügt Text an das Ende der aktuellen Zeile an
<code>C[HANGE] / old / new</code>	Ersetzt in der aktuellen Zeile alten Text (<i>old</i>) durch neuen Text (<i>new</i>)
<code>C[HANGE] / text /</code>	Löscht Text (<i>text</i>) aus der aktuellen Zeile
<code>CL[EAR] BUFF[ER]</code>	Löscht alle Zeilen aus dem SQL-Puffer
<code>DEL</code>	Löscht die aktuelle Zeile
<code>DEL n</code>	Löscht die Zeile <i>n</i>
<code>DEL m n</code>	Löscht die Zeilen <i>m</i> bis einschließlich <i>n</i>

Richtlinien

- Wenn Sie die EINGABETASTE drücken, bevor Sie einen Befehl abgeschlossen haben, zeigt SQL*Plus die Nummer der entsprechenden Befehlszeile an.
- Sie beenden den SQL-Puffer, indem Sie eines der Abschlusszeichen (Semikolon oder Schrägstrich) eingeben oder zweimal die EINGABETASTE drücken. Anschließend wird die SQL-Eingabeaufforderung angezeigt.

SQL*Plus – Bearbeitungsbefehle

- I [NPUT]
- I [NPUT] *text*
- L [IST]
- L [IST] *n*
- L [IST] *m n*
- R [UN]
- *n*
- *n text*
- 0 *text*

ORACLE

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

Befehl	Beschreibung
I [NPUT]	Fügt eine unbestimmte Anzahl von Zeilen ein
I [NPUT] <i>text</i>	Fügt eine Zeile mit Text (<i>text</i>) ein
L [IST]	Listet alle Zeilen im SQL-Puffer auf
L [IST] <i>n</i>	Listet eine Zeile auf (durch <i>n</i> angegeben)
L [IST] <i>m n</i>	Listet einen Bereich von Zeilen auf (<i>m</i> bis einschließlich <i>n</i>)
R [UN]	Zeigt die aktuell im Puffer befindliche SQL-Anweisung an und führt sie aus
<i>n</i>	Macht <i>n</i> zur aktuellen Zeile
<i>n text</i>	Ersetzt die Zeile <i>n</i> durch den Text (<i>text</i>)
0 <i>text</i>	Fügt eine Zeile vor Zeile 1 ein

Hinweis: Sie können nur jeweils einen SQL*Plus-Befehl pro SQL-Eingabeaufforderung eingeben. SQL*Plus-Befehle werden nicht im SQL-Puffer gespeichert. Um einen SQL*Plus-Befehl in der nächsten Zeile fortzusetzen, geben Sie am Ende der ersten Zeile einen Bindestrich (-) ein.

LIST, n und APPEND

```
LIST
1  SELECT last_name
2* FROM    employees
```

```
1
1* SELECT last_name
```

```
A , job_id
1* SELECT last_name, job_id
```

```
LIST
1  SELECT last_name, job_id
2* FROM    employees
```

ORACLE

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

- Mit dem Befehl `L[IST]` zeigen Sie den Inhalt des SQL-Puffers an. Das Sternchen (*) neben der 2. Zeile im Puffer gibt an, dass die 2. Zeile die aktuelle Zeile ist. Alle eingegebenen Bearbeitungsbefehle gelten für die aktuelle Zeile.
- Sie können die aktuelle Zeile wechseln, indem Sie die Nummer (n) der Zeile eingeben, die Sie bearbeiten möchten. Die neue aktuelle Zeile wird angezeigt.
- Mit dem Befehl `A[PPEND]` fügen Sie der aktuellen Zeile Text hinzu. Die neu bearbeitete Zeile wird angezeigt. Den neuen Inhalt des Puffers prüfen Sie mit dem Befehl `LIST`.

Hinweis: Viele SQL*Plus-Befehle, einschließlich `LIST` und `APPEND`, können mit ihrem ersten Buchstaben abgekürzt werden. `LIST` lässt sich mit `L` abkürzen, `APPEND` mit `A`.

Befehl CHANGE

```
LIST  
1* SELECT * from employees
```

```
c/employees/departments  
1* SELECT * from departments
```

```
LIST  
1* SELECT * from departments
```

ORACLE

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

- Mit `L[IST]` zeigen Sie den Inhalt des Puffers an.
- Mit dem Befehl `C[HANGE]` ändern Sie den Inhalt der aktuellen Zeile im SQL-Puffer. In diesem Fall ersetzen Sie die Tabelle `employees` durch die Tabelle `departments`. Die neue aktuelle Zeile wird angezeigt.
- Mit dem Befehl `L[IST]` prüfen Sie den neuen Inhalt des SQL-Puffers.

SQL*Plus – Dateibefehle

- `SAVE filename`
- `GET filename`
- `START filename`
- `@ filename`
- `EDIT filename`
- `SPOOL filename`
- `EXIT`

ORACLE

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

SQL-Anweisungen kommunizieren mit dem Oracle-Server. SQL*Plus-Befehle steuern die Umgebung, formatieren Abfrageergebnisse und verwalten Dateien. Sie können die in der folgenden Tabelle beschriebenen Befehle verwenden:

Befehl	Beschreibung
<code>SAV[E] filename [.ext]</code> <code>[REPL[ACE]APP[END]]</code>	Speichert den aktuellen Inhalt des SQL-Puffers in einer Datei. Verwenden Sie <code>APPEND</code> , um den Pufferinhalt einer vorhandenen Datei hinzuzufügen, oder <code>REPLACE</code> , um eine vorhandene Datei zu ersetzen. Die Standardextension ist <code>.sql</code> .
<code>GET filename [.ext]</code>	Schreibt den Inhalt einer zuvor gespeicherten Datei in den SQL-Puffer. Die Standardextension für den Dateinamen ist <code>.sql</code> .
<code>STA[RT] filename [.ext]</code>	Führt eine zuvor gespeicherte Befehlsdatei aus
<code>@ filename</code>	Führt eine zuvor gespeicherte Befehlsdatei aus (identisch mit <code>START</code>)

Befehl	Beschreibung
ED[IT]	Ruft den Editor auf und speichert den Pufferinhalt in der Datei <code>afiedt.buf</code>
ED[IT] [<i>filename</i> [.ext]]	Ruft den Editor auf, um den Inhalt einer gespeicherten Datei zu bearbeiten
SPO[OL] [<i>filename</i> [.ext]] OFF OUT]	Speichert die Abfrageergebnisse in einer Datei. OFF schließt die Spool-Datei. OUT schließt die Spool-Datei und sendet die Dateiergebnisse an den Drucker.
EXIT	Beendet SQL*Plus

Befehle SAVE und START

```
LIST
1  SELECT last_name, manager_id, department_id
2*  FROM employees
```

```
SAVE my_query
Created file my_query
```

```
START my_query

LAST_NAME                MANAGER_ID DEPARTMENT_ID
-----
King                      100             90
Kochhar                   100             90
...
107 rows selected.
```

ORACLE

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

SAVE

Mit dem Befehl `SAVE` speichern Sie den aktuellen Inhalt des Puffers in einer Datei. So können Sie häufig verwendete Skripte zur späteren Wiederverwendung speichern.

START

Mit dem Befehl `START` führen Sie ein Skript in SQL*Plus aus. Alternativ können Sie Skripte auch über das Symbol `@` ausführen.

```
@my_query
```

Befehl SERVEROUTPUT

- Mit dem Befehl `SET SERVEROUT [PUT]` steuern Sie, ob die Ausgabe von gespeicherten Prozeduren und PL/SQL-Blöcken in SQL*Plus angezeigt wird.
- Die Zeilenlängenbeschränkung für `DBMS_OUTPUT` wurde von 255 Byte auf 32767 Byte erhöht.
- Die Standardgröße ist nun unbegrenzt.
- Ist `SERVEROUTPUT` festgelegt, werden keine Ressourcen reserviert.
- Verwenden Sie `UNLIMITED`, da damit keine Performance-einbußen verbunden sind. Ausnahme: Sie möchten physischen Speicher einsparen.

```
SET SERVEROUT[PUT] {ON | OFF} [SIZE {n | UNL[IMITED]}]  
[FOR[MAT] {WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]}]
```

ORACLE

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

Die meisten PL/SQL-Programme führen Ein- und Ausgaben über SQL-Anweisungen durch, um Daten in Datenbanktabellen zu speichern oder diese Tabellen abzufragen. Alle anderen PL/SQL-Eingaben/Ausgaben erfolgen über APIs, die mit anderen Programmen interagieren. Beispiel: Das Package `DBMS_OUTPUT` verfügt über Prozeduren wie `PUT_LINE`. Um das Ergebnis außerhalb von PL/SQL anzuzeigen, ist zum Lesen und Anzeigen der an `DBMS_OUTPUT` übergebenen Daten ein anderes Programm (etwa SQL*Plus) erforderlich.

SQL*Plus zeigt Daten von `DBMS_OUTPUT` nur an, wenn Sie zuvor den SQL*Plus-Befehl `SET SERVEROUTPUT ON` wie folgt abgesetzt haben:

```
SET SERVEROUTPUT ON
```

Hinweise

- `SIZE` legt die Anzahl der Ausgabebyte fest, die im Oracle Database-Server gepuffert werden können. Der Standardwert ist `UNLIMITED`. `n` darf nicht unter 2.000 oder über 1.000.000 liegen.
- Weitere Informationen zu `SERVEROUTPUT` finden Sie im *Oracle Database PL/SQL User's Guide and Reference 12c*.

SQL*Plus-Befehl SPOOL

```
SPO[OL] [file_name[.ext] [CRE[ATE] | REP[LACE] |  
APP[END]] | OFF | OUT]
```

Option	Beschreibung
file_name[.ext]	Schreibt die Ausgabe in die angegebene Datei
CRE[ATE]	Erstellt eine neue Datei mit dem angegebenen Namen
REP[LACE]	Ersetzt den Inhalt einer vorhandenen Datei. Ist die Datei nicht vorhanden, wird sie durch REPLACE erstellt.
APP[END]	Fügt den Pufferinhalt am Ende der angegebenen Datei ein
OFF	Stoppt das Spool-Verfahren
OUT	Stoppt das Spool-Verfahren und sendet die Datei an den Standarddrucker des Rechners

ORACLE

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

Der Befehl `SPOOL` speichert die Abfrageergebnisse in einer Datei oder sendet die Datei optional an einen Drucker. Der Befehl `SPOOL` wurde erweitert, sodass Sie jetzt Daten an das Ende einer Datei anhängen oder eine vorhandene Datei ersetzen können. Bisher konnten Sie mit `SPOOL` lediglich Dateien erstellen (und ersetzen). `REPLACE` ist der Standardwert.

Sie können die über Befehle generierte Ausgabe mit `SET TERMOUT OFF` in ein Skript schreiben, ohne die Ausgabe auf dem Bildschirm anzuzeigen. `SET TERMOUT OFF` hat keine Auswirkungen auf die Ausgabe von Befehlen, die interaktiv ausgeführt werden.

Dateinamen, die Leerzeichen enthalten, müssen in Anführungszeichen gesetzt werden. Um mit `SPOOL APPEND`-Befehlen eine gültige HTML-Datei zu erstellen, müssen Sie mit `PROMPT` oder einem ähnlichen Befehl den Header und Footer der HTML-Seite erstellen. Der Befehl `SPOOL APPEND` parst keine HTML-Tags. Um die Parameter `CREATE`, `APPEND` und `SAVE` zu deaktivieren, stellen Sie `SQLPLUSCOMPAT[IBILITY]` auf 9.2 oder früher ein.

Befehl AUTOTRACE

- Zeigt nach der erfolgreichen Ausführung von SQL-DML-Anweisungen wie `SELECT`, `INSERT`, `UPDATE` und `DELETE` einen Bericht an
- Der Bericht kann nun Ausführungsstatistiken und den Abfrageausführungspfad umfassen.

```
SET AUTOT[RACE] {ON | OFF | TRACE[ONLY]} [EXP[LAIN]]  
[STAT[ISTICS]]
```

```
SET AUTOTRACE ON  
-- The AUTOTRACE report includes both the optimizer  
-- execution path and the SQL statement execution  
-- statistics
```

ORACLE®

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

`EXPLAIN` zeigt den Abfrageausführungspfad durch Ausführung von `EXPLAIN PLAN` an. `STATISTICS` zeigt Statistiken zu SQL-Anweisungen an. Die Formatierung des `AUTOTRACE`-Berichts kann je nach Version des Servers, bei dem Sie angemeldet sind, und je nach Serverkonfiguration variieren. Das Package `DBMS_XPLAN` ermöglicht Ihnen die einfache Anzeige der Ausgabe des Befehls `EXPLAIN PLAN` in verschiedenen vordefinierten Formaten.

Hinweise

- Weitere Informationen zum Package und zu Unterprogrammen finden Sie in der *Oracle Database PL/SQL Packages and Types Reference 12c*.
- Weitere Informationen zu `EXPLAIN PLAN` finden Sie in der *Oracle Database SQL Reference 12c*.
- Weitere Informationen zu Ausführungsplänen und Statistiken finden Sie im *Oracle Database Performance Tuning Guide 12c*.

Zusammenfassung

In diesem Anhang haben Sie gelernt, SQL*Plus als Umgebung für folgende Aufgaben zu verwenden:

- SQL-Anweisungen ausführen
- SQL-Anweisungen bearbeiten
- Ausgabe formatieren
- Mit Skriptdateien interagieren

ORACLE

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

SQL*Plus ist eine Ausführungsumgebung, mit der Sie SQL-Befehle an den Datenbankserver senden sowie SQL-Befehle bearbeiten und speichern. Die Ausführung der Befehle erfolgt über die SQL-Eingabeaufforderung oder eine Skriptdatei.

Häufig verwendete SQL-Befehle



ORACLE®

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

Ziele

Nach Ablauf dieses Anhangs haben Sie folgende Ziele erreicht:

- Einfache `SELECT`-Anweisungen ausführen
- Mit DDL-Anweisungen Tabellen erstellen, ändern und löschen
- Mit DML-Anweisungen Zeilen aus einer oder mehreren Tabellen einfügen, aktualisieren und löschen
- Savepoints mit Anweisungen zur Transaktionskontrolle festschreiben, zurücksetzen und erstellen
- Join-Vorgänge für eine oder mehrere Tabellen durchführen

ORACLE

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

In dieser Lektion wird erläutert, wie Sie mit `SELECT`-Anweisungen Daten aus einer oder mehreren Tabellen abrufen, mit DDL-Anweisungen die Struktur von Datenobjekten ändern sowie mit DML-Anweisungen Daten in vorhandenen Schemaobjekten bearbeiten und die vorgenommenen Änderungen verwalten. Darüber hinaus lernen Sie, mithilfe von Joins und der SQL:1999-konformen Join-Syntax Daten aus mehreren Tabellen anzuzeigen.

Einfache SELECT-Anweisungen

- Mit SELECT-Anweisungen können Sie:
 - die anzuzeigenden Spalten bestimmen
 - Daten aus einzelnen oder mehreren Tabellen, Objekttabellen, Views, Objekt-Views und Materialized Views abrufen
- SELECT-Anweisungen werden auch als Abfragen bezeichnet, da sie eine Datenbank abfragen.
- Syntax:

```
SELECT { * | [DISTINCT] column | expression [alias], ... }  
FROM      table;
```

ORACLE

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

Eine SELECT-Anweisung muss in ihrer einfachsten Form folgende Elemente enthalten:

- Eine SELECT-Klausel, in der die anzuzeigenden Spalten angegeben sind
- Eine FROM-Klausel, in der die Tabelle mit den in der Klausel SELECT aufgeführten Spalten angegeben ist

Für die Syntax gilt:

SELECT	Listet eine oder mehrere Spalten
*	Wählt alle Spalten
DISTINCT	Unterdrückt doppelte Werte
column expression	Wählt die angegebene Spalte oder den angegebenen Ausdruck
alias	Weist den gewählten Spalten andere Überschriften zu
FROM table	Gibt die Tabelle an, die die Spalten enthält

Hinweis: Im gesamten Kurs werden die Begriffe *Schlüsselwort*, *Klausel* und *Anweisung* wie folgt verwendet:

- Ein *Schlüsselwort* bezieht sich auf ein einzelnes SQL-Element. Beispiele für Schlüsselwörter: SELECT und FROM
- Eine *Klausel* ist ein Teil einer SQL-Anweisung. Beispiel: SELECT employee_id, last_name
- Eine *Anweisung* ist eine Kombination aus zwei oder mehr Klauseln. Beispiel: SELECT * FROM employees

SELECT-Anweisungen

- Alle Spalten wählen:

```
SELECT *  
FROM job_history;
```

	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-01	24-JUL-06	IT_PROG	60
2	101	21-SEP-97	27-OCT-01	AC_ACCOUNT	110
3	101	28-OCT-01	15-MAR-05	AC_MGR	110
4	201	17-FEB-04	19-DEC-07	MK_REP	20
5	114	24-MAR-06	31-DEC-07	ST_CLERK	50
6	122	01-JAN-07	31-DEC-07	ST_CLERK	50
7	200	17-SEP-95	17-JUN-01	AD_ASST	90
8	176	24-MAR-06	31-DEC-06	SA_REP	80
9	176	01-JAN-07	31-DEC-07	SA_MAN	80
10	200	01-JUL-02	31-DEC-06	AC_ACCOUNT	90

- Bestimmte Spalten wählen:

```
SELECT manager_id, job_id  
FROM employees;
```

	MANAGER_ID	JOB_ID
1	(null)	AD_PRES
2	100	AD_VP
3	100	AD_VP
4	102	IT_PROG
5	103	IT_PROG
6	103	IT_PROG
7	100	ST_MAN
8	124	ST_CLERK
9	124	ST_CLERK
10	124	ST_CLERK

...

ORACLE

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

Um alle Datenspalten einer Tabelle anzuzeigen, können Sie nach dem Schlüsselwort `SELECT` ein Sternchen (*) angeben oder die Namen aller Spalten auflisten. Im ersten Beispiel auf der Folie werden alle Zeilen der Tabelle `job_history` angezeigt. Um bestimmte Spalten der Tabelle anzuzeigen, geben Sie die entsprechenden Spaltennamen durch Kommas getrennt an. Im zweiten Beispiel auf der Folie werden die Spalten `manager_id` und `job_id` aus der Tabelle `employees` angezeigt.

Geben Sie in der Klausel `SELECT` die Spalten in der Reihenfolge an, in der sie in der Ausgabe angezeigt werden sollen. Beispiel: Die folgende SQL-Anweisung zeigt die Spalte `location_id` vor der Spalte `department_id` an:

```
SELECT location_id, department_id FROM departments;
```

Hinweis: Um Anweisungen in SQL Developer auszuführen, können Sie die SQL-Anweisung in ein SQL Worksheet eingeben und auf das Symbol **Run Statement** klicken oder F9 drücken. Die Ausgabe wird in der Registerkarte **Results** angezeigt, wie auf der Folie dargestellt.

WHERE-Klauseln

- Mit der optionalen WHERE-Klausel:
 - Zeilen in einer Abfrage filtern
 - Teilmenge von Zeilen erstellen
- Syntax:

```
SELECT * FROM table  
[WHERE condition];
```

- Beispiele:

```
SELECT location_id from departments  
WHERE department_name = 'Marketing';
```

ORACLE

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

Die Klausel `WHERE` gibt eine Bedingung an, mit der Zeilen gefiltert werden und eine Teilmenge von Zeilen aus der Tabelle abgerufen wird. Bedingungen umfassen einen oder mehrere Ausdrücke und logische (boolesche) Operatoren. Sie geben als Ergebnis den Wert `TRUE`, `FALSE` oder `NULL` zurück. Im Beispiel auf der Folie wird die `location_id` der Marketingabteilung abgerufen.

Mit `WHERE`-Klauseln können Sie auch Daten aus der Datenbank aktualisieren oder löschen.

Beispiele:

```
UPDATE departments  
SET department_name = 'Administration'  
WHERE department_id = 20;  
und  
DELETE from departments  
WHERE department_id = 20;
```

ORDER BY-Klauseln

- Die optionale ORDER BY-Klausel regelt die Reihenfolge der Zeilen.
- Syntax:

```
SELECT * FROM table
[WHERE condition]
[ORDER BY {<column>|<position> } [ASC|DESC] [, ...] ];
```

- Beispiel:

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id ASC, salary DESC;
```

ORACLE

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

Mit der Klausel ORDER BY legen Sie fest, in welcher Reihenfolge die Zeilen angezeigt werden sollen. Zeilen können auf- oder absteigend sortiert werden. Die standardmäßige Sortierfolge für Zeilen ist aufsteigend.

Im Beispiel auf der Folie werden Zeilen aus der Tabelle `employees` abgerufen. Die Zeilen werden zuerst aufsteigend nach `department_id` und anschließend absteigend nach `salary` sortiert.

GROUP BY-Klauseln

- Mit der optionalen GROUP BY-Klausel fassen Sie Spalten mit übereinstimmenden Werten zu Untergruppen zusammen.
- In keiner Gruppe gibt es zwei Zeilen mit identischem Wert für die Spalte(n), nach der oder denen die Gruppierung erfolgt.
- Syntax:

```
SELECT <column1, column2, ... column_n>  
FROM  table  
[WHERE  condition]  
[GROUP BY <column> [, ...] ]  
[ORDER BY <column> [, ...] ] ;
```

- Beispiel:

```
SELECT department_id, MIN(salary), MAX (salary)  
FROM employees  
GROUP BY department_id ;
```

ORACLE

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

Mit der Klausel GROUP BY werden gewählte Zeilen entsprechend den Werten einer oder mehrerer Ausdrücke zu einer Gruppe zusammengefasst. Die Klausel gruppiert Zeilen, garantiert jedoch nicht die Reihenfolge der Ergebnismenge. Um die Reihenfolge der Gruppen festzulegen, geben Sie die Klausel ORDER BY an.

Alle Elemente der SELECT-Liste, die nicht Teil von Aggregatfunktionen sind, müssen in die Elementliste von GROUP BY aufgenommen werden. Dies gilt sowohl für Spalten als auch für Ausdrücke. Die Datenbank gibt für jede Gruppe eine einzelne Zeile mit Aggregatinformationen zurück.

Im Beispiel auf der Folie wird für jede Abteilung aus der Tabelle `employees` das niedrigste und das höchste Gehalt zurückgegeben.

Data Definition Language (DDL)

- Mit DDL-Anweisungen können Sie Schemaobjekte definieren und löschen sowie die Struktur von Schemaobjekten ändern.
- Häufige DDL-Anweisungen:
 - CREATE TABLE, ALTER TABLE und DROP TABLE
 - GRANT, REVOKE
 - TRUNCATE

ORACLE

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

Mit DDL-Anweisungen können Sie die Attribute von Objekten ändern, ohne die Anwendungen zu ändern, die auf das Objekt zugreifen. Darüber hinaus können Sie die Struktur von Objekten ändern, während Benutzer in der Datenbank arbeiten. Häufige Verwendungsbereiche von DDL-Anweisungen:

- Schemaobjekte und andere Datenbankstrukturen (einschließlich der Datenbank selbst und der Datenbankbenutzer) erstellen, ändern und löschen
- Alle Daten in Schemaobjekten löschen, ohne die Struktur der Objekte zu entfernen
- Berechtigungen und Rollen erteilen und entziehen

Oracle Database schreibt die aktuelle Transaktion vor und nach jeder DDL-Anweisung implizit fest.

CREATE TABLE-Anweisungen

- Mit der Anweisung `CREATE TABLE` erstellen Sie Tabellen in der Datenbank.
- Syntax:

```
CREATE TABLE tablename (  
  {column-definition | Table-level constraint}  
  [ , {column-definition | Table-level constraint} ] * )
```

- Beispiel:

```
CREATE TABLE teach_dept (  
  department_id  NUMBER(3) PRIMARY KEY,  
  department_name VARCHAR2(10));
```

ORACLE

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

Mit der Anweisung `CREATE TABLE` erstellen Sie Tabellen in der Datenbank. Um Tabellen erstellen zu können, müssen Sie über die Berechtigung `CREATE TABLE` und einen Speicherbereich verfügen, in dem die Objekte erstellt werden.

Tabellen- und Datenbankeigentümer erhalten nach Erstellung der Tabelle automatisch folgende Berechtigungen:

- `INSERT`
- `SELECT`
- `REFERENCES`
- `ALTER`
- `UPDATE`

Tabellen- und Datenbankeigentümer können diese Berechtigungen auch anderen Benutzern erteilen.

ALTER TABLE-Anweisungen

- Mit der Anweisung `ALTER TABLE` ändern Sie die Definition einer vorhandenen Tabelle in der Datenbank.
- 1. Beispiel:

```
ALTER TABLE teach_dept  
ADD location_id NUMBER NOT NULL;
```

- 2. Beispiel:

```
ALTER TABLE teach_dept  
MODIFY department_name VARCHAR2(30) NOT NULL;
```

ORACLE

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

Mit der Anweisung `ALTER TABLE` können Sie Änderungen an einer vorhandenen Tabelle vornehmen.

Sie können:

- Spalten zu Tabellen hinzufügen
- Constraints zu Tabellen hinzufügen
- vorhandene Spaltendefinitionen ändern
- Spalten aus Tabellen löschen
- vorhandene Constraints aus Tabellen löschen
- die Breite der Spalten `VARCHAR` und `CHAR` vergrößern
- Tabellen in den schreibgeschützten Status setzen

Im 1. Beispiel auf der Folie wird die Tabelle `TEACH_DEPT` um die neue Spalte `LOCATION_ID` erweitert.

Im 2. Beispiel wird die vorhandene Spalte `department_name` von `VARCHAR2(10)` in `VARCHAR2(30)` geändert und das Constraint `NOT NULL` hinzugefügt.

DROP TABLE-Anweisungen

- Mit der Anweisung `DROP TABLE` löschen Sie die Tabelle mit sämtlichen darin enthaltenen Daten aus der Datenbank.
- Beispiel:

```
DROP TABLE teach_dept;
```

- `DROP TABLE` mit der Klausel `PURGE` löscht die Tabelle und gibt den zugewiesenen Speicherplatz frei.

```
DROP TABLE teach_dept PURGE;
```

- Bei Verwendung der Klausel `CASCADE CONSTRAINTS` werden alle Constraints zur referenziellen Integrität aus der Tabelle gelöscht.

```
DROP TABLE teach_dept CASCADE CONSTRAINTS;
```

ORACLE

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

Mit der Anweisung `DROP TABLE` werden Tabellen mitsamt ihrem Inhalt aus der Datenbank gelöscht und in den Papierkorb verschoben. Beim Löschen einer Tabelle werden die abhängigen Objekte invalidiert und die Objektberechtigungen für die Tabelle entfernt.

Um den der Tabelle zugewiesenen Speicherplatz für den Tablespace wieder freizugeben, verwenden Sie die Anweisung `DROP TABLE` mit der Klausel `PURGE`. `DROP TABLE`-Anweisungen mit der Klausel `PURGE` können nicht zurückgerollt werden. Ebenso können Tabellen, die mit der Klausel `PURGE` gelöscht wurden, nicht wiederhergestellt werden.

Mit der Klausel `CASCADE CONSTRAINTS` können Sie die Referenz auf den Primärschlüssel und die eindeutigen Schlüssel in der gelöschten Tabelle aufheben.

GRANT-Anweisungen

- Die Anweisung `GRANT` erteilt Benutzern die Berechtigung zur Ausführung folgender Vorgänge:
 - Daten einfügen und löschen
 - Fremdschlüsselreferenz auf die benannte Tabelle oder eine Teilmenge von Spalten aus der Tabelle erstellen
 - Daten, Views oder eine Teilmenge von Spalten aus der Tabelle wählen
 - Trigger für Tabellen erstellen
 - Angegebene Funktionen oder Prozeduren ausführen
- Beispiel:

```
GRANT SELECT any table to PUBLIC;
```

ORACLE®

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

Mit der Anweisung `GRANT` können Sie:

- bestimmten Benutzern bzw. Rollen oder allen Benutzern Berechtigungen zur Ausführung von Aktionen an Datenbankobjekten erteilen
- Benutzern, `PUBLIC` oder anderen Rollen eine Rolle zuweisen

Prüfen Sie vor dem Absetzen einer `GRANT`-Anweisung, ob für die Autorisierungseigenschaft `derby.database.sql` der Wert `True` eingestellt ist. Mit dieser Eigenschaft wird der SQL-Autorisierungsmodus aktiviert. Wenn Sie der Eigentümer der Datenbank sind, können Sie Berechtigungen für ein Objekt erteilen.

Um allen Benutzern Berechtigungen zu erteilen, verwenden Sie das Schlüsselwort `PUBLIC`. Ist `PUBLIC` angegeben, gelten die Berechtigungen oder Rollen für alle aktuellen und künftigen Benutzer.

Typen von Berechtigungen

Folgende Berechtigungen mit der Anweisung `GRANT` zuweisen:

- `ALL PRIVILEGES`
- `DELETE`
- `INSERT`
- `REFERENCES`
- `SELECT`
- `UPDATE`

ORACLE

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

Oracle Database bietet verschiedene Berechtigungstypen für Benutzer und Rollen:

- Mit dem Berechtigungstyp `ALL PRIVILEGES` erteilen Sie Benutzern und Rollen sämtliche Berechtigungen für die angegebene Tabelle.
- Mit dem Berechtigungstyp `DELETE` erteilen Sie die Berechtigung, Zeilen aus der angegebenen Tabelle zu löschen.
- Mit dem Berechtigungstyp `INSERT` erteilen Sie die Berechtigung, Zeilen in die angegebene Tabelle einzufügen.
- Mit dem Berechtigungstyp `REFERENCES` erteilen Sie die Berechtigung zur Erstellung einer Fremdschlüsselreferenz auf die angegebene Tabelle.
- Mit dem Berechtigungstyp `SELECT` erteilen Sie die Berechtigung zur Ausführung von `SELECT`-Anweisungen für eine Tabelle oder View.
- Mit dem Berechtigungstyp `UPDATE` erteilen Sie die Berechtigung zur Verwendung von `UPDATE`-Anweisungen für die angegebene Tabelle.

REVOKE-Anweisungen

- Mit REVOKE-Anweisungen entziehen Sie Benutzern Berechtigungen zur Ausführung von Aktionen an Datenbankobjekten.
- Benutzern eine *Systemberechtigung* entziehen:

```
REVOKE DROP ANY TABLE  
FROM hr;
```

- Benutzern eine *Rolle* entziehen:

```
REVOKE dw_manager  
FROM sh;
```

ORACLE

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

Mit der Anweisung `REVOKE` entziehen Sie bestimmten oder allen Benutzern oder Rollen die Berechtigungen zur Ausführung von Aktionen an Datenbankobjekten. Folgende Vorgänge sind möglich:

- Benutzern, `PUBLIC` oder anderen Rollen eine Rolle entziehen
- Objekten Berechtigungen entziehen, wenn Sie der Eigentümer des Objekts oder der Datenbank sind

Hinweis: Um Rollen oder Systemberechtigungen entziehen zu können, müssen Sie über eine Berechtigung mit der `ADMIN OPTION` verfügen.

TRUNCATE TABLE-Anweisungen

- Mit TRUNCATE TABLE-Anweisungen löschen Sie alle Zeilen einer Tabelle.
- Beispiel:

```
TRUNCATE TABLE employees_demo;
```

- Oracle Database führt standardmäßig folgende Aufgaben durch:
 - Von gelöschten Zeilen belegten Speicherplatz freigeben
 - Speicherparameter NEXT auf die Größe des letzten Extents einstellen, das durch den Leerungsprozess aus dem Segment entfernt wurde

ORACLE

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

Mit der Anweisung TRUNCATE TABLE löschen Sie alle Zeilen aus einer angegebenen Tabelle. Diese Vorgehensweise ist häufig effizienter, als die Tabelle vollständig zu löschen und neu zu erstellen. Beim Löschen und Neuerstellen von Tabellen gilt:

- Die abhängigen Objekte der Tabelle werden invalidiert.
- Objektberechtigungen müssen neu vergeben werden.
- Indizes, Integritäts-Constraints und Trigger müssen neu erstellt werden.
- Speicherparameter müssen neu festgelegt werden.

Bei Verwendung der Anweisung TRUNCATE TABLE können Sie auf diese Arbeitsschritte verzichten.

Hinweis: TRUNCATE TABLE-Anweisungen können nicht zurückgerollt werden.

Data Manipulation Language (DML)

- Mit DML-Anweisungen können Sie Daten in vorhandenen Schemaobjekten abfragen und bearbeiten.
- DML-Anweisungen werden ausgeführt, wenn Sie:
 - mit der Anweisung `INSERT` neue Zeilen in Tabellen einfügen
 - mit der Anweisung `UPDATE` vorhandene Zeilen in Tabellen ändern
 - mit der Anweisung `DELETE` vorhandene Zeilen aus Tabellen löschen
- Eine *Transaktion* enthält eine Zusammenstellung von DML-Anweisungen, die eine logische Arbeitseinheit bilden.

ORACLE

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

Mit DML-Anweisungen können Sie den Inhalt vorhandener Schemaobjekte abfragen oder ändern. Häufige Verwendungsbereiche von DML-Anweisungen:

- Neue Datenzeilen in Tabellen oder Views einfügen (durch Angabe einer Liste von Spaltenwerten oder durch Auswahl und Bearbeitung vorhandener Daten mithilfe einer Unterabfrage)
- Spaltenwerte in den vorhandenen Zeilen einer Tabelle oder View ändern
- Zeilen aus Tabellen oder Views entfernen

Eine Zusammenstellung von DML-Anweisungen, die eine logische Arbeitseinheit bilden, wird als Transaktion bezeichnet. Im Gegensatz zu DDL-Anweisungen schreiben DML-Anweisungen die aktuelle Transaktion nicht implizit fest.

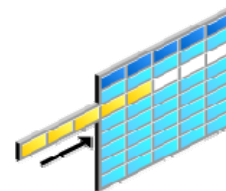
INSERT-Anweisungen

- Mit INSERT-Anweisungen fügen Sie Tabellen neue Zeilen hinzu.
- Syntax:

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- Beispiel:

```
INSERT INTO  departments  
VALUES      (200, 'Development', 104, 1400);  
1 rows inserted.
```



ORACLE

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

Mit der Anweisung `INSERT` fügen Sie einer Tabelle neue Zeilen hinzu. Die neuen Zeilen müssen Werte für die einzelnen Spalten enthalten und die Werte in der für die Spalten der Tabelle gültigen Standardreihenfolge aufgeführt sein. Optional können Sie die Spalten auch in der Anweisung `INSERT` auflisten.

Beispiel:

```
INSERT INTO job_history (employee_id, start_date, end_date,  
                        job_id)  
VALUES (120, '25-JUL-06', '12-FEB-08', 'AC_ACCOUNT');
```

Mit der auf der Folie gezeigten Syntax können Sie jeweils eine einzelne Zeile einfügen. Das Schlüsselwort `VALUES` weist die Werte von Ausdrücken den entsprechenden Spalten in der Spaltenliste zu.

UPDATE-Anweisungen – Syntax

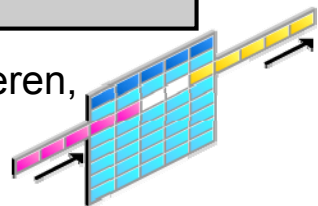
- Mit der Anweisung `UPDATE` ändern Sie die vorhandenen Zeilen in einer Tabelle.
- Falls erforderlich, können mehrere Zeilen gleichzeitig aktualisiert werden.

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- Beispiel:

```
UPDATE      copy_emp
SET
22 rows updated
```

- Um einen Spaltenwert mit `NULL` zu aktualisieren, geben Sie `SET column_name= NULL` an.



ORACLE

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

Mit der Anweisung `UPDATE` ändern Sie die in einer Tabelle vorhandenen Werte. Sie können den Aktualisierungsvorgang prüfen, indem Sie die Tabelle abfragen und die aktualisierten Zeilen anzeigen. Um nur bestimmte Zeilen zu ändern, geben Sie die Klausel `WHERE` an.

Beispiel:

```
UPDATE employees
SET      salary = 17500
WHERE    employee_id = 102;
```

Im Allgemeinen identifizieren Sie die zu aktualisierende Zeile in der Klausel `WHERE` mit der Primärschlüsselspalte. Beispiel: Sie möchten eine bestimmte Zeile in der Tabelle `employees` aktualisieren. In diesem Fall geben Sie die Zeile nicht mit `employee_name`, sondern mit `employee_id` an, da eventuell mehrere Mitarbeiter denselben Namen haben können.

Hinweis: Das Schlüsselwort `condition` setzt sich in der Regel aus Spaltennamen, Ausdrücken, Konstanten, Unterabfragen und Vergleichsoperatoren zusammen.

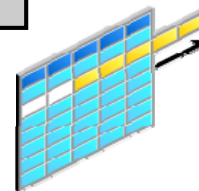
DELETE-Anweisungen

- Mit der Anweisung `DELETE` löschen Sie vorhandene Zeilen aus einer Tabelle.
- Syntax:

```
DELETE    [FROM]    table
[WHERE    condition];
```

- Um bestimmte Zeilen aus einer Tabelle zu löschen, geben Sie die Anweisung `DELETE` mit der Klausel `WHERE` an.

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 rows deleted
```



ORACLE

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

Mit der Anweisung `DELETE` entfernen Sie vorhandene Zeilen aus einer Tabelle. Um bestimmte Zeilen auf Basis einer Bedingung zu löschen, verwenden Sie die Klausel `WHERE`. Die Bedingung (`condition`) gibt die zu löschenden Zeilen an. Sie kann Spaltennamen, Ausdrücke, Konstanten, Unterabfragen und Vergleichsoperatoren enthalten.

Im ersten Beispiel auf der Folie wird die Finanzabteilung aus der Tabelle `departments` gelöscht. Um den Löschvorgang zu prüfen, fragen Sie die Tabelle mit der Anweisung `SELECT` ab.

```
SELECT  *
FROM    departments
WHERE   department_name = 'Finance';
```

Wenn Sie die Klausel `WHERE` weglassen, werden alle Zeilen aus der Tabelle gelöscht. Beispiel:

```
DELETE FROM copy_emp;
```

Im vorstehenden Beispiel werden alle Zeilen aus der Tabelle `COPY_EMP` gelöscht.

Anweisungen zur Transaktionskontrolle

- Mit Anweisungen zur Transaktionskontrolle werden die von DML-Anweisungen vorgenommenen Änderungen verwaltet.
- Die DML-Anweisungen werden zu Transaktionen zusammengefasst.
- Anweisungen zur Transaktionskontrolle:
 - COMMIT
 - ROLLBACK
 - SAVEPOINT

ORACLE

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

Eine Transaktion besteht aus einer Folge von SQL-Anweisungen, die von Oracle Database als Einheit behandelt werden. Anweisungen zur Transaktionskontrolle werden in einer Datenbank verwendet, um die über DML-Anweisungen vorgenommenen Änderungen zu verwalten und die Anweisungen zu Transaktionen zusammenzufassen.

Jeder Transaktion wird eine eindeutige `transaction_id` zugewiesen. Die in der Transaktion enthaltenen SQL-Anweisungen werden entweder alle festgeschrieben (das heißt, auf die Datenbank angewendet) oder alle zurückgerollt (das heißt, in der Datenbank rückgängig gemacht).

COMMIT-Anweisungen

- Mit COMMIT-Anweisungen können Sie:
 - die während der aktuellen Transaktion vorgenommenen Änderungen endgültig speichern
 - alle Savepoints in der Transaktion löschen
 - Transaktionssperren aufheben
- Beispiel:

```
INSERT INTO departments
VALUES      (201, 'Engineering', 106, 1400);
COMMIT;
```

```
1 rows inserted.
committed.
```

ORACLE

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

Wenn Sie die aktuelle Transaktion mit der Anweisung `COMMIT` beenden, werden alle noch nicht gespeicherten Datenänderungen dauerhaft festgeschrieben. Dabei werden alle Zeilen- und Tabellensperren aufgehoben sowie alle seit dem letzten Commit- oder Rollback-Vorgang eventuell festgelegten Savepoints gelöscht. Die mit der Anweisung `COMMIT` vorgenommenen Änderungen sind für alle Benutzer sichtbar.

Oracle empfiehlt, alle in Ihren Anwendungsprogrammen ausgeführten Transaktionen (einschließlich der letzten Transaktion) explizit mit `COMMIT` oder `ROLLBACK` zu beenden, bevor Sie sich bei Oracle Database abmelden. Andernfalls wird im Falle eines Programmabsturzes die letzte nicht festgeschriebene Transaktion automatisch zurückgerollt.

Hinweis: Oracle Database setzt vor und nach jeder DDL-Anweisung implizite `COMMIT`-Anweisungen ab.

ROLLBACK-Anweisungen

- Mit der Anweisung `ROLLBACK` machen Sie die während der aktuellen Transaktion an der Datenbank vorgenommenen Änderungen rückgängig.
- Um nur den Teil der Transaktion nach dem Savepoint rückgängig zu machen, verwenden Sie die Klausel `TO SAVEPOINT`.
- Beispiel:

```
UPDATE      employees
SET         salary = 7000
WHERE       last_name = 'Ernst';
SAVEPOINT   Ernst_sal;

UPDATE      employees
SET         salary = 12000
WHERE       last_name = 'Mourgos';

ROLLBACK TO SAVEPOINT Ernst_sal;
```

ORACLE

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

Mit der Anweisung `ROLLBACK` werden die in der aktuellen Transaktion durchgeführten Aktionen zurückgesetzt. Um die aktuelle Transaktion zurückzurollen, sind keine Berechtigungen erforderlich.

Mit `ROLLBACK` und der Klausel `TO SAVEPOINT` werden die folgenden Vorgänge durchgeführt:

- Nur den nach dem Savepoint ausgeführten Teil der Transaktion zurückrollen
- Alle nach dem angegebenen Savepoint erstellten Savepoints löschen. Der angegebene Savepoint bleibt erhalten, sodass Sie mehrere Rollbacks zu diesem Savepoint vornehmen können.

Mit `ROLLBACK` ohne Angabe der Klausel `TO SAVEPOINT` werden folgende Vorgänge durchgeführt:

- Transaktion beenden
- Alle während der aktuellen Transaktion vorgenommenen Änderungen rückgängig machen
- Alle Savepoints in der Transaktion löschen

SAVEPOINT-Anweisungen

- Mit der Anweisung `SAVEPOINT` benennen und markieren Sie den aktuellen Verarbeitungspunkt einer Transaktion.
- Geben Sie für jeden Savepoint einen Namen an.
- Vergeben Sie innerhalb einer Transaktion eindeutige Savepoint-Namen, um Überschreibungen zu vermeiden.
- Syntax:

```
SAVEPOINT savepoint;
```

ORACLE

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

Mit der Anweisung `SAVEPOINT` kennzeichnen Sie einen Punkt in der Transaktion, bis zu dem Sie die Transaktion im Falle eines Rollbacks zurücksetzen können. Jeder Savepoint muss über einen eindeutigen Namen verfügen. Wenn Sie einen zweiten Savepoint mit demselben Namen erstellen, wird der frühere Savepoint gelöscht.

Nach der Erstellung eines Savepoints können Sie entweder mit der Verarbeitung fortfahren, Ihre Arbeit festschreiben, die gesamte Transaktion zurückrollen oder alle nach dem Savepoint ausgeführten Aktionen zurückrollen.

Bei einem einfachen Rollback- oder Commit-Vorgang werden alle Savepoints gelöscht. Bei einem Rollback bis zu einem Savepoint werden alle späteren Savepoints gelöscht. Der Savepoint, bis zu dem die Transaktion zurückgerollt wurde, bleibt erhalten.

Wenn Savepoint-Namen innerhalb einer Transaktion mehrfach verwendet werden, verschiebt Oracle Database den Savepoint von der bisherigen an die aktuelle Position in der Transaktion und setzt damit den älteren Savepoint außer Kraft.

Joins

Mit Joins fragen Sie Daten aus mehreren Tabellen ab:

```
SELECT  table1.column, table2.column
FROM    table1, table2
WHERE   table1.column1 = table2.column2;
```

- Join-Bedingungen werden in der Klausel `WHERE` angegeben.
- Setzen Sie den Tabellennamen vor den Spaltennamen, wenn derselbe Spaltenname in mehreren Tabellen vorkommt.

ORACLE

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

Um Daten aus mehreren Tabellen einer Datenbank abzurufen, werden *Join*-Bedingungen verwendet. Zeilen aus zwei Tabellen werden in Abhängigkeit von gemeinsamen Werten verknüpft, die in den entsprechenden Spalten (in der Regel Primär- und Fremdschlüsselspalten) vorhanden sind.

Um Daten aus zwei oder mehr verknüpften Tabellen anzuzeigen, geben Sie in der Klausel `WHERE` eine einfache Join-Bedingung an.

Für die Syntax gilt:

<code>table1.column</code>	Tabelle und Spalte, aus der Daten abgerufen werden
<code>table1.column1 =</code>	Bedingung für die Verknüpfung (oder Relation) der Tabellen
<code>table2.column2</code>	

Richtlinien

- Um den Code übersichtlicher zu gestalten und den Datenbankzugriff zu vereinfachen, sollten Sie in `SELECT`-Anweisungen, die Tabellen verknüpfen, den Spaltennamen Tabellennamen voranstellen.
- Kommt derselbe Spaltenname in mehreren Tabellen vor, ist der Tabellenname als Präfix obligatorisch.
- Um n Tabellen miteinander zu verknüpfen, sind mindestens $n-1$ Join-Bedingungen erforderlich. Beispiel: Um vier Tabellen zu verknüpfen, werden mindestens drei Joins benötigt. In Tabellen mit verkettetem Primärschlüssel trifft diese Regel möglicherweise nicht zu. In diesem Fall werden mehrere Spalten benötigt, um die einzelnen Zeilen eindeutig zu identifizieren.

Typen von Joins

- Natural Joins
- Equi Joins
- Non-Equi Joins
- Outer Joins
- Self Joins
- Cross Joins

ORACLE®

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

Mit der Join-Syntax von Oracle können Sie Tabellen verknüpfen.

Hinweis: In Datenbankreleases bis Oracle 9i galt für Joins eine Oracle-spezifische Syntax. Die SQL:1999-konforme Join-Syntax bietet gegenüber der Oracle-spezifischen Join-Syntax keine Performancevorteile.

Mehrdeutige Spaltennamen eindeutig kennzeichnen

- Spaltennamen, die in mehreren Tabellen vorkommen, müssen durch ein Tabellenpräfix eindeutig gekennzeichnet werden.
- Tabellenpräfixe verbessern die Performance.
- Anstelle von vollständigen Tabellennamen können Sie Tabellenaliasnamen verwenden.
- Tabellenaliasnamen verkürzen den Tabellennamen:
 - Kürzerer SQL-Code, weniger Speicherbedarf
- Spaltenaliasnamen dienen der Unterscheidung von Spalten mit identischen Namen, die in verschiedenen Tabellen vorkommen.

ORACLE

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

Wenn Sie zwei oder mehr Tabellen miteinander verknüpfen, müssen die Spaltennamen durch den Tabellennamen gekennzeichnet werden, um Mehrdeutigkeit zu vermeiden. Ohne das Tabellenpräfix könnte sich der Spaltenname `DEPARTMENT_ID` in der `SELECT`-Liste auf die entsprechende Spalte in der Tabelle `DEPARTMENTS` oder in der Tabelle `EMPLOYEES` beziehen. Daher müssen Sie vor Ausführung der Abfrage das Tabellenpräfix hinzufügen. Gibt es in den Tabellen keine identischen Spaltennamen, besteht keine Notwendigkeit, die Spalten durch Angabe des Tabellennamens eindeutig zu kennzeichnen. Tabellenpräfixe verbessern jedoch die Performance, weil sie dem Oracle-Server genau angeben, wo sich die Spalten befinden.

Die eindeutige Kennzeichnung von Spaltennamen durch Tabellennamen kann insbesondere bei langen Tabellennamen zeitaufwendig sein. Sie können daher anstelle von Tabellennamen einen kürzeren *Tabellenalias* verwenden (analog zu Aliasnamen für Spalten). Tabellenaliasnamen verkürzen den SQL-Code und sind daher weniger speicherintensiv.

Der Tabellenalias folgt auf den vollständig angegebenen Tabellennamen, getrennt durch ein Leerzeichen. Beispielsweise kann die Tabelle `EMPLOYEES` den Alias `e` und die Tabelle `DEPARTMENTS` den Alias `d` erhalten.

Richtlinien

- Tabellenaliasnamen können bis zu 30 Zeichen lang sein, kürzere Aliasnamen sind jedoch besser.
- Tabellenaliasnamen, die in der Klausel `FROM` bestimmte Tabellennamen ersetzen, müssen in der gesamten `SELECT`-Anweisung anstelle des Tabellennamens verwendet werden.
- Wählen Sie möglichst aussagekräftige Tabellenaliasnamen.
- Tabellenaliasnamen gelten nur für die aktuelle `SELECT`-Anweisung.

Natural Joins

- Die Klausel `NATURAL JOIN` verbindet zwei Tabellen auf der Basis von allen Spalten, die in beiden Tabellen denselben Namen haben.
- Die Klausel wählt die Zeilen, deren Spalten in den Tabellen identische Namen und Datenwerte aufweisen.
- Beispiel:

```
SELECT country_id, location_id, country_name, city  
FROM countries NATURAL JOIN locations;
```

	COUNTRY_ID	LOCATION_ID	COUNTRY_NAME	CITY
1	US	1400	United States of America	Southlake
2	US	1500	United States of America	South San Francisco
3	US	1700	United States of America	Seattle
4	CA	1800	Canada	Toronto
5	UK	2500	United Kingdom	Oxford

ORACLE

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

Sie können Tabellen automatisch verknüpfen, deren Spalten in beiden Tabellen denselben Namen und Datentyp aufweisen. Hierzu verwenden Sie die Schlüsselwörter `NATURAL JOIN`.

Hinweis: Joins sind nur für Spalten möglich, die in beiden Tabellen denselben Namen und denselben Datentyp aufweisen. Haben die Spalten denselben Namen, jedoch unterschiedliche Datentypen, verursacht die Syntax `NATURAL JOIN` einen Fehler.

Im Beispiel auf der Folie werden die Tabellen `COUNTRIES` und `LOCATIONS` über die Spalte `COUNTRY_ID` verknüpft, da dies der einzige Spaltenname ist, der in beiden Tabellen vorkommt. Wären weitere übereinstimmende Spalten vorhanden, würde der Join alle diese Spalten berücksichtigen.

Equi Joins

EMPLOYEES

	EMPLOYEE_ID	DEPARTMENT_ID
1	200	10
2	201	20
3	202	20
4	205	110
5	206	110
6	100	90
7	101	90
8	102	90
9	103	60
10	104	60
...		

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

Fremdschlüssel

Primärschlüssel

ORACLE

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

Ein **Equi Join** ist ein Join mit einer Join-Bedingung, die einen Gleichheitsoperator enthält. Equi Joins verbinden Zeilen, die in den angegebenen Spalten äquivalente Werte enthalten. Um die Abteilung eines Mitarbeiters zu ermitteln, vergleichen Sie die Werte der Spalte `DEPARTMENT_ID` in der Tabelle `EMPLOYEES` mit den Werten der Spalte `DEPARTMENT_ID` in der Tabelle `DEPARTMENTS`. Die Beziehung zwischen den Tabellen `EMPLOYEES` und `DEPARTMENTS` ist ein *Equi Join*, das heißt, die Werte der Spalte `DEPARTMENT_ID` müssen in beiden Tabellen gleich sein. Häufig sind bei dieser Art von Join sich ergänzende Primärschlüssel und Fremdschlüssel beteiligt.

Hinweis: Equi Joins werden auch als *einfache Joins* bezeichnet.

Datensätze mit Equi Joins abrufen

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     e.department_id = d.department_id;
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	144	Vargas	50	50	1500
5	143	Matos	50	50	1500
6	142	Davies	50	50	1500
7	141	Rajs	50	50	1500
8	124	Mourgos	50	50	1500
9	103	Hunold	60	60	1400
10	104	Ernst	60	60	1400
11	107	Lorentz	60	60	1400

...

ORACLE

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

Im Beispiel auf der Folie gilt:

- **Die Klausel SELECT gibt die abzurufenden Spaltennamen an:**
 - Nachname, Personalnummer und Abteilungsnummer des Mitarbeiters aus den entsprechenden Spalten der Tabelle EMPLOYEES
 - Abteilungs-ID und Standort-ID aus den entsprechenden Spalten der Tabelle DEPARTMENTS
- **Die Klausel FROM gibt die beiden Tabellen an, auf die die Datenbank zugreifen muss:**
 - EMPLOYEES
 - DEPARTMENTS
- **Die Klausel WHERE gibt an, wie die Tabellen verknüpft werden sollen:**
`e.department_id = d.department_id`

Da die Spalte DEPARTMENT_ID in beiden Tabellen enthalten ist, muss dem Spaltennamen der Tabellenalias vorangestellt werden, um Mehrdeutigkeit zu vermeiden. Eine Kennzeichnung anderer Spalten, die nicht in beiden Tabellen vorkommen, durch einen Tabellenalias ist nicht erforderlich, empfiehlt sich jedoch aus Performancegründen.

Hinweis: Wenn Sie die Abfrage mit dem Symbol **Run Statement** ausführen, fügt SQL Developer zur Unterscheidung der beiden DEPARTMENT_IDS das Suffix "_1" an.

Zusätzliche Suchbedingungen mit den Operatoren AND und WHERE

```
SELECT d.department_id, d.department_name, l.city
FROM departments d JOIN locations l
ON d.location_id = l.location_id
AND d.department_id IN (20, 50);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	CITY
1	20	Marketing	Toronto
2	50	Shipping	South San Francisco

```
SELECT d.department_id, d.department_name, l.city
FROM departments d JOIN locations l
ON d.location_id = l.location_id
WHERE d.department_id IN (20, 50);
```

ORACLE

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

Neben dem Join können Sie auch Kriterien für die Klausel `WHERE` angeben, um die für eine oder mehrere Tabellen im Join berücksichtigten Zeilen einzuschränken. Im Beispiel auf der Folie werden die Tabellen `DEPARTMENTS` und `LOCATIONS` verknüpft und darüber hinaus nur die Abteilungen mit der ID 20 oder 50 angezeigt. Um zusätzliche Bedingungen für die Klausel `ON` aufzunehmen, können Sie `AND`-Klauseln hinzufügen. Alternativ können Sie zusätzliche Bedingungen über eine `WHERE`-Klausel anwenden.

Beide Abfragen führen zum selben Ergebnis.

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Matos	50	Shipping

Datensätze mit Non-Equi Joins abrufen

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
      BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Fay	6000	C

...

ORACLE

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

Im Beispiel auf der Folie wird ein Non-Equi Join erstellt, um die Gehaltsstufe eines Mitarbeiters auszuwerten. Das Gehalt muss *zwischen* dem Mindest- und Höchstwert für die einzelnen Gehaltsbereiche liegen.

Beachten Sie, dass jeder Mitarbeiter bei der Ausführung dieser Abfrage genau einmal angezeigt wird. Kein Mitarbeiter wird in der Liste mehrfach angezeigt. Dafür gibt es zwei Gründe:

- Keine der Zeilen in der Tabelle `job_grades` enthält Gehaltsstufen, die sich überschneiden. Das bedeutet, der Gehaltswert für einen Mitarbeiter kann nur zwischen dem Mindest- und Höchstwert aus einer der Zeilen der Tabelle `job_grades` liegen.
- Alle Mitarbeitergehälter liegen innerhalb der durch die Tabelle `job_grades` festgelegten Grenzwerte. Kein Mitarbeiter verdient also weniger als das niedrigste Gehalt aus der Spalte `LOWEST_SAL` oder mehr als das höchste Gehalt aus der Spalte `HIGHEST_SAL`.

Hinweis: Es können auch andere Bedingungen verwendet werden, z. B. `<=` und `>=`. `BETWEEN` ist jedoch die einfachste Möglichkeit. Geben Sie zuerst den Mindest- und dann den Höchstwert an, wenn Sie den Operator `BETWEEN` verwenden. Der Oracle-Server übersetzt den Operator `BETWEEN` in ein `AND`-Bedingungs paar. Daher bietet `BETWEEN` keine Performancevorteile und dient nur der logischen Einfachheit.

Die Tabellenaliasnamen wurden im Beispiel auf der Folie aus Performancegründen verwendet, nicht wegen möglicher Mehrdeutigkeit.

Datensätze mit USING-Klauseln abrufen

- Um aus mehreren übereinstimmenden Spalten nur eine Spalte abzurufen, verwenden Sie die Klausel `USING`.
- Diese Klausel kann nicht mit einem `NATURAL`-Join angegeben werden.
- Kennzeichnen Sie den Spaltennamen nicht mit einem Tabellennamen oder -alias.
- Beispiel:

```
SELECT country_id, country_name, location_id, city
FROM countries JOIN locations
USING (country_id) ;
```

	COUNTRY_ID	COUNTRY_NAME	LOCATION_ID	CITY
1	US	United States of America	1400	Southlake
2	US	United States of America	1500	South San Francisco
3	US	United States of America	1700	Seattle
4	CA	Canada	1800	Toronto
5	UK	United Kingdom	2500	Oxford

ORACLE

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

Im Beispiel auf der Folie werden die `COUNTRY_ID`-Spalten aus den Tabellen `COUNTRIES` und `LOCATIONS` verknüpft. Damit wird die `LOCATION_ID` für den Standort angezeigt, an dem ein Mitarbeiter arbeitet.

Datensätze mit ON-Klauseln abrufen

- Die Join-Bedingung für Natural Joins ist im Prinzip ein Equi Join aller Spalten mit gleichem Namen.
- Mit der Klausel ON können Sie beliebige Bedingungen oder zu verknüpfende Spalten angeben.
- Die Klausel ON macht den Code verständlicher.

```
SELECT e.employee_id, e.last_name, j.department_id,  
FROM   employees e JOIN job_history j  
ON     (e.employee_id = j.employee_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	101	Kochhar	110
2	101	Kochhar	110
3	102	De Haan	60
4	176	Taylor	80
5	176	Taylor	80
6	200	Whalen	90
7	200	Whalen	90
8	201	Hartstein	20

ORACLE

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

Mit der Klausel ON können Sie Join-Bedingungen getrennt von Such- oder Filterkriterien in der Klausel WHERE angeben.

In diesem Beispiel werden die EMPLOYEE_ID-Spalten aus den Tabellen EMPLOYEES und JOB_HISTORY mit der Klausel ON verknüpft. Immer wenn eine Personalnummer in der Tabelle EMPLOYEES einer Personalnummer in der Tabelle JOB_HISTORY entspricht, wird die Zeile zurückgegeben. Der Tabellenalias ist erforderlich, um die übereinstimmenden Spaltennamen zu kennzeichnen.

Mit der Klausel ON können Sie auch Spalten mit unterschiedlichen Namen verknüpfen. Die Klammern um die verknüpften Spalten im Beispiel auf der Folie (e.employee_id = j.employee_id) sind optional. Entsprechend wäre auch ON e.employee_id = j.employee_id korrekt.

Hinweis: Wenn Sie die Abfrage mit dem Symbol **Run Statement** ausführen, fügt SQL Developer zur Unterscheidung der beiden employee_ids das Suffix "_1" an.

Left Outer Joins

- Ein Join zwischen zwei Tabellen, der alle übereinstimmenden Zeilen sowie die nicht übereinstimmenden Zeilen aus der linken Tabelle zurückgibt, wird als **LEFT OUTER JOIN** bezeichnet.
- Beispiel:

```
SELECT c.country_id, c.country_name, l.location_id, l.city
FROM   countries c LEFT OUTER JOIN locations l
ON     (c.country_id = l.country_id) ;
```

	COUNTRY_ID	COUNTRY_NAME	LOCATION_ID	CITY
1	CA	Canada	1800	Toronto
2	DE	Germany	(null)	(null)
3	UK	United Kingdom	2500	Oxford
4	US	United States of America	1400	Southlake
5	US	United States of America	1500	South San Francisco
6	US	United States of America	1700	Seattle

ORACLE

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

Diese Abfrage ruft alle Zeilen aus der Tabelle **COUNTRIES** (linke Tabelle) ab, auch wenn keine Übereinstimmung in der Tabelle **LOCATIONS** gegeben ist.

Right Outer Joins

- Ein Join zwischen zwei Tabellen, der alle übereinstimmenden Zeilen sowie die nicht übereinstimmenden Zeilen aus der rechten Tabelle zurückgibt, wird als `RIGHT OUTER JOIN` bezeichnet.
- Beispiel:

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

ORACLE

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

Diese Abfrage ruft alle Zeilen aus der Tabelle `DEPARTMENTS` (rechte Tabelle) ab, auch wenn keine Übereinstimmung in der Tabelle `EMPLOYEES` gefunden wird.

Full Outer Joins

- Ein Join zwischen zwei Tabellen, der neben allen übereinstimmenden Zeilen auch die nicht übereinstimmenden Zeilen aus beiden Tabellen zurückgibt, wird als **FULL OUTER JOIN** bezeichnet.
- Beispiel:

```
SELECT e.last_name, d.department_id, d.manager_id,  
       d.department_name  
FROM   employees e FULL OUTER JOIN departments d  
ON     (e.manager_id = d.manager_id) ;
```

	1 LAST_NAME	2 DEPARTMENT_ID	3 MANAGER_ID	4 DEPARTMENT_NAME
1	King	(null)	(null)	(null)
2	Kochhar	90	100	Executive
3	De Haan	90	100	Executive
4	Hunold	(null)	(null)	(null)

...

19	Higgins	(null)	(null)	(null)
20	Gietz	110	205	Accounting
21	(null)	190	(null)	Contracting
22	(null)	10	200	Administration

ORACLE

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

Diese Abfrage ruft alle Zeilen aus der Tabelle `EMPLOYEES` ab, auch wenn in der Tabelle `DEPARTMENTS` keine Übereinstimmung gefunden wird. Außerdem werden alle Zeilen aus der Tabelle `DEPARTMENTS` abgerufen, unabhängig davon, ob Übereinstimmungen in der Tabelle `EMPLOYEES` vorhanden sind.

Self Joins – Beispiel

```
SELECT worker.last_name || ' works for '
       || manager.last_name
FROM   employees worker JOIN employees manager
ON worker.manager_id = manager.employee_id
ORDER BY worker.last_name;
```

	WORKER.LAST_NAME 'WORKS FOR' MANAGER.LAST_NAME
1	Abel works for Zlotkey
2	Davies works for Mourgos
3	De Haan works for King
4	Ernst works for Hunold
5	Fay works for Hartstein
6	Gietz works for Higgins
7	Grant works for Zlotkey
8	Hartstein works for King
9	Higgins works for Kochhar

...

ORACLE

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

In manchen Fällen müssen Sie eine Tabelle mit sich selbst verknüpfen. Um für jeden Mitarbeiter den Namen des zuständigen Managers zu ermitteln, verknüpfen Sie die Tabelle `EMPLOYEES` mit sich selbst und führen damit einen Self Join durch. Im Beispiel auf der Folie wird die Tabelle `EMPLOYEES` mit sich selbst verknüpft. Um in der Klausel `FROM` zwei Tabellen zu simulieren, werden für dieselbe Tabelle (`EMPLOYEES`) zwei Aliasnamen verwendet: `worker` und `manager`.

In diesem Beispiel enthält die Klausel `WHERE` den Join, der wie folgt interpretiert wird: "wobei die Managernummer eines Mitarbeiters der Personalnummer des Managers entspricht".

Cross Joins

- Ein CROSS JOIN ist ein JOIN-Vorgang, der das kartesische Produkt aus zwei Tabellen zurückgibt.
- Beispiel:

```
SELECT department_name, city  
FROM department CROSS JOIN location;
```

	DEPARTMENT_NAME	CITY
1	Administration	Oxford
2	Administration	Seattle
3	Administration	South San Francisco
4	Administration	Southlake
5	Administration	Toronto
6	Marketing	Oxford
7	Marketing	Seattle
8	Marketing	South San Francisco
9	Marketing	Southlake
10	Marketing	Toronto

...

ORACLE

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

Die Syntax für CROSS JOIN gibt das Kreuzprodukt an, das auch als kartesisches Produkt bezeichnet wird. Ein Cross Join gibt das Kreuzprodukt aus zwei Relationen zurück und entspricht im Prinzip einer durch Kommas getrennten Oracle Database-Notation.

Zwischen den beiden Tabellen im CROSS JOIN geben Sie keine WHERE-Bedingungen an.

Zusammenfassung

In diesem Anhang haben Sie Folgendes gelernt:

- Mit `SELECT`-Anweisungen Zeilen aus einzelnen oder mehreren Tabellen abrufen
- Mit DDL-Anweisungen die Struktur von Objekten ändern
- Mit DML-Anweisungen Daten in vorhandenen Schemaobjekten abfragen oder bearbeiten
- Mit Anweisungen zur Transaktionskontrolle die über DML-Anweisungen vorgenommenen Änderungen verwalten
- Mithilfe von Joins Daten aus mehreren Tabellen anzeigen

ORACLE

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

Zu den zahlreichen häufig verwendeten SQL-Anweisungen und -Befehlen gehören DDL-, DML- und Transaktionskontrollanweisungen sowie Joins.

