

Einführung

Abfragesprache SQL in ORACLE

Literatur

Oracle Online-Dokumentation – Database Administration

http://docs.oracle.com/database/121/nav/portal_4.htm

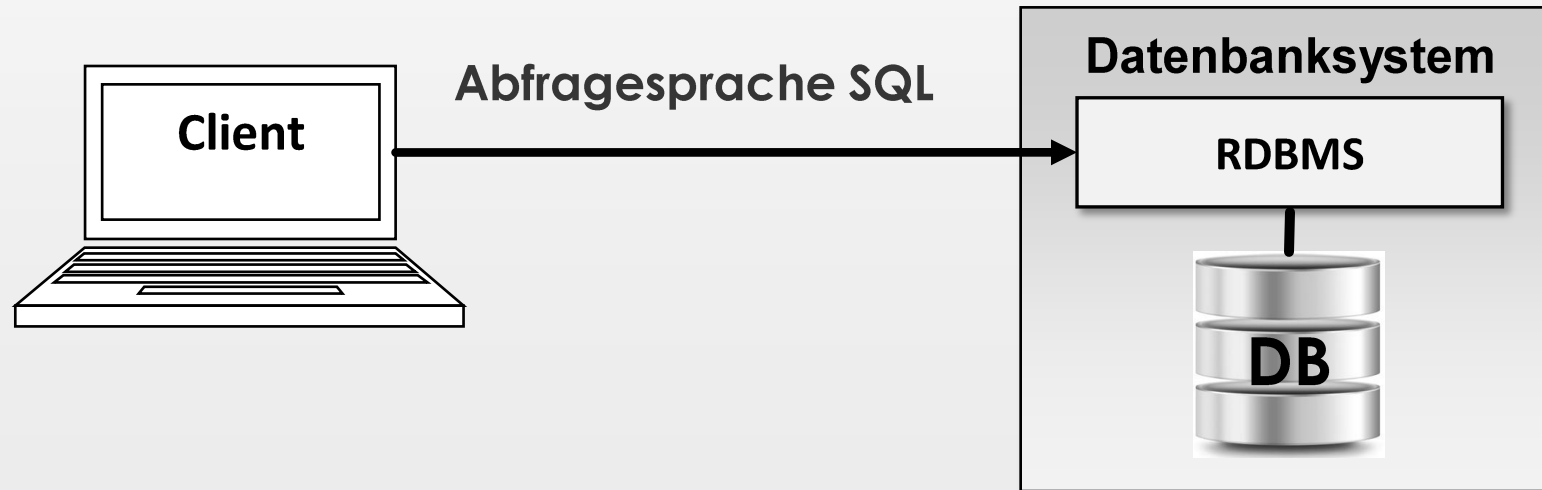
- [Database SQL Language Reference](#)
- [Database SQL Language Quick Reference](#)

AGENDA

- 1 Structured Query Language (SQL)**
- 2 DDL und DML Grundlagen**
 - 2.1 Tabellen anlegen, ändern, löschen
 - 2.2 Daten anzeigen, einfügen, ändern und löschen

1. Structured Query Language

- **SQL - Structured Query Language**
- **Standardisierte** Abfragesprache
- ermöglicht Zugriff auf Daten in **relationalen** Datenbanksystemen
- Trotz Standardisierung haben die meisten RDBMS **eigenen „SQL-Dialekt“**, der geringfügig vom Standard abweicht



SQL-Befehle lassen sich in zwei Kategorien einteilen

➤ **Data Definition Language (DDL)**

- Diese Befehle definieren Daten
- Darunter fällt das **Anlegen, Löschen, Ändern von Tabellen**
- die **Vergabe von Berechtigungen**
z.B.: Welcher User darf auf welche Tabelle zugreifen

➤ **Data Manipulation Language (DML)**

- Diese Befehle manipulieren Daten
- Darunter fällt
 - **Einfügen, Löschen, Ändern von Daten** in den Tabellen

2. DDL Grundlagen

Tabellen anlegen, ändern und löschen

- In einer Tabelle wird jeder Spalte (Attribut) ein eigener Datentyp zugeordnet
- Trotz Datentypen-Standard unterscheiden sich Datentypen von RDBMS zu RDBMS.

Wichtigste Datentypen in Oracle

- VARCHAR2(n) : Text der Länge n.
- NUMBER(n,p) : Dezimalzahl mit insgesamt n Stellen, davon p Nachkommastellen
- DATE : Datum inklusive Uhrzeit (Stunde, Minute, Sekunde)

Tabellen anlegen

SYNTAX

```
CREATE TABLE tabellenname
(
    spaltenname datentyp (NOT NULL)
    , spaltenname datentyp (NOT NULL)
    , spaltenname datentyp (NOT NULL)
    ...
);
```

BEISPIEL

```
CREATE TABLE person
(
    nachname VARCHAR2(30) NOT NULL,
    geburtstag DATE,
    gewicht NUMBER(4,1)
);
```

Nachname	Geburtstag	Gewicht

- Es wird eine Tabelle PERSON angelegt mit den Spalten Nachname, Geburtstag und Gewicht
- **NOT NULL:**
Beim Einfügen von Daten in die Tabelle muss bei Nachname ein Wert angegeben werden, bei Geburtstag und Gewicht nicht

Tabellen anlegen – CREATE TABLE

BEISPIEL

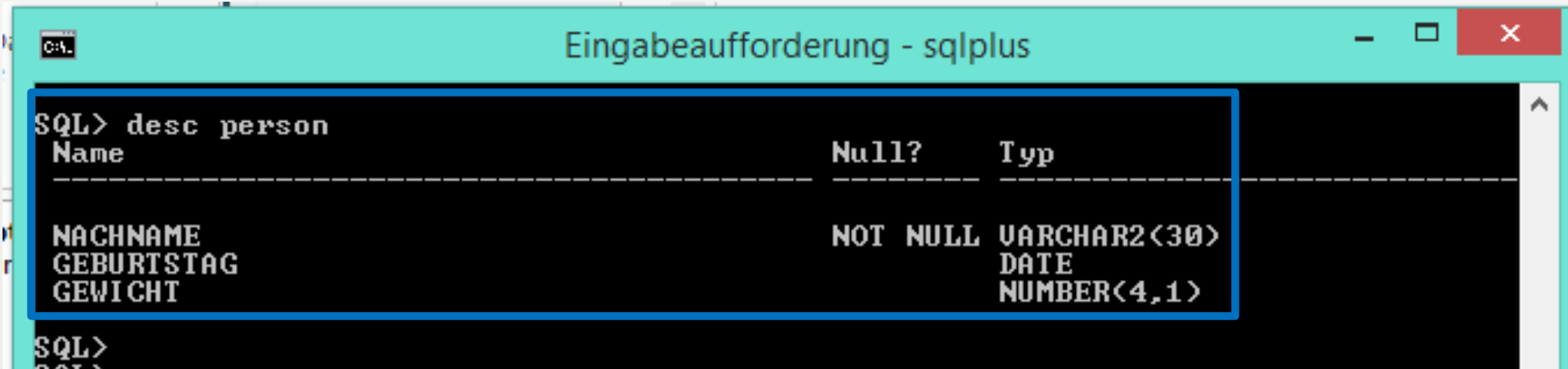
```
CREATE TABLE person
(
    nachname    VARCHAR2(30) NOT NULL,
    geburstag DATE,
    gewicht    NUMBER(4,1)
);
```

HINWEIS

- Tabelle wird automatisch **im Schema des angemeldeten Users** gespeichert
- Andere DB-Nutzer können die Tabelle mit **schemaname.person** ansprechen, wenn sie die Berechtigung dazu haben
- Schemaname entspricht dabei dem Usernamen

SQLPlus DESCRIBE

- Um den Aufbau der angelegten Tabelle abzufragen, kann der SQLPlus-Befehl **DESCRIBE (DESC)** verwendet werden.
- Durch den Befehl DESC person wird z.B. der Aufbau der angelegten Tabelle Person angezeigt



```
SQL> desc person
Name                               Null?    Typ
-----
NACHNAME                           NOT NULL VARCHAR2(30)
GEBURTSTAG                          DATE
GEWICHT                             NUMBER(4,1)

SQL>
```

Attribut (Spalte) hinzufügen

```
ALTER TABLE tabellenname  
ADD (spaltenname datentyp,  
      spaltenname datentyp,...) ;
```

```
ALTER TABLE person  
ADD (anz_autos NUMBER(1)) ;
```

- Der Tabelle person wird eine Spalte anz_autos hinzugefügt mit dem Datentyp Number (ganzzahlig, einstellig)

Datentyp eines Attributs ändern

```
ALTER TABLE tabellenname  
MODIFY (spaltenname datentyp,  
        spaltenname datentyp,...) ;
```

```
ALTER TABLE person  
MODIFY (anz_autos NUMBER(2)) ;
```

- Bei der Tabelle person wird die Spalte pnr für anz_autos auf NUMBER(2) abgeändert

Attribute löschen

```
ALTER TABLE tabellenname  
DROP (spaltenname,  
      spaltenname,...);
```

```
ALTER TABLE person  
DROP (anz_autos);
```

- Bei der Tabelle person wird die Spalte anz_autos gelöscht.

Tabelle umbenennen

```
ALTER TABLE tabellenname_alt  
RENAME TO tabellenname_neu;
```

```
ALTER TABLE person  
RENAME TO mitarbeiter;
```

- Tabelle person wird in mitarbeiter umbenannt

Tabelle löschen

Tabelle löschen

```
DROP TABLE tabellenname  
(purge);
```

```
DROP TABLE mitarbeiter purge;
```

- Wird **purge** beim Löschen nicht angegeben, wandert die Tabelle in einen **Papierkorb** (Recycle Bin).
- Aus dem Recycle Bin kann sie ggf. **wiederhergestellt** werden
- Bei **Angabe von PURGE** wird die Tabelle **vollständig gelöscht** und befindet sich nicht mehr im Recycle Bin

1. Lege eine Tabelle kaeufer an mit den Attributen Kundennr, Vorname, Nachname, Erstkaufdatum und Kundenwert (Betrag in Euro). Entscheide selbst, welche Datentypen für die Attribute verwendet werden sollten.
Stelle sicher, dass Kundennummer, Vorname und Nachname beim Einfügen in die Tabelle immer angegeben werden müssen.
2. Füge der Tabelle eine Spalte anz_kinder hinzu.
3. Lösche die Spalte Kundenwert
4. Ändere den Datentyp des Vornamen auf VARCHAR2(100)
5. Ändere den Namen der Tabelle in **kunde**

2. DML-Grundlagen

Einfügen von Datensätzen

Einfügen eines Datensatzes in eine Tabelle

```
INSERT INTO tabellenname  
(spaltennameA, spaltennameB, ..)  
values (wertA, wertB)
```

```
INSERT INTO person (nachname, gewicht)  
values ('Mustermann', 72.5);
```

- In die Tabelle person wird ein Datensatz (eine Zeile) eingefügt, in der nur Nachname und Gewicht angegeben werden

Nachname	Geburtstag	Gewicht
Mustermann	NULL	72.5

- Ein fehlende Wert (wie hier bei Geburtstag) wird durch den Platzhalter „NULL“ repräsentiert.
- Attributwerte, deren Attribute beim Anlegen der Tabelle als NOT NULL spezifiziert wurden, müssen immer angegeben werden
→ Sonst Fehlermeldung: Einfügen von NULL nicht möglich

Einfügen von Datensätzen

Einfügen eines Datensatzes in eine Tabelle

```
INSERT INTO tabellenname  
values (wertA, wertB, wertC);
```

```
INSERT INTO person  
values ('Mustermann', '01.01.1980', 70.5 );
```

- Werden hinter dem Tabellennamen keine Spaltennamen angegeben, muss in VALUES (...) für jede Spalte der Tabelle ein Wert festgelegt werden.
- Wird in VALUES (..) nicht für jede Tabellenspalte ein Wert angegeben, führt das zu einer Fehlermeldung → Anzahl der Werte reicht nicht aus

Einfügen von Datumswerten

Ein Datum kann beim Einfügen in einfachen Hochkomma angegeben werden.

```
INSERT INTO person  
values ('Mustermann', '01.01.1980', 70.5);
```

Ob Datums-Format richtig erkannt wird, hängt jedoch von **Formateinstellungen der aktiven Session** ab.

Exkurs: Session und Parameter

- Meldet sich ein Benutzer an der DB an, wird eine so genannte Session gestartet. Diese endet, wenn der Benutzer sich von der Datenbank abmeldet.
- Für jede Session, können bestimmte Parameter (z.B. für Formateinstellungen von Datum und Zahlen) festgelegt werden.

Beispiel Änderung Datumsformat:

```
ALTER SESSION SET NLS_DATE_FORMAT='YYYY.MM.DD';
```

Einfügen von Datumswerten

Um sicherzugehen, dass ein Datum unabhängig von den Formateinstellungen der Session immer richtig erkannt wird, sollte die Funktion **to_date** verwendet werden.

```
INSERT INTO person  
values ('Mustermann', to_date('01.01.1980', 'dd.mm.yyyy'), 70.5);
```

Aufbau der Funktion to_date('datum', 'format')

- An erster Stelle wird das Datum angegeben
- An zweiter Stelle wird das Format festgelegt, in dem das Datum vorliegt:
dd.mm.yyyy entspricht dem Format des Datums 01.01.1980
- Weitere Formatmodelle: siehe [Oracle-Doku](#)

Filtern und Anzeigen der Daten

Alle Spalten (Attribute) und Zeilen (Datensätze) einer Tabelle anzeigen

```
SELECT *  
FROM tabellenname;
```

```
SELECT *  
FROM person;
```

- Alle Attribute (* = alle) und alle Datensätze der Tabelle Person anzeigen

Ausgewählte Spalten einer Tabelle anzeigen

```
SELECT spaltenname, spaltenname...  
FROM tabellenname;
```

```
SELECT nachname, geburtstag  
FROM person;
```

- Für alle Datensätze in der Tabelle Person Vorname und Nachname anzeigen

Filtern und Anzeigen der Daten

Nur Datensätze der Tabelle anzeigen, die bestimmte Bedingungen erfüllen

```
SELECT *  
FROM tabellenname  
WHERE condition;
```

Nur Datensätze mit bestimmten Attributwerten anzeigen

```
SELECT *  
FROM person  
WHERE nachname = 'Mustermann';
```

```
SELECT *  
FROM person  
WHERE gewicht = 70.5;
```

- Alle Datensätze, bei denen Nachname Mustermann ist
(Achtung: case-sensitive)
- Alle Datensätze, bei denen Gewicht = 70.5 ist

Filtern und Anzeigen der Daten

Datensätze anzeigen, bei denen Attributwert größer/kleiner/ungleich einem Wert ist

```
SELECT *  
FROM person  
WHERE gewicht > 60;
```

- Alle Datensätze, bei denen das Gewicht größer 60 ist

Weiter Vergleiche:

größer (gleich)/kleiner (gleich): <, >, <=, >=
ungleich: !=

Datensätze anzeigen, bei denen Attributwert in Liste von Werten ist

```
SELECT *  
FROM person  
WHERE gewicht IN(70.5, 80.5, 90.5);
```

- Alle Datensätze, bei denen Gewicht 70.5, 80.5 oder 90.5 ist

LIKE und Wildcards

Ein Wert kann mit dem LIKE-Operator auch auf ein **bestimmtes Muster** überprüft werden.

Dazu werden Wildcards (Platzhalter) verwendet

%: kein bis beliebig viele Zeichen

_: genau ein Zeichen

```
SELECT *  
FROM person  
WHERE nachname LIKE 'M%';
```

- Alle Datensätze, bei denen der Nachname mit großem M anfängt:
z.B: Meyer, Mueller
Nicht: meyer, Schmitt

```
SELECT *  
FROM person  
WHERE nachname LIKE 'M_stermann';
```

- Alle Datensätze, bei denen Nachname mit großem M anfängt, dann ein Zeichen und dann *stermann* folgt:
z.B: Mastermann, Mistermann....
Nicht: Meistermann, mastermann

Verknüpfung von Bedingungen

Über die Operatoren AND und OR können Bedingungen miteinander verknüpft werden

```
SELECT *  
FROM person  
WHERE nachname LIKE 'M%'  
      AND gewicht != 60;
```

- Alle Datensätze, bei denen der Nachname mit großem M anfängt und das Gewicht ungleich 60 ist

```
SELECT *  
FROM person  
WHERE gewicht > 90  
      OR gewicht < 50;
```

- Alle Datensätze, bei denen der Gewicht größer 90 oder kleiner 60 ist.

Negation von Bedingungen

Der Operatoren NOT führt dazu, dass alle Datensätze angezeigt werden, bei denen die Bedingung nicht zutrifft.

```
SELECT *  
FROM person  
WHERE gewicht NOT IN(70.5, 80.5,90.5) ;
```

- Alle Datensätze, bei denen Gewicht **NICHT** 70.5, 80.5 oder 90.5 ist

Filtern von NULL-Werten

```
SELECT *  
FROM person  
WHERE geburtstag IS NULL;
```

- Alle Datensätze, bei denen der Geburtstag NULL ist
(hier ist auch IS NOT NULL möglich)

Achtung, da NULL für einen nicht definierten Wert steht, liefern Vergleiche mit NULL niemals TRUE zurück.

SELECT WHERE geburtstag = NULL → kein Ergebnis

Filtern und Anzeigen der Daten

- Um Duplikate (mehrfach vorkommende Werte) zu unterdrücken, kann der **DISTINCT-Befehl** verwendet werden

```
SELECT DISTINCT *  
FROM person;
```

- Unterdrückt Wiederholung identischer Datensätze

```
SELECT DISTINCT nachname,  
          geburtstag  
FROM person;
```

- Jede vorhandene Kombinat aus Nachname und Geburtstag wird nur einmal angezeigt

Ändern der Attributwerte eines/mehrerer Datensätze

```
UPDATE tabellenname  
SET spaltenname = wert, spaltenname = wert, ...  
WHERE bedingung;
```

Ändern eines Attributwertes für alle Datensätze

```
UPDATE person  
SET gewicht = 60.5,  
nachname = 'Meier';
```

- Bei ALLEN Datensätzen in der Tabelle Person wird das Gewicht auf 60.5 und der Nachname auf Meier gesetzt

Ändern eines Attributwertes für gezielte Datensätze

```
UPDATE person  
SET gewicht = 60.5  
WHERE gewicht = 70.5;
```

- Bei Datensätzen mit Gewicht von 70.5 wird es auf 60.5 gesetzt.

Ändern von Datensätzen

Neuen Attributwert aus altem berechnen

```
UPDATE person  
SET gewicht = gewicht * 1.05;
```

- Das Gewicht wird in allen Datensätzen um 5% erhöht
- Hier können auch andere arithmetische Operatoren (+ - /) verwendet werden

Löschen von Datensätzen

Löschen eines/mehrerer Datensätze aus einer Tabelle

```
DELETE FROM tabellenname  
WHERE bedingung;
```

Löschen aller Datensätze

```
DELETE FROM person;
```

- Löscht alle Datensätze aus der Tabelle Person

Löschen ausgewählter Datensätze

```
DELETE FROM person  
WHERE nachname NOT IN  
( 'Meyer' , 'Schuster' ) ;
```

- Löscht alle Datensätze aus der Tabelle Person, deren Nachname nicht Meyer oder Schuster ist

Übung – Beispielschema

Für die nächsten Übungsaufgaben werden die Tabellen emp und dept verwendet. Diese wurden bei Ausführung des Vorlesungs-SQL-Skriptes im eigenen Schema angelegt.

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17.12.1980	800	NULL	20
...

DEPT

DEPTNO	DNAME	LOC
20	RESEARCH	DALLAS
...

Löse die Aufgaben 1, 2 und 3 mit den vorgestellten Wildcards. Achte darauf, ob die Namen in der Tabelle emp groß oder klein geschrieben werden. (Case-Sensitivity)

1. Zeige Namen und Job aller Mitarbeiter an, deren Name ein a beinhaltet.
2. Zeige Name und Einstellungsdatum aller Mitarbeiter an, deren Name mit ,er' endet.
3. Zeige Mitarbeiternummer und Namen aller Mitarbeiter an, deren Name mit A beginnt und genau 5 Zeichen lang ist.
4. Zeige alle Informationen für die Mitarbeiter an, deren Einkommen höher als 2.000 ist.
5. Zeige alle Informationen für die Mitarbeiter an, deren Job Clerk oder Manager ist.
6. Zeige alle Informationen für die Mitarbeiter an, bei denen
 - a) Comm NULL ist.
 - b) Comm nicht NULL ist.
7. Zeige alle Mitarbeiter an, deren Einkommen kleiner als 1000 oder größer als 3000 ist.

8. Erzeuge mit Hilfe eines SQL-Befehls folgenden Output:

```
JOB
-----
CLERK
SALESMAN
PRESIDENT
MANAGER
ANALYST
```

9. Erhöhe das Einkommen aller Mitarbeiter, deren Comm NULL ist, um 300.

10. Setze das Einkommen des Mitarbeiters mit Mitarbeiternummer 7876 auf 2500 und Comm auf 500.

11. Füge einen neuen Mitarbeiter ein mit:

- Empno 1234, Name Mueller
- Job CLERK
- Manager mit Mitarbeiternummer 7839
- Einstellungsdatum heute
- Einkommen von 3000 (keine Provision (comm))
- Abteilung Accounting (finde hier die entsprechende DEPTNO heraus)

12. Lösche den Mitarbeiter mit EMPNO 1234 aus der Tabelle emp.