

Sortierung der Ausgabe einer Select-Anweisung:

SELECT Nachname, Vorname FROM Mitarbeiter **ORDER BY** Nachname;
(ORDER BY muss immer am Ende der Anweisung stehen und kann selbst etwas angehängt bekommen)

WHERE-Klausel (Kriterien/ Bedingungen):

SELECT Nachname, Ort FROM Mitarbeiter **WHERE** Ort = 'Karlsruhe'
→ mit AND/OR: SELECT Nachname, Ort FROM Mitarbeiter WHERE Ort = 'Karlsruhe' OR Ort = 'Freiburg';

Vergleichsoperatoren: =, <=, <, >=, >, <>

BETWEEN (bezieht Randwerte mit ein):

SELECT Gehalt FROM Mitarbeiter WHERE Mitarbeiternummer **BETWEEN 2000 AND 3000**;

IN (Vergleich mit vorgegebener Wertemenge):

SELECT Ename FROM Emp WHERE Deptno **IN (10, 30)**;

LIKE (Vergleich mit einem string):

SELECT Empno FROM Emp WHERE Ename **LIKE 'M%'**;
("%" ist Stellvertreter für 0 oder mehrere beliebige Zeichen, "_" ist Stellvertreter für genau ein Zeichen)
→ SELECT Mitarbeiternummer FROM Mitarbeiter WHERE Nachname LIKE '_A%';
→ liefert alle Mitarbeiternummern der Mitarbeiter deren Nachname an zweiter Stelle ein A hat

UPPER:

wird benutzt, wenn unbekannt ist, ob ein string groß oder klein geschrieben ist (bei Suchen)

Datum:

SELECT Empno, Ename FROM Emp WHERE Hiredate = '17.11.1981';
= '17-11-1981';
= '17/11/1981';
= '17-Nov-1981';
→ besser: SELECT Empno, Ename FROM Emp WHERE Hiredate = **TO_DATE ('11,17,1981', 'MM,DD,YYYY')**;

Datumsanzeige des eigenen Systems herausfinden (Format):

SELECT SYSDATE FROM DUAL;

Arithmetische Operationen mit Datumswerten:

SELECT Ename, (SYSDATE-Hiredate)/7 Wochen FROM Emp WHERE Deptno = 10;

Date Functions (liefern alle Werte vom Typ DATE zurück, außer MONTHS-BETWEEN liefert numerische Werte):

- MONTHS_BETWEEN (date1, date2) → ist date1 nach date 2 → positiv
→ ist date1 vor date 2 → negative
- ADD_MONTHS (date, n) → addiert n zu den Monaten (n kann negativ sein)
- NEXT_DAY (date, 'char') →
- LAST_DAY (date) → gibt letzten Tag des Monats an
- ROUND (date [, 'fmt']) →
 - CEIL (number) → liefert zur Dezimalzahl die nächst größere ganze Zahl
 - FLOOR (number) → liefert zur Dezimalzahl die nächst kleinere ganze Zahl

Formatsteuerung für Datumsfelder:

- Standardausgabeformat wird geändert durch TO_CHAR
- bei der Eingabe in die Datenbank wird TO_DATE benutzt
→ SELECT Nachname, **TO_CHAR** (Eintritts_Datum, 'DD.MM.YYYY') FROM Mitarbeiter;
(Tabelle heißt Eintritts_Datum)

Zeit:

- SELECT Nachname, TO_CHAR (Eintritts_Datum, 'DD.MM.YYYY **HH:MI**') FROM Mitarbeiter;
(gibt zusätzlich zum Datum die Zeit 12:00 an)
- SELECT Nachname, TO_CHAR (Eintritts_Datum, 'DD.MM.YYYY **HH24:MI**') Eintrittsdatum FROM Mitarbeiter;
(gibt zusätzlich zum Datum die Zeit 00:00 an)

Tabelle erstellen:

CREATE TABLE Mitarbeiter (Mitarbeiternummer INTEGER NOT NULL);

Tabelle automatisch befüllen (für jede Einfügeoperation wird ein neuer fortlaufender Wert generiert und eingesetzt):

(CREATE TABLE Mitarbeiter (Mitarbeiternummer INTEGER NOT NULL AUTO_INCREMENT);)

→ in Oracle: CREATE SEQUENCE P_SEQ;

INSERT INTO Mitarbeiter VALUES (P_SEQ.NEXTVAL, 'Maier', ...);

Einen Primary Key setzen:

CREATE TABLE Mitarbeiter (Mitarbeiternummer INTEGER NOT NULL PRIMARY KEY, ...);

Mehrere Primary Keys setzen:

CREATE TABLE Mitarbeiter (Mitarbeiternummer INTEGER NOT NULL,
Lfd_Nr INTEGER NOT NULL,
PRIMARY KEY (Mitarbeiternummer, Lfd_Nr));

Primary Key nachträglich setzen:

ADD (PRIMARY KEY (Mitarbeiternummer, Lfd_Nr));

DESCRIBE (Daten des Data Dictionary beschreiben):

DESCRIBE <Tabellenname>

→ wichtig: DESCRIBE USER_TABLES;

DESCRIBE USER_OBJECTS;

DESCRIBE USER_CATALOG;

DESCRIBE CAT;

Daten einfügen (neue Tabellenzeile):

INSERT INTO Mitarbeiter VALUES (4711, 'Maier', ...);

Daten ändern:

UPDATE Mitarbeiter SET Gehalt = 50 WHERE Mitarbeiternummer = 10;

DISTINCT:

Sorgt dafür, dass nur Daten ausgegeben werden, die sich in mindestens einem Feld unterscheiden

AS (neue Tabellenspalte anlegen):

SELECT Nachname AS Name, Vorname AS Vname, Ort AS Wohnort FROM Mitarbeiter;

Zählen:

- SELECT COUNT * (zählt alle Tupel (Tabellezeilen) inklusive Duplikate und Nulls)

- SELECT COUNT (Provision) FROM Mitarbeiter;

- SELECT COUNT (DISTINCT Provision) FROM Mitarbeiter;

String/ Zahlenkonstante einsetzen durch Konkatenationsoperator:

SELECT 'Das Gehalt von', Vorname || ' ' || Nachname, 'beträgt', Gehalt || 'Euro' FROM Mitarbeiter;

SUM, MAX, AVG, +, -, *, /:

- SELECT Mitarbeiternummer, Gehalt, Provision, Gehalt+Provision AS Endgehalt FROM Mitarbeiter;

- SELECT MAX (Gehalt) From Mitarbeiter;

IS NULL (nicht = NULL):

SELECT Ename FROM Emp WHERE Mgr IS NULL;

Funktionen sind möglich, z.B.: runden:

ROUND (n1, n2) rundet die Zahl n1 auf n2 Nachkommastellen

Kopie einer kompletten (Mitarbeiter-)Tabelle:

CREATE TABLE Mitarbeiter_2 AS SELECT * FROM Mitarbeiter;

Bestehende Tabelle mit Daten versorgen:

INSERT INTO Namenliste **SELECT** Nachname, Vorname **FROM** Mitarbeiter WHERE Ort = 'Karlsruhe';

COMMIT:

speichert nach regulärem Ende einer Transaktion

ROLLBACK:

kontrollierter Abbruch einer Transaktion

AUTOCOMMIT:

speichert automatisch nach jeder Eingabe

→ **SET AUTOCOMMIT OFF**

Explizite Sperren (unterschiedliche Grade: SHARE, EXCLUSIVE):

LOCK TABLE

→ Freigabe durch COMMIT oder ROLLBACK

User einrichten:

CREATE USER → Rechte vergeben (Werte lesen, ändern usw.)

Rechte vergeben:

GRANT SELECT, INSERT, UPDATE ON Tabelle1 **TO** Oma;