Handout
Themenfeld: Datenbanken und SQL
Abschnitt: 07.01. SQL: DML, DQL

Autor: Thomas Krause
Stand: 14.11.2022 12:03:00

Inhalt

1 Vorbereitung

Für den Vortrag wird verwendet:

- database_hochbau_MUSTER_DB.sql
- database_hochbau_MUSTER_DB_ERM_RM.dia

Installieren Sie bei Bedarf die Datenbank und verschaffen Sie sich nochmals einen Überblick.

2 DML – Data Manipulation Language

2.1 Überblick

- Aufgabe der DML: **Eingabe, Ändern und Löschen von Datensätzen/ Tupeln in Tabellen**
- Grundlegende Befehle:
  - Einfügen von Datensätzen/ Tupeln: INSERT
    - **INSERT INTO <tablename> VALUES (<wert_1>, <wert_2>, ..., <wert_n>);**
  - Ändern von Datensätzen/ Tupeln: UPDATE
    - **UPDATE <tablename> SET <attributname> = <arithmetischer Ausdruck>**
  - Löschen von Datensätzen/ Tupeln: DELETE
    - **DELETE FROM <tablename> WHERE <bedingung>**

2.2 Datensätze in Tabellen einfügen mit INSERT INTO

Aufgabe / Beispiel:

Ausgangstabelle:

tbl_abeilung

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau

Ergebnistabelle:

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
13	Buchhaltung

Beachte: Der Tabelle wurde 1 Datensatz hinzugefügt.

Syntax:

```
INSERT INTO <name_der_tabelle> (<name_spalte1>, <name_spalte2>, ..., <name_spalteN>)
       VALUES  (<wert1>, <wert2>, ..., <wertN>)[,
               (<wert1>, <wert2>, ..., <wertN>),...]
```

Erläuterung:

- nach VALUES innerhalb der Klammern wird immer ein einzufügender Datensatz mit den Werten angegeben, wie sie in der Spaltenliste nach dem Tabellennamen angegeben sind → Reihenfolge beachten, Datentypen beachten
- Wird die Liste der Spaltennamen nicht angegeben, muss die Reihenfolge der Werte in (<wert1>,<wert2>,...,<wertN>) so eingehalten werden, wie die Reihenfolge der Spalten für diese Tabelle im DBMS angelegt ist und es müssen alle Spalten in der Werte-Liste angegeben werden.

ZU BEACHTEN:

Wenn Fremdschlüsselbeziehungen zwischen den Tabellen einer Datenbank bestehen, müssen diese auch bei der Reihenfolge der Eingabe der Daten in die Tabellen beachtet werden. Erst müssen die Datensätze in die Tabellen eingefügt werden, die Primärschlüssel-Tabelle sind. Danach können die Datensätze in die Fremdschlüssel-Tabelle eingefügt werden.

praktische Anwendung:

Beispiel 1:

Ausgangstabelle:

tbl_abeilung

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau

SQL-Anweisung:

```
insert into tbl_abteilung(abteilungsnummer,abteilungname) values (13,'Buchhaltung');
```

Ergebnistabelle:

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
13	Buchhaltung

Beachte: Der Tabelle wurde 1 Datensatz hinzugefügt.

Beispiel 2:

Ausgangstabelle:

tbl_abeilung

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
13	Buchhaltung

SQL-Anweisung:

```
insert into tbl_abteilung(abteilungsnummer,abteilungname) values
        (15,'Logistik'),        -- 15 ist eine Zahl → deshalb keine Anführungszeichen
        (16,'Kundendienst'),    -- 'Kundendienst' ist eine Zeichenkette → Anführungszeichen
        (17,'Raumgestaltung');
```

Ergebnistabelle:

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
13	Buchhaltung
15	Logistik
16	Kundendienst
17	Raumgestaltung

Beachte: Der Tabelle wurden 3 Datensätze hinzugefügt.

Beispiel 3:

Ausgangstabelle:

tbl_mitarbeiter

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9

SQL-Anweisung:

```
insert into tbl_mitarbeiter (mitarbeiterPLZ, mitarbeitername, maschinenberechtigung,
mitarbeiternummer) values
        ('04106', 'Rohr', 1, 'M031'),
        ('04838', 'Stellmacher', 0, 'M032'),
        ('04509', 'Sauber', 0, 'M033');
```

Ergebnistabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

Beachte:

- Der Tabelle wurden 3 Datensätze hinzugefügt.
- Die Reihenfolge und Anzahl der Spalten wurde individuell angepasst.
- Spalten mit dem Constraint NOT NULL müssen ausgefüllt werden

2.3 Vorhandene Datensätze in Tabellen verändern mit UPDATE

Aufgabe / Beispiel:

Ausgangstabelle:

tbl_abeilung

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
13	Buchhaltung
15	Logistik
16	Kundendienst
17	Raumgestaltung

Ergebnistabelle:

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau und Sanierung
13	Buchhaltung
15	Logistik
16	Kundendienst
17	Raumgestaltung

Beachte: Datensatz mit der abteilungsnummer 12 wurde geändert.

Syntax:

```
UPDATE <name_der_tabelle>
SET <name_spalte1>=<wert1>[, <name_spalteN>=<wertN>]
WHERE <bedingung>
```

Erläuterung:

- Wertzuweisung an die zu ändernden Spalten nach SET
- die zu ändernden Datensätze müssen mit WHERE selektiert werden → ansonsten werden u.U. alle Datensätze verändert

praktische Anwendung:

Beispiel 1:

Ausgangstabelle:

tbl_abeilung

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
13	Buchhaltung
15	Logistik
16	Kundendienst
17	Raumgestaltung

SQL-Anweisung:

```
UPDATE tbl_abteilung SET abteilungname = 'Ausbau und Sanierung'
WHERE abteilungsnummer = 12;
```

Ergebnistabelle:

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau und Sanierung
13	Buchhaltung
15	Logistik
16	Kundendienst
17	Raumgestaltung

Beachte: Datensatz mit der abteilungsnummer 12 wurde geändert.

Beispiel 2:

Ausgangstabelle:

tbl_mitarbeiter

(siehe skript)

SQL-Anweisung:

```
UPDATE tbl_mitarbeiter SET
      mitarbeitername = 'Müller',
      mitarbeiterPLZ = '04156'
WHERE mitarbeiternummer = 'M010';;
```

Ergebnistabelle:

N.A.

Beachte: Kollege Stein hat geheiratet und heißt jetzt Müller und die neue PLZ ist jetzt 04156

2.4 Datensätze aus Tabellen löschen mit DELETE

Aufgabe/ Beispiel:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M024	B056	8.00
M024	B253	24.00

BEACHTE: Die Datensätze mit der Mitarbeiternummer M021 wurde aus der Tabelle gelöscht.

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M024	B056	8.00
M024	B253	24.00

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
NULL	NULL	NULL

BEACHTE: Alle Datensätze wurden aus der Tabelle gelöscht.

Syntax:

```
DELETE FROM <name_der_tabelle> [WHERE <logischer_ausdruck>];
```

Erläuterung:

- löscht Datensätze aus einer Tabelle
- ohne den WHERE-Ausdruck löscht diese Anweisung alle Datensätze aus der Tabelle → löscht die Datensätze aus der Tabelle, für die die Bedingung WAHR ist (siehe auch Hinweis im Kapitel unten)

Ein weitere Befehl zum kompletten Löschen aller Datensätze einer Tabelle:

```
TRUNCATE TABLE <name_der_tabelle>;
```

Hinweis:

- bei diesem Befehl ist kein WHERE möglich
- Bestehen beim Löschen bzw. bei TRUNCATE Fremdschlüsselbeziehungen auf die Datensätze einer Tabelle, müssen die Fremdschlüsselbeziehungen zunächst entfernt werden

Beachte:

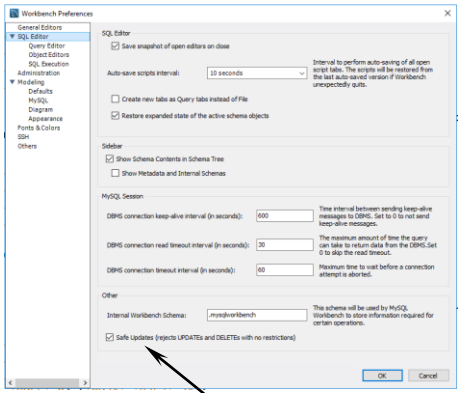
```
DELETE FROM <tabellenName>;
```

Diese Syntax wird bei einer unveränderten Standardkonfiguration zunächst zu einer Fehlermeldung führen:

"… Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect. …"

Lösung dieser Situation:

(1) Hinzufügen einer WHERE-Anweisung ODER
(2) Deaktivierung von "safe mode" (siehe unten)
Nach Änderung der Einstellung die Workbench neu starten.



bei Bedarf deaktivieren

praktische Anwendung:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

SQL-Anweisung:

```
delete from tbl_ma_auf_baustelle
where mitarbeiternummer='M021';
```

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M024	B056	8.00
M024	B253	24.00

BEACHTE: Die Datensätze mit der Mitarbeiternummer M021 wurde aus der Tabelle gelöscht.

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M024	B056	8.00
M024	B253	24.00

SQL-Anweisung:

```
delete from tbl_ma_auf_baustelle;
```

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
NULL	NULL	NULL

BEACHTE: Alle Datensätze wurden aus der Tabelle gelöscht.

Beispiel 3:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

SQL-Anweisung:

```
delete from tbl_ma_auf_baustelle
where stunden_auf_baustelle>=20;
```

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B021	12.00
M024	B056	8.00

Erläuterung: Es wurden alle Datensätze gelöscht, bei denen mehr als 20 Stunden bzw. genau 20 Stunden eingetragen waren.

3 DQL – Data Query Language

3.1 Überblick

- Aufgabe der DQL: **Abfrage von Daten aus einer (oder mehreren) Tabellen**
- Grundlegende Befehle:
  - **SELECT …**
  - Anzeige aller Datensätze aus einer Tabelle:
    SELECT * FROM <tablename>
  - Anzeige aller Datensätze aus der Tabelle ‚tbl_mitarbeiter'
    SELECT * FROM tbl_mitarbeiter
- Select-Grundoperationen:
  - Projektion
  - Selektion
  - Join

3.2 Datensätze aus einer Tabelle abfragen mit SELECT

Beispiele:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

Ergebnisliste:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

Erläuterung:

- eine Abfrage über alle Spalten und Datensätze aus der Tabelle tbl_mitarbeiter

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungsnummer
M009	Örtel	9
M010	Stein	12
M021	Hahn	10
M024	Holzer	9
M031	Rohr	NULL
M032	Stellmacher	NULL
M033	Sauber	NULL

Erläuterung:

- eine Abfrage über ausgewählte Spalten und alle Datensätze aus der Tabelle tbl_mitarbeiter

Beispiel 3:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

Ergebnisliste:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M024	0	Holzer	04119	9

Erläuterung:

- eine Abfrage über alle Spalten und ausgewählte Datensätze aus der Tabelle tbl_mitarbeiter

Syntax:

```
-- VARIANTE 1: allgemeine Syntax:
SELECT <liste_der_spaltennamen> FROM <name_der_tabelle> [WHERE <logischer_ausdruck>]

-- VARIANTE 2: Syntax mit Verwendung von Aliassen für Spalten
SELECT
        <spaltenname_1> [AS] ['|"| ]<alias_name1>[ |"|']['],
        <spaltenname_2> [AS] ['|"| ]<alias_name2>[ |"|']]['],
        <spaltenname_N> [AS] ['|"| ]<alias_nameN>[ |"|']]
FROM <name_der_tabelle> [WHERE <logischer_ausdruck>]
```

Erläuterung:

- <liste_der_spaltennamen> kann die Spalten der Tabelle in beliebiger Anzahl und Reihenfolge beinhalten
- wird für <liste_der_spaltennamen> ein * geschrieben, werden alle Spalten ausgegeben in der Reihenfolge, wie sie im DBMS angelegt wurden
- es werden nur die Datensätze ausgegeben, für die der logische Ausdruck nach dem WHERE den Wahrheitswert TRUE liefert
- zu den Spaltenaliassen:
    - die Anführungszeichen für die Aliasnamen müssen paarweise gleich verwendet werden
    - sollen keine Anführungszeichen verwendet werden, muss der Alias aus einem String bestehen (ohne Leerzeichen)

praktische Anwendungen:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

SQL-Anweisung:

```
SELECT * FROM tbl_mitarbeiter;          -- zeigt ALLE Datensätze und ALLE Spalten
```

Ergebnisliste:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

Erläuterung:

- eine Abfrage über alle Spalten und Datensätze aus der Tabelle tbl_mitarbeiter

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

SQL-Anweisung:

```
SELECT
	mitarbeiternummer,
	mitarbeitername,
	abteilungsnummer
FROM
	tbl_mitarbeiter;
```

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungsnummer
M009	Örtel	9
M010	Stein	12
M021	Hahn	10
M024	Holzer	9
M031	Rohr	NULL
M032	Stellmacher	NULL
M033	Sauber	NULL

Erläuterung:

- eine Abfrage über ausgewählte Spalten und alle Datensätze aus der Tabelle tbl_mitarbeiter

Beispiel 3:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

SQL-Anweisung:

```
SELECT *
FROM tbl_mitarbeiter
WHERE abteilungsnummer = 9;
```

Ergebnisliste:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M024	0	Holzer	04119	9

Erläuterung:

- eine Abfrage über alle Spalten und ausgewählte Datensätze aus der Tabelle tbl_mitarbeiter

Beispiel 4:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

SQL-Anweisung:

```
SELECT
      mitarbeitername,
      mitarbeiterPLZ
FROM tbl_mitarbeiter
WHERE abteilungsnummer = 9;
```

Ergebnisliste:

mitarbeitername	mitarbeiterPLZ
Örtel	04105
Holzer	04119

Erläuterung:

- eine Abfrage über alle Spalten und ausgewählte Datensätze aus der Tabelle tbl_mitarbeiter

Beispiel 5:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M031	1	Rohr	04106	NULL
M032	0	Stellmacher	04838	NULL
M033	0	Sauber	04509	NULL

SQL-Anweisung:

```
SELECT
       mitarbeiternummer AS "Mitarbeiter-Nr.",
       mitarbeitername AS 'Familien-Name',
       mitarbeiterPLZ PLZ,
       abteilungsnummer AS Abteilung_neu
FROM tbl_mitarbeiter
WHERE abteilungsnummer = 9;
```

Ergebnisliste:

Mitarbeiter-Nr.	Familien-Name	PLZ	Abteilung_neu
M009	Örtel	04105	9
M024	Holzer	04119	9

Erläuterung:

- für die Spaltenaliasse wurden unterschiedliche Schreibweisen gewählt

3.3 Begrenzung der Anzahl der auszugebenden Datensätze

Aufgabe/ Beispiel:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00

Erläuterung:

- es wurden nur die ersten 3 Datensätze angezeigt

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

Ergebnisliste:

mitarbeiternummer
M009
M010
M021
M024

Erläuterung:

- es wurden alle Datensätze der Tabelle angezeigt → dabei wurden alle die Datensätze unterdrückt, bei denen die Mitarbeiternummer mehrfach vorkommt

Syntax/ Aufbau/ Erläutern:

```
SELECT DISTINCT {<spalte> | <spaltenliste>}
FROM <name_der_tabelle>
WHERE <bedingung>
[LIMIT <numerischer_wert>];
```

praktische Anwendung/ Wirkungsweise/ Üben:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

SQL-Anweisung:

```
select * from tbl_ma_auf_baustelle limit 3;
```

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00

Erläuterung:

- es wurden nur die ersten 3 Datensätze angezeigt

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

SQL-Anweisung:

```
select distinct mitarbeiternummer from tbl_ma_auf_baustelle;
```

Ergebnisliste:

mitarbeiternummer
M009
M010
M021
M024

Erläuterung:

- es wurden alle Datensätze der Tabelle angezeigt → dabei wurden alle die Datensätze unterdrückt, bei denen die Mitarbeiternummer mehrfach vorkommt

3.4 Listen sortiert ausgeben

Aufgabe / Beispiel:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B021	12.00
M021	B056	21.00
M024	B056	8.00
M010	B112	23.00
M021	B112	24.00
M009	B253	37.00
M021	B253	34.00
M024	B253	24.00

Erläuterung:

- die Ausgangsliste wurde nach baustellennummer aufsteigend sortiert

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B021	12.00
M021	B056	21.00
M024	B056	8.00
M021	B112	24.00
M010	B112	23.00
M009	B253	37.00
M021	B253	34.00
M024	B253	24.00

Erläuterung:

- die Ausgangsliste wurde nach baustellennummer aufsteigend und nach stunden_auf_baustelle absteigend sortiert

Syntax:

```
SELECT <liste_der_spaltennamen>
FROM <name_der_tabelle>
[WHERE <bedingung>]
ORDER BY <name_der_spalte1>[ASC|DESC][, <name_der_spalteN>[ASC|DESC]];
```

Erläuterung:

- asc = ascending → aufsteigend
- desc = descending → absteigend
- Angabe von mehreren Sortierschlüsseln möglich
- wird keine Sortierrichtung angegeben, wird standardmäßig aufsteigend sortiert
- die Anweisung ORDER BY steht bei einem SELECT-Befehl immer ganz zum Schluß

praktische Anwendung:

Beispiel 1:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

SQL-Anweisung:

```
select * from tbl_ma_auf_baustelle
ORDER BY baustellennummer ASC;
```

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B021	12.00
M021	B056	21.00
M024	B056	8.00
M010	B112	23.00
M021	B112	24.00
M009	B253	37.00
M021	B253	34.00
M024	B253	24.00

Erläuterung:

- die Ausgangsliste wurde nach baustellennummer aufsteigend sortiert

Beispiel 2:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

SQL-Anweisung:

```
select * from tbl_ma_auf_baustelle
ORDER BY baustellennummer ASC, stunden_auf_baustelle DESC;
```

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B021	12.00
M021	B056	21.00
M024	B056	8.00
M021	B112	24.00
M010	B112	23.00
M009	B253	37.00
M021	B253	34.00
M024	B253	24.00

Erläuterung:

- die Ausgangsliste wurde nach baustellennummer aufsteigend und nach stunden_auf_baustelle absteigend sortiert

3.5 Datensätze mit Verwendung von WHERE selektieren (Grundlagen)

Aufgabe/ Beispiel:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B021	12.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B056	8.00
M024	B253	24.00

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M009	B253	37.00
M010	B112	23.00
M021	B056	21.00
M021	B112	24.00
M021	B253	34.00
M024	B253	24.00

Erläuterung:

- es werden nur Datensätze angezeigt, bei den die Spalte stunden_auf_baustelle Werte größer als 20 hat

Syntax/ Aufbau/ Erläutern:

```
SELECT * FROM  <name_der_tabelle>
WHERE <logischer_ausdruck>;
```

Erläuterung:

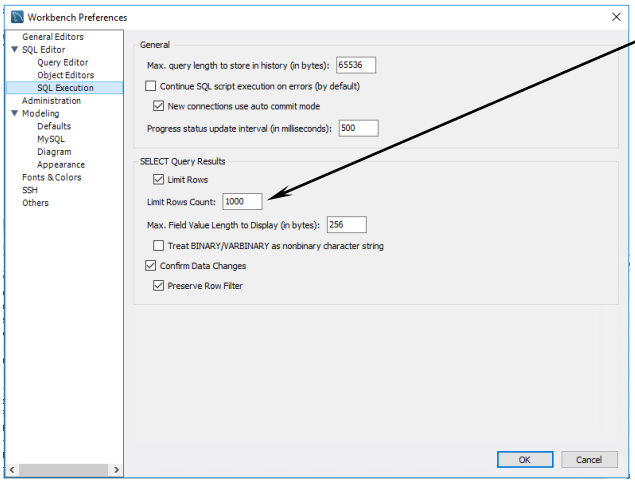
- der Befehl wird auf die Datensätze aus der Tabelle angewendet, für die die Bedingung WAHR ist
- in <logischer_ausdruck> des WHERE-Befehls können die Vergleichsoperatoren < > = <= >= != <> verwendet werden
- <logischer_ausdruck> ::= <name_einer_spalte> <vergleichsoperator> <vergleichswert>
- numerische Vergleichswerte sind ohne Anführungszeichen und bei Bedarf mit Dezimalpunkt anzugeben
- textliche Vergleichswerte (Zeichen, Strings) sind in Hochkomma einzuschließen
- mehrere Bedingungen können über logisches AND bzw. OR verknüpft werden

praktische Anwendung:

- N/A

Hinweis:

Begrenzung der Anzahl der Datensätze bei Abfragen



Voreinstellung = Begrenzung

3.6 Zusammenfassung von mehreren Ergebnislisten

Aufgabe/ Zweck/ Beispiel/ Demonstrieren:

Ausgangstabellen → Ergebnisse von 2 separaten Abfragen:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M024	0	Holzer	04119	9

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M010	1	Stein	04838	12

Ergebnisliste:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M024	0	Holzer	04119	9
M010	1	Stein	04838	12

Syntax/ Aufbau/ Erläutern:

```
<SELECT-Anweisung1>
UNION [ALL]
<SELECT-Anweisung2>
[UNION …]
```

Erläuterung:

- Zusammenfassung von Ergebnislisten von 2 oder mehr SQL-Abfragen zu einer einzigen Liste
- Die Ergebnislisten der einzelnen SELECT-Anweisungen müssen in Anzahl und Datentyp der Spalten übereinstimmen (… und natürlich auch die Bedeutung der einzelnen Spalten).
- Die durch UNION entstandene gesamte Liste wird sortiert und doppelte Datensätze werden nicht angezeigt (siehe DISTINCT).
- Mit dem Befehl UNION ALL werden die Ergebnisse aller Teilabfragen komplett (ohne DISTINCT) geliefert.

praktische Anwendung/ Wirkungsweise/ Üben:

N/A