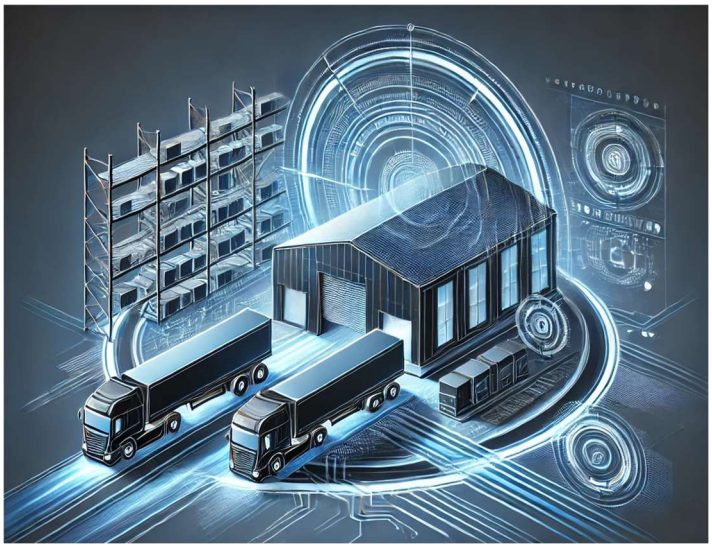
Hochschule Bremerhaven
University of Applied Sciences

Projekt: Logistik in 20 Jahren



Das Projekt „Logistik in 20 Jahren“ entwickelt eine zukunftsorientierte Datenbanklösung zur effizienten Verwaltung von Waren, Lagerbeständen und Transportprozessen. Durch den Einsatz moderner SQL-Abfragen, automatisierter Bestandsanalysen und optimierter Transportplanung ermöglicht das System eine realistische Simulation logistischer Abläufe, die in Unternehmen integriert und als Grundlage für zukünftige Optimierungen genutzt werden kann.

Name: Fabian Tober
Matrikel-Nummer: 40607
E-Mail: fabian.tober@hs-bremerhaven.de

Zeitraum: 17.10.2024 - 27.02.2025
Fertigstellung: 26.02.2025

Inhaltsverzeichnis

Semesteraufgabe 1 [17.10.2024]

1.1 Aufgabe:

Installieren und Testen des Oracle SQL Developer Data Modeler.

1.2 Lösung:

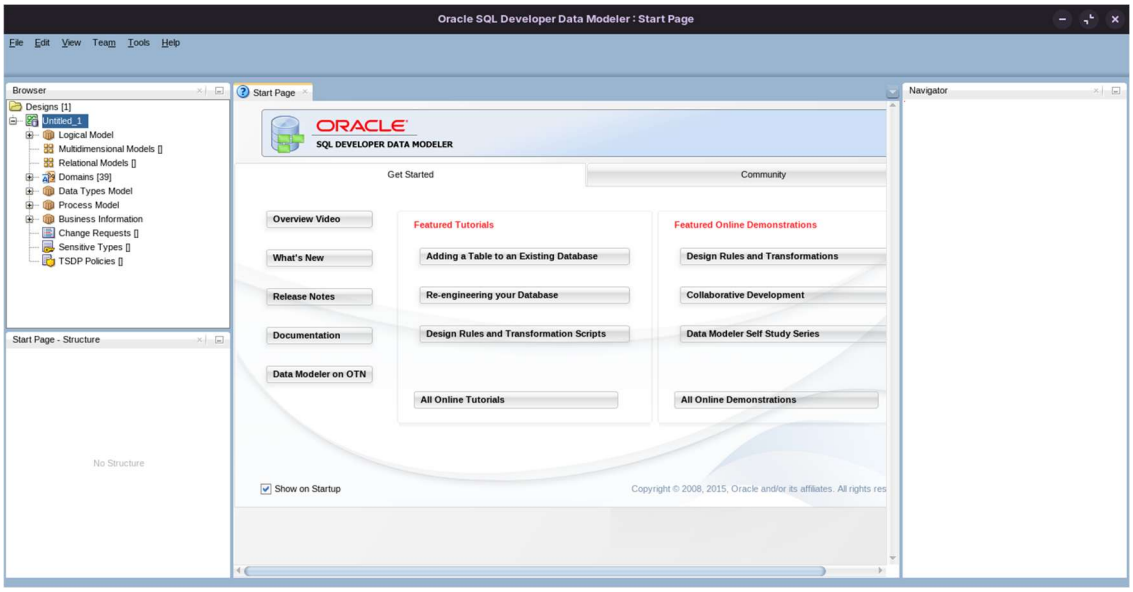


Abbildung 1 - SQL Developer Data Modeler

Wie man hier sieht, wurde die Aufgabe erfolgreich abgeschlossen, indem der Oracle SQL Developer Data Modeler erfolgreich installiert wurde.

Semesteraufgabe 2 [24.10.2024]:

1.1 Aufgabe:

Beschreiben und Visualisieren des Projekts „Logistik in 20 Jahren“ (mindestens 2 Seiten)

1.2 Lösung:

Anhang 1: „Beschreibung und Visualisierung des Projekts Logistik in 20 Jahren.pdf“

2.1 Aufgabe:

Erstellung von 4 Objekttypen bzw. Objektklassen mit mindestens 4 Attribute pro Klasse sowie mindestens 3 Beziehungen (1:1; 1:n; m:n) sowie Schlüssel. Zum Thema „Logistik in 20 Jahren“. Dafür soll ein Entity-Entwurf gemacht werden.

2.2 Lösung Entity-Entwurf:

1. Entitäten & Attribute

Lager

- **Primärschlüssel:** Lagerstandort
- **Attribute:**
    - Kapazität
    - Spezialisierung
    - Automatisierungsgrad

Ware

- **Primärschlüssel:** Gewicht
- **Attribute:**
    - Bezeichnungstyp
    - Volumen
    - Gefahrgut
- **Fremdschlüssel:** Lagerstandort (bezogen auf Lager)

Transportauftrag

- **Primärschlüssel:** Standort
- **Attribute:**
    - Auftragsdatum
    - Zielort
    - Lieferfrist
- **Fremdschlüssel:** Gewicht (bezogen auf Ware)

Fahrzeug

- **Primärschlüssel:** Kapazität
- **Attribute:**
    - Energiequelle
    - Hersteller
    - Typ
- **Fremdschlüssel:** Standort (bezogen auf Transportauftrag)

2. Beziehungen zwischen Entitäten

- **Lager → Ware** (**1:N**)
  - Ein Lager kann mehrere Waren enthalten.
  - Eine Ware gehört zu genau einem Lager.
  - **Fremdschlüssel:** Lagerstandort in Ware referenziert Lagerstandort in Lager.
- **Ware → Transportauftrag** (**M:N**)
  - Eine Ware kann in mehreren Transportaufträgen transportiert werden.
  - Ein Transportauftrag kann mehrere Waren enthalten.
  - **Lösung:** Diese M:N-Beziehung wird durch die **Relation_4** (Zwischentabelle) realisiert.
  - **Fremdschlüssel:**
    - Ware_Gewicht referenziert Gewicht in Ware.
    - Transportauftrag_Standort referenziert Standort in Transportauftrag.
- **Transportauftrag → Fahrzeug** (**1:N**)
  - Ein Transportauftrag wird von genau einem Fahrzeug ausgeführt.
  - Ein Fahrzeug kann mehrere Transportaufträge ausführen.
  - **Fremdschlüssel:** Transportauftrag_Standort in Fahrzeug referenziert Standort in Transportauftrag.

3. Geschäftsregeln & Einschränkungen

1. Jedes **Lager** kann nur Waren speichern, die sein **Kapazitätslimit** nicht überschreiten.
2. Eine **Ware** muss genau einem **Lager** zugeordnet sein.
3. Ein **Transportauftrag** kann nur für eine Ware erstellt werden, die sich in einem Lager befindet.
4. Fahrzeuge dürfen nur Transportaufträge annehmen, wenn ihre **Kapazität** ausreicht.
5. Gefahrgut benötigt spezielle Fahrzeuge mit einer entsprechenden **Energiequelle** oder Sicherheitsvorkehrungen.

4. Normalisierungsüberlegungen

- Die Entitäten sind bereits in **3. Normalform (3NF)**, da alle Attribute **voll funktional abhängig** von ihrem Primärschlüssel sind.
- Falls nötig, könnte man eine zusätzliche Tabelle für Fahrzeugtypen einführen, falls sich Hersteller und Typ oft wiederholen.
- Falls eine **Ware an mehreren Standorten lagern kann**, müsste man eine separate Zuordnungstabelle (Lager_Ware) erstellen.

3.1 Aufgabe:

Erstellung eines ER-Diagramms.

3.2 Lösung:

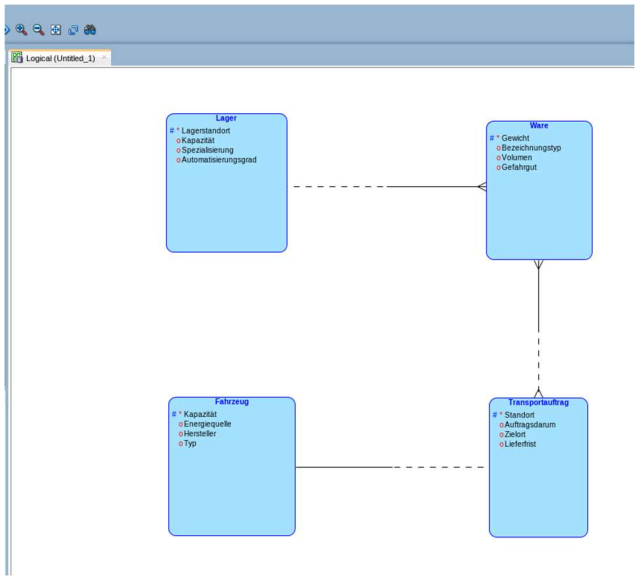


Abbildung 2 - ER_Diagramm

4.1 Aufgabe:

Installieren und Testen der H2 Database

4.2 Lösung:

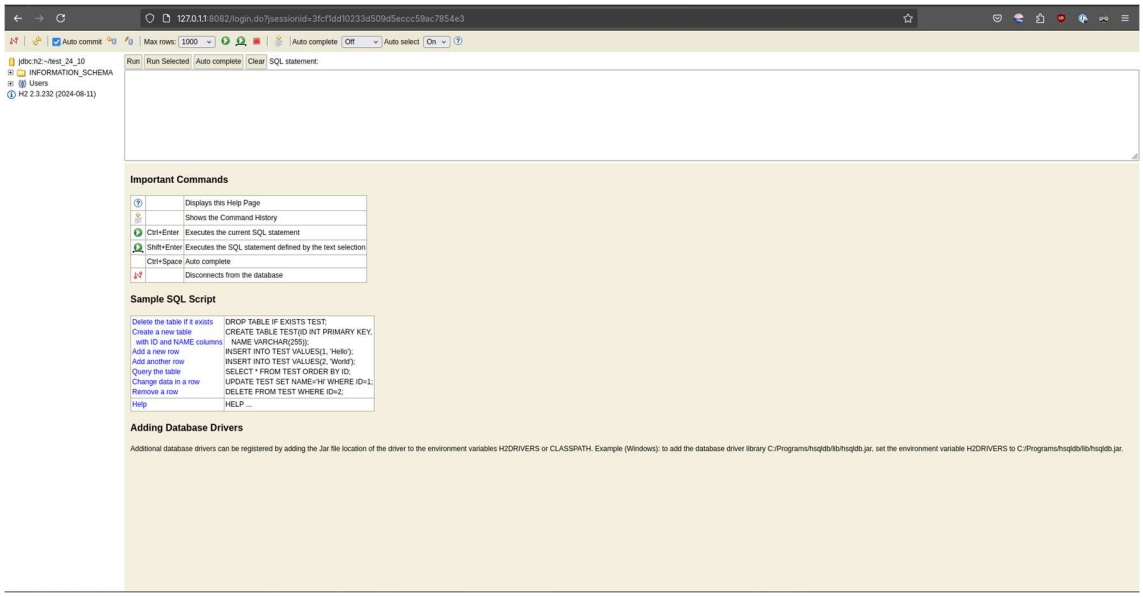


Abbildung 3 – H2 Database install

5.1 Aufgabe:

Installieren und Testen von DBeaver.

5.2 Lösung:

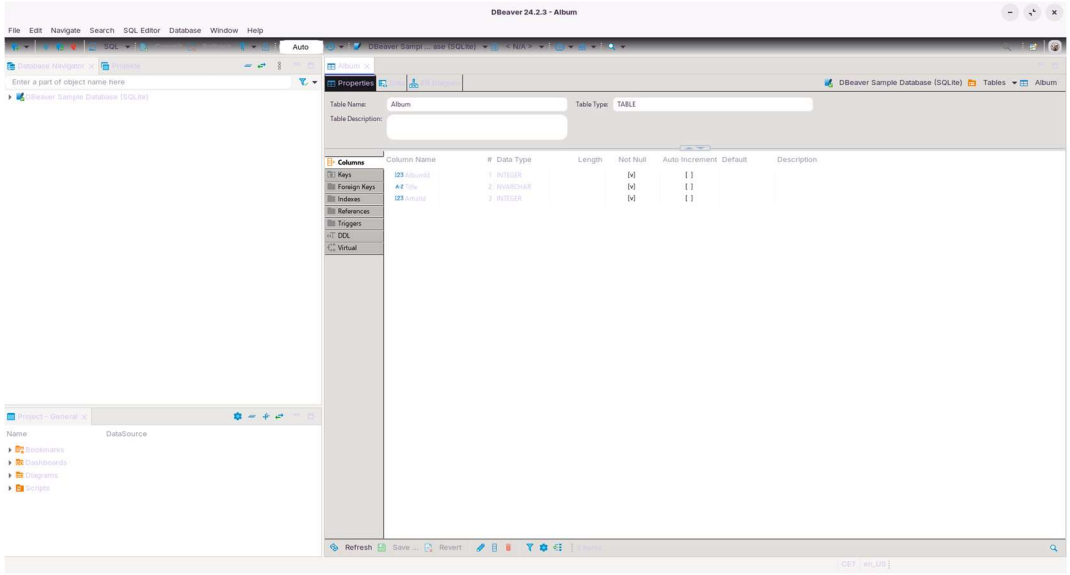


Abbildung 4 - DBeaver install

5.3 Anhang:

SQL_erzeugnis.ddl; SQL_erzeugnis.sql; test_24_10.mv.db; OSDM; OSDM2

Semesteraufgabe 3 [07.11.2024]:

1.1 Aufgabe:

ER-Diagramm in ein relationales Schema abbilden.

1.2 Lösung:

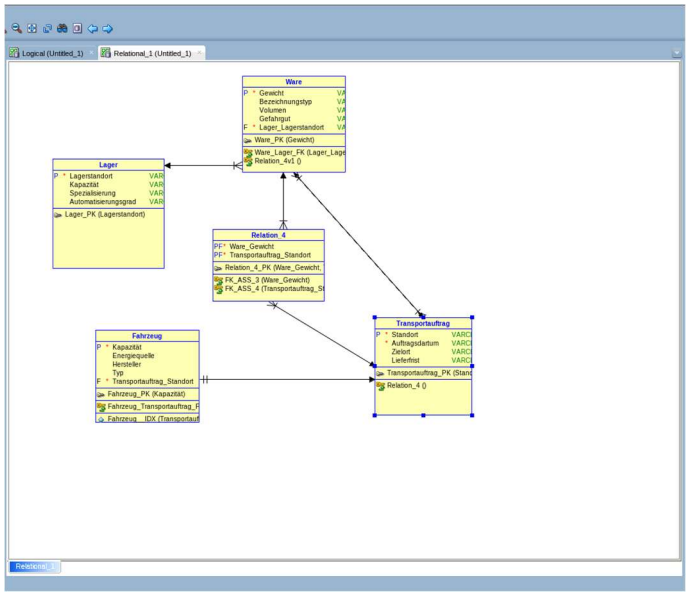


Abbildung 5 – Relationales Schema

2.1 Aufgabe:

Erstellung eines Relationalen Entwurf.

2.2 Lösung:

Relationaler Entwurf

1. Entitäten und Attribute

Lager (Lager)

- Lagerstandort (Primärschlüssel)
- Kapazität
- Spezialisierung
- Automatisierungsgrad

Ware (Ware)

- Gewicht (Primärschlüssel)
- Bezeichnungstyp
- Volumen
- Gefahrgut
- Lager_Lagerstandort (Fremdschlüssel → Lager.Lagerstandort)

Transportauftrag (Transportauftrag)

- Standort (Primärschlüssel)
- Auftragsdatum
- Zielort
- Lieferfrist

Fahrzeug (Fahrzeug)

- Kapazität (Primärschlüssel)
- Energiequelle
- Hersteller
- Typ
- Transportauftrag_Standort (Fremdschlüssel → Transportauftrag.Standort)

Relation_4 (Zwischentabelle für M:N-Beziehung zwischen Ware und Transportauftrag)

- Ware_Gewicht (Primärschlüssel, Fremdschlüssel → Ware.Gewicht)
- Transportauftrag_Standort (Primärschlüssel, Fremdschlüssel → Transportauftrag.Standort)

2. Beziehungen und Fremdschlüssel

1. **Lager – Ware (1:N)**
    - **Fremdschlüssel:** Ware.Lager_Lagerstandort → Lager.Lagerstandort
    - **Bedeutung:** Ein Lager kann mehrere Waren enthalten, aber jede Ware gehört genau zu einem Lager.
2. **Ware – Transportauftrag (M:N) über Relation_4**
    - **Fremdschlüssel:**
        - Relation_4.Ware_Gewicht → Ware.Gewicht
        - Relation_4.Transportauftrag_Standort → Transportauftrag.Standort
    - **Bedeutung:** Eine Ware kann in mehreren Transportaufträgen transportiert werden, und ein Transportauftrag kann mehrere Waren enthalten.
3. **Transportauftrag – Fahrzeug (1:N)**
    - **Fremdschlüssel:** Fahrzeug.Transportauftrag_Standort → Transportauftrag.Standort
    - **Bedeutung:** Ein Fahrzeug kann mehrere Transportaufträge übernehmen, aber jeder Transportauftrag wird von genau einem Fahrzeug ausgeführt.

3. Geschäftsregeln und Einschränkungen

- Eine Ware **muss** einem Lager zugeordnet sein (FK Ware.Lager_Lagerstandort ist **NOT NULL**).
- Ein Transportauftrag kann nur für existierende Waren erteilt werden (über Relation_4).
- Ein Fahrzeug kann nur einem existierenden Transportauftrag zugewiesen werden.
- **Referentielle Integrität** muss durch ON DELETE/ON UPDATE Regeln gesichert sein (z. B. CASCADE oder SET NULL).

4. Normalisierungsüberlegungen

- **1. Normalform (1NF)**: Alle Attribute sind atomar (keine mehrfachen Werte in einem Feld).
- **2. Normalform (2NF)**: Keine partiellen Abhängigkeiten, da alle Nicht-Schlüssel-Attribute vom gesamten Primärschlüssel abhängen.

- **3. Normalform (3NF)**: Keine transitive Abhängigkeiten zwischen Nicht-Schlüssel-Attributen.

3.1 Aufgabe:

Erläuterung Fremdschlüssel-Beziehung.

3.2 Lösung:

Fremdschlüssel	Referenziert	Funktion
Ware.Lager_Lagerstandort	Lager.Lagerstandort	Gewährleistet, dass jede Ware in einem existierenden Lager liegt.
Relation_4.Ware_Gewicht	Ware.Gewicht	Stellt sicher, dass nur existierende Waren in Transportaufträgen enthalten sind.
Relation_4.Transportauftrag_Standort	Transportauftrag.Standort	Stellt sicher, dass Transportaufträge nur mit existierenden Standorten verknüpft werden.
Fahrzeug.Transportauftrag_Standort	Transportauftrag.Standort	Gewährleistet, dass ein Fahrzeug nur existierenden Transportaufträgen zugewiesen wird.

Die Tabellen in der Datenbank wurden für die nächsten Aufgaben leicht angepasst, um besser auf die Anforderungen einzugehen. Da das grundlegende Prinzip jedoch unverändert bleibt, bleiben die Tabellen und Daten aus den Aufgaben 1–3 in ihrer aktuellen Form bestehen. Diese basieren auf dem ursprünglichen Entity-Model und den bisherigen Datenstrukturen.

4.1 Aufgabe:

Create-Table-Anweisung generieren und Filtern sowie Tabellen erstellen (später).

4.2 Lösung:

```
CREATE TABLE Fahrzeug
 (
  Fahrzeug_ID            VARCHAR2 (100) NOT NULL ,
  Modell                 VARCHAR2 (50) ,
  Kapazität              VARCHAR2 (1000) ,
  Energie_Quelle         VARCHAR2 (30) ,
  Hersteller             VARCHAR2 (50) ,
  Transportauftrag_Auftrags_ID VARCHAR2 (100) NOT NULL ,
  PRIMARY KEY ( Fahrzeug_ID ) ,
  FOREIGN KEY ( Transportauftrag_Auftrags_ID ) REFERENCES Transportauftrag ( Auftrags_ID )
 ) ;


CREATE TABLE Lager
 (
  Lager_ID               VARCHAR2 (100) NOT NULL ,
  Lagerstandort      VARCHAR2 (50) ,
  Kapazität              VARCHAR2 (1000) ,
  Spezialisierung        VARCHAR2 (100) ,
  Automatisierungsgrad VARCHAR2 (100) ,
  PRIMARY KEY ( Lager_ID )
 ) ;
```

„Datenbank_erzeugen.ddl“ enthält den automatisch generierten Code des SQL Developer Data Modelers, während die Datei „Bereinigt - Table Create.txt“ die von mir bereinigte und angepasste Version des Codes enthält, entsprechend den Vorgaben aus der Vorlesung.

4.3 Anhang:

Datenbank_erzeugen.ddl; Bereinigt - Table Create.txt;

5.1 Aufgabe:

Die erstellten Tabellen mit einigen Datenfüllen, dabei mindestens 3 Insert-Operationen pro Tabelle verwenden.

5.2 Lösung:

SELECT * FROM FAHRZEUG;

FAHRZEUG_ID	MODELL	KAPAZITÄT	ENERGIE_QUELLE	HERSTELLER	TRANSPORTAUFTRAG_AUFTRAGS_ID
125	KLKW	50	E	Scania	555
10	LKW	410	E	Scania	447
43	LKW	153	D	MAN	4125

(3 rows, 3 ms)

SELECT * FROM LAGER;

LAGER_ID	LAGERSTANDORT	KAPAZITÄT	SPEZIALISIERUNG	AUTOMATISIERUNGSGRAD
14	BER	147	E	73
24	BRE	23	D	95
12	MUN	504	F	43
2	HUM	873	A	99

(4 rows, 1 ms)

SELECT * FROM TRANSPORTAUFTRAG;

AUFTRAGS_ID	AUFTRAGSDATUM	STANDORT	ZIELORT	LIEFERFRIST
555	SEP	BER	FAM	2
447	SEP	BER	BRE	5
4125	JUN	MUN	HAM	8

Abbildung 6 – Tabellen erstellung und füllung 1

SELECT * FROM WARE ;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 2 ms)

Abbildung 7 – Tabellen erstellung und füllung 2

Genutzte Befehle:

```
INSERT INTO LAGER (LAGER_ID, LAGERSTANDORT, KAPAZITÄT, SPEZIALISIERUNG, AUTOMATISIERUNGSGRAD) VALUES (14, 'BER', 147, 'E', 73);
```

Dieser Befehl ist ein Beispiel anhand der ersten Zeile der Lagertabelle. Zusätzlich folgt ein weiteres Beispiel, das zeigt, wie mehrere Zeilen gleichzeitig eingefügt werden können. Letzteres dient nur zur Veranschaulichung und existiert so nicht in der Datenbank.

```
INSERT INTO BESTELLUNGEN (AUFTRAGS_ID, BESTELLER, ZIELORT, WARE_ID, FAHRZEUG_ID, LIEFERDATUM) VALUES
(1001, 'Alpha Robotics', 'Lunar Base 1', 147, 125, '2045-01-01'),
(1002, 'SpaceCargo', 'Mars Colony', 25, 10, '2045-03-15'),
(1003, 'TerraSupplies', 'Earth Hub', 7, 43, '2045-06-10'),
(1004, 'Lunar Supplies', 'Lunar Base 2', 47, NULL, '2045-08-22');
```

Semesteraufgabe 4 [14.11.2024]:

1.1 Aufgabe:

Ausführen einiger SQL-Operationen (siehe Bild).

SELECT * **FROM** *TABELLE*
Lesen aller Spalten einer Tabelle

SELECT *SPALTE1,SPALTE2,...* **FROM** *TABELLE*
Lesen einzelner Spalten einer Tabelle

SELECT *SPALTE1,SPALTE2,...* **FROM** *TABELLE* **ORDER BY** *SPALTE1,SPALTE2,...*
Aufsteigendes Sortieren des Spalteninhalts

SELECT *SPALTE1,SPALTE2,...* **FROM** *TABELLE* **ORDER BY** *SPALTE1,...* **DESC**
Absteigendes Sortieren des Spalteninhalts

SELECT * **FROM** *TABELLE* **WHERE** *BEDINGUNG*
Selektives Lesen einzelner Zeilen

SELECT DISTINCT *SPALTE1,...* **FROM** *TABELLE*
In der Ergebnistabelle keine doppelten Zeilen

1.2 Lösung:

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 1 ms)

Abbildung 1_4 - SELECT_1

SELECT WAREN_ID, BEZEICHNUNG FROM WARE;

WAREN_ID	BEZEICHNUNG
147	E154
25	E554
47	D33
7	F14
258	A548

(5 rows, 1 ms)

Abbildung 2_4 - SELECT_2

SELECT * FROM WARE ORDER BY GEFAHRGUT, WAREN_ID ;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
258	A548	400	79	76	N	2
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
25	E554	4	5	15	Y	14

(5 rows, 2 ms)

Abbildung 3_4 - ORDER_BY_1

SELECT * FROM WARE ORDER BY GEFAHRGUT DESC, WAREN_ID DESC;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
25	E554	4	5	15	Y	14
7	F14	150	51	66	N	12
47	D33	550	55	114	N	24
258	A548	400	79	76	N	2
147	E154	43	51	14	N	14

(5 rows, 4 ms)

Abbildung 3_4 – ORDER_BY_2

SELECT * FROM WARE WHERE GEFAHRGUT = 'Y';

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
25	E554	4	5	15	Y	14

(1 row, 1 ms)

Abbildung 5_4 - WHERE

SELECT DISTINCT GEFAHRGUT FROM WARE;

GEFAHRGUT
N
Y

(2 rows, 2 ms)

Abbildung 6_4 - DISTINCT

Semesteraufgabe 5 [28.11.2024]:

1.1 Aufgabe:

SQL-Operationen ausführen: UPDATE, ALTER, DELETE jeweils 4x pro Operation.

1.2 Lösung:

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 0 ms)

ALTER TABLE WARE ADD lagerbestand INT;
Update count: 0
(34 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	LAGERBESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

Abbildung 8 - ALTER_1

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	LAGERBESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

ALTER TABLE WARE RENAME COLUMN lagerbestand TO bestand;
Update count: 0
(1 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

Abbildung 9 - ALTER_2

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 0 ms)

ALTER TABLE WARE ALTER COLUMN bestand SET DATA TYPE VARCHAR(50);
Update count: 0
(25 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

Abbildung 10 - ALTER_3

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 0 ms)

ALTER TABLE WARE DROP COLUMN bestand;
Update count: 0
(14 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 0 ms)

Abbildung 11 - ALTER_4

Diese vier Abbildungen zeigen die Lösungen zur ALTER-Operation. Die obere Tabelle stellt die ursprüngliche Tabelle dar, während die untere das Ergebnis nach der ALTER-Operation zeigt. Die durchgeführte ALTER-Operation befindet sich zwischen den beiden Tabellen.

Dasselbe Prinzip gilt auch für die folgenden Operationen - DELETE, NULL und UPDATE – jeweils in genau dieser Reihenfolge. Jede Operation wird in vier Abbildungen dargestellt: oben die Originaltabelle, dann die entsprechende Operation und darunter das Ergebnis.

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 0 ms)

DELETE FROM WARE WHERE WAREN_ID = 147;
Update count: 1
(1 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(4 rows, 1 ms)

Abbildung 12 - DELETE_1

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	LAGERBESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

ALTER TABLE WARE RENAME COLUMN lagerbestand TO bestand;
Update count: 0
(1 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	*null*
25	E554	4	5	15	Y	14	*null*
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	*null*
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

Abbildung 13 - DELETE_2

SELECT * FROM WARE_TEST;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
2	Stuhl	0.80	M	5.00	FALSE	102
5	Regal	2.50	XL	15.00	FALSE	105
6	Stofftisch	1.00	M	6.00	TRUE	106
7	Lampe	0.40	S	1.20	FALSE	107

(4 rows, 1 ms)

DELETE FROM WARE_TEST;
Update count: 4
(1 ms)

SELECT * FROM WARE_TEST;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID

(no rows, 1 ms)

Abbildung 14 - DELETE_3

SELECT * FROM WARE_TEST;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
1	Tisch	1.50	L	10.50	FALSE	101
2	Stuhl	0.80	M	5.00	FALSE	102
3	Tischlampe	0.50	S	2.00	FALSE	103
4	Kleiner Tisch	1.20	L	7.00	FALSE	104
5	Regal	2.50	XL	15.00	FALSE	105
6	Stofftisch	1.00	M	6.00	TRUE	106
7	Lampe	0.40	S	1.20	FALSE	107
8	Tischdecke	0.30	S	0.50	FALSE	108

(8 rows, 1 ms)

DELETE FROM WARE_TEST WHERE BEZEICHNUNG LIKE '%Tisch%';
Update count: 4
(1 ms)

SELECT * FROM WARE_TEST;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
2	Stuhl	0.80	M	5.00	FALSE	102
5	Regal	2.50	XL	15.00	FALSE	105
6	Stofftisch	1.00	M	6.00	TRUE	106
7	Lampe	0.40	S	1.20	FALSE	107

(4 rows, 1 ms)

Abbildung 15 - DELETE_4

SELECT * FROM LOGISTIK_ZUKUNFT;

PRODUKT_ID	PRODUKT_NAME	LAGER_ID	TEMPERATUR	LAGERBESTAND	LIEFERDATUM
1	Solarpanel 3000	101	25.5	500	2045-01-10
2	Quantum-Batterie	102	*null*	300	*null*
3	Hyperloop-Modul	*null*	*null*	*null*	2045-06-15
4	KI-Chip X7	103	-10.0	*null*	*null*
5	Holo-Display	*null*	*null*	200	2045-03-22

(5 rows, 1 ms)

SELECT * FROM logistik_zukunft WHERE (temperatur > 0 OR lagerbestand < 55) AND temperatur IS NOT NULL;

PRODUKT_ID	PRODUKT_NAME	LAGER_ID	TEMPERATUR	LAGERBESTAND	LIEFERDATUM
1	Solarpanel 3000	101	25.5	500	2045-01-10

(1 row, 2 ms)

Abbildung 16 - NULL_1

SELECT * FROM LOGISTIK_ZUKUNFT;

PRODUKT_ID	PRODUKT_NAME	LAGER_ID	TEMPERATUR	LAGERBESTAND	LIEFERDATUM
1	Solarpanel 3000	101	25.5	500	2045-01-10
2	Quantum-Batterie	102	*null*	300	*null*
3	Hyperloop-Modul	*null*	*null*	*null*	2045-06-15
4	KI-Chip X7	103	-10.0	*null*	*null*
5	Holo-Display	*null*	*null*	200	2045-03-22

(5 rows, 0 ms)

SELECT * FROM logistik_zukunft WHERE temperatur IS NULL OR lagerbestand IS NULL;

PRODUKT_ID	PRODUKT_NAME	LAGER_ID	TEMPERATUR	LAGERBESTAND	LIEFERDATUM
2	Quantum-Batterie	102	*null*	300	*null*
3	Hyperloop-Modul	*null*	*null*	*null*	2045-06-15
4	KI-Chip X7	103	-10.0	*null*	*null*
5	Holo-Display	*null*	*null*	200	2045-03-22

(4 rows, 2 ms)

Abbildung 17 - NULL_2

SELECT * FROM LOGISTIK_ZUKUNFT ;

PRODUKT_ID	PRODUKT_NAME	LAGER_ID	TEMPERATUR	LAGERBESTAND	LIEFERDATUM	LUFTFEUCHTE
1	Solarpanel 3000	101	25.5	500	2045-01-10	30.0
2	Quantum-Batterie	102	*null*	300	*null*	50.0
3	Hyperloop-Modul	*null*	*null*	*null*	2045-06-15	10.0
4	KI-Chip X7	103	-10.0	*null*	*null*	-5.0
5	Holo-Display	*null*	*null*	200	2045-03-22	40.0

(5 rows, 20 ms)

SELECT * FROM logistik_zukunft WHERE temperatur IS NOT NULL;

PRODUKT_ID	PRODUKT_NAME	LAGER_ID	TEMPERATUR	LAGERBESTAND	LIEFERDATUM	LUFTFEUCHTE
1	Solarpanel 3000	101	25.5	500	2045-01-10	30.0
4	KI-Chip X7	103	-10.0	*null*	*null*	-5.0

(2 rows, 0 ms)

Abbildung 18 - NULL_3

SELECT * FROM LIEFERWEGE;

LIEFERWEG_ID	PRODUKT_ID	TRANSPORTMITTEL	STATUS
1	1	Drohne	In Auslieferung
2	2	*null*	Geplant
3	3	Hyperloop	Verspätet
4	4	*null*	Unbekannt
5	5	Roboter-LKW	In Auslieferung

(5 rows, 1 ms)

SELECT * FROM LOGISTIK_ZUKUNFT ;

PRODUKT_ID	PRODUKT_NAME	LAGER_ID	TEMPERATUR	LAGERBESTAND	LIEFERDATUM	LUFTFEUCHTE
1	Solarpanel 3000	101	25.5	500	2045-01-10	30.0
2	Quantum-Batterie	102	*null*	300	*null*	50.0
3	Hyperloop-Modul	*null*	*null*	*null*	2045-06-15	10.0
4	KI-Chip X7	103	-10.0	*null*	*null*	-5.0
5	Holo-Display	*null*	*null*	200	2045-03-22	40.0

(5 rows, 0 ms)

SELECT produkt_id, produkt_name, temperatur, luftfeuchte FROM logistik_zukunft WHERE temperatur <= luftfeuchte;

PRODUKT_ID	PRODUKT_NAME	TEMPERATUR	LUFTFEUCHTE
1	Solarpanel 3000	25.5	30.0
4	KI-Chip X7	-10.0	-5.0

(2 rows, 1 ms)

Abbildung 19 - NULL_4

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 1 ms)

UPDATE WARE SET GEFAHRGUT = 'Y' WHERE WAREN_ID = 147;
Update count: 1
(6 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	Y	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 0 ms)

Abbildung 20 - UPDATE_1

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 1 ms)

UPDATE WARE SET GEFAHRGUT = 'Y', BEZEICHNUNG = 'X111' WHERE WAREN_ID = 147;
Update count: 1
(3 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	X111	43	51	14	Y	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 1 ms)

Abbildung 21 - UPDATE_2

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 0 ms)

UPDATE WARE SET GEFAHRGUT = 'N';
Update count: 5
(3 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	N	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 1 ms)

Abbildung 22 - UPDATE_3

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
25	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 0 ms)

UPDATE WARE SET WAREN_ID = WAREN_ID + 10 WHERE GEFAHRGUT = 'Y';
Update count: 1
(4 ms)

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID
147	E154	43	51	14	N	14
35	E554	4	5	15	Y	14
47	D33	550	55	114	N	24
7	F14	150	51	66	N	12
258	A548	400	79	76	N	2

(5 rows, 0 ms)

Abbildung 23 - UPDATE_4

Semesteraufgabe 6 [04.12.2024]:

1.1 Aufgabe:

In dieser Aufgabe ging es um die Anwendung von Aggregatfunktionen wie COUNT, SUM, AVG, MIN und MAX sowie deren Kombination mit GROUP BY, ORDER BY und HAVING.

1.2 Lösung:

Die folgenden Screenshots zeigen – wie in der Semesteraufgabe 5 – die Originaltabelle oben und die darunter liegende Tabelle mit der jeweils angewendeten Operation. Die Reihenfolge der Screenshots ist wie folgt:

1. COUNT(*)
2. COUNT(attr)
3. COUNT(DISTINCT attr)
4. SUM(attr)
5. AVG(attr)
6. MIN(attr), MAX(attr)
7. Kombinationen mit GROUP BY, ORDER BY, HAVING und Aggregatfunktionen

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

SELECT COUNT(*) AS Anzahl_Tupel FROM WARE;

ANZAHL_TUPEL
5

(1 row, 1 ms)

Abbildung 24 - COUNT

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 0 ms)

SELECT COUNT(BESTAND) AS Anzahl_Bestand FROM WARE;

ANZAHL_BESTAND
3

(1 row, 4 ms)

Abbildung 25 – COUNT_attr

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 0 ms)

SELECT COUNT(DISTINCT GEFAHRGUT) AS Anzahl_Gefahrgut FROM WARE;

ANZAHL_GEFAHRGUT
2

(1 row, 2 ms)

Abbildung 26 – COUNT_distinct

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 5 ms)

SELECT SUM(BESTAND) AS Gesamt_Bestand FROM WARE;

GESAMT_BESTAND
450

(1 row, 1 ms)

Abbildung 27 - SUM

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

SELECT AVG(BESTAND) AS Durchschnitt_Bestand FROM WARE;

DURCHSCHNITT_BESTAND
150.0

(1 row, 1 ms)

Abbildung 28 - AVG

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 0 ms)

SELECT MIN(Bestand) AS Min_Bestand, MAX(Bestand) AS Max_Bestand FROM WARE;

MIN_BESTAND	MAX_BESTAND
100	200

(1 row, 0 ms)

Abbildung 29 – MIN_MAX

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 0 ms)

```
SELECT GEFAHRGUT,
    COUNT(*) AS Anzahl_Waren,
    SUM(BESTAND) AS Gesamt_Bestand,
    AVG(CAST(GEWICHT AS DECIMAL(10, 2))) AS Durchschnitt_Gewicht,
    SUM(CASE WHEN BESTAND IS NULL THEN 1 ELSE 0 END) AS Anzahl_Null_Bestand
FROM WARE
WHERE GEFAHRGUT = 'N' GROUP BY GEFAHRGUT
HAVING COUNT(*) > 1 AND SUM(CASE WHEN BESTAND IS NULL THEN 1 ELSE 0 END) > 1 ORDER BY Durchschnitt_Gewicht DESC;
```

GEFAHRGUT	ANZAHL_WAREN	GESAMT_BESTAND	DURCHSCHNITT_GEWICHT	ANZAHL_NULL_BESTAND
N	4	250	67.500000000000	2

(1 row, 2 ms)

Abbildung 30 - MULTI_Abfrage

Semesteraufgabe 7 [11.12.2024]:

1.1 Aufgabe:

In der Semesteraufgabe 7 sollten verschiedene SQL-Operationen anhand von Testdaten ausgeführt werden. Dabei sind jeweils zwei Beispiele pro Operation zu erstellen. Der Aufbau der Aufgabe entspricht genau dem der letzten beiden Semesteraufgaben: Die obere Tabelle zeigt die Originaldaten, darunter folgt die angewendete Operation, und anschließend wird das Ergebnis dargestellt.

Die Screenshots sind genau in der Reihenfolge der aufgelisteten Operationen angeordnet, wobei pro Operation immer zwei Screenshots vorhanden sind.

- Mengenoperationen mit OR und AND
- Den natürlichen Verbund (JOIN)
- INNER JOIN
- Kartesisches Produkt (CROSS JOIN)
- Die Verwendung von AS
- Linker äußerer Join (LEFT OUTER JOIN)
- Rechter äußerer Join (RIGHT OUTER JOIN, RIGHT JOIN)

1.2 Lösung:

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

SELECT * FROM WARE WHERE GEFAHRGUT = 'Y' OR BESTAND > 130;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200

(2 rows, 1 ms)

Abbildung 31 - OR_1

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

SELECT * FROM WARE WHERE GRÖSSE < 50 OR VOLUMEN > 500;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*

(2 rows, 1 ms)

Abbildung 32 - OR_2

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

SELECT * FROM WARE WHERE GEFAHRGUT = 'N' AND BESTAND > 100;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150

(1 row, 0 ms)

Abbildung 33 - AND_1

SELECT * FROM WARE;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
147	E154	43	51	14	N	14	150
25	E554	4	5	15	Y	14	200
47	D33	550	55	114	N	24	*null*
7	F14	150	51	66	N	12	100
258	A548	400	79	76	N	2	*null*

(5 rows, 1 ms)

SELECT * FROM WARE WHERE VOLUMEN < 500 AND GEFAHRGUT = 'Y';

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND
25	E554	4	5	15	Y	14	200

(1 row, 1 ms)

Abbildung 34 - AND_2

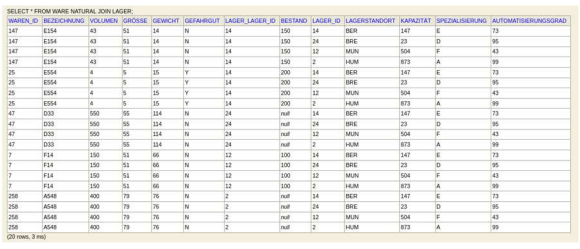


Abbildung 35 - JOIN_1

Abbildung 36 - JOIN_2

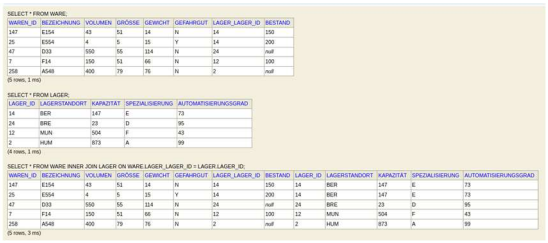


Abbildung 37 - INNER_JOIN_1

Abbildung 38 - INNER_JOIN_2

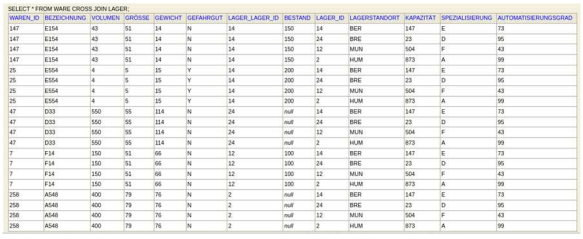


Abbildung 39 - CROSS_JOIN_1

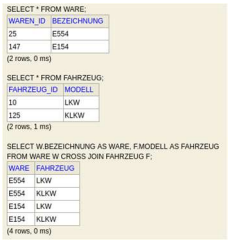


Abbildung 40 - CROSS_JOIN_2

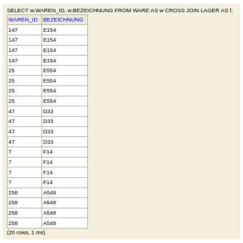


Abbildung 41 - AS_1

SELECT WAREN_ID AS Artikelnummer, BEZEICHNUNG AS Produktname FROM WARE;

ARTIKELNUMMER	PRODUKTNAME
147	E154
25	E554
47	D33
7	F14
258	A548

(5 rows, 0 ms)

Abbildung 42 - AS_2

SELECT WARE.*, LAGER.LAGER_ID FROM WARE LEFT OUTER JOIN LAGER ON WARE.LAGER_LAGER_ID = LAGER.LAGER_ID;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND	LAGER_ID
147	E154	43	51	14	N	14	150	14
25	E554	4	5	15	Y	14	200	14
47	D33	550	55	114	N	24	*null*	24
7	F14	150	51	66	N	12	100	12
258	A548	400	79	76	N	2	*null*	2

(5 rows, 1 ms)

Abbildung 43 - LEFT_OUTER_JOIN_1

Abbildung 44 - LEFT_OUTER_JOIN_2

SELECT WARE.*, LAGER.LAGER_ID FROM WARE RIGHT OUTER JOIN LAGER ON WARE.LAGER_LAGER_ID = LAGER.LAGER_ID;

WAREN_ID	BEZEICHNUNG	VOLUMEN	GRÖSSE	GEWICHT	GEFAHRGUT	LAGER_LAGER_ID	BESTAND	LAGER_ID
7	F14	150	51	66	N	12	100	12
147	E154	43	51	14	N	14	150	14
25	E554	4	5	15	Y	14	200	14
258	A548	400	79	76	N	2	*null*	2
47	D33	550	55	114	N	24	*null*	24

(5 rows, 2 ms)

Abbildung 45 - RIGHT_OUTER_JOIN_1

Abbildung 46 - RIGHT_OUTER_JOIN_2

Semesteraufgabe 8 [09.01.2025]:

1.1 Aufgabe:

Normalisierung bis zur 3. Normalform.

1.2 Lösung:

Um zur dritten Normalform zu kommen, fangen wir mit der "Nullten Normalform" an. (Dafür wurden Beispiel Tabellen erstellt.)

Wichtig hierbei ist, man kann nicht von der ersten in die dritte Normalform kommen. Es geht nur von der ersten in die zweite und dann in die Dritte.

1.2.1 Nullte Normalform:

- Bei einer Tabelle in der nullten Normalform befinden sich alle Daten unnormalisiert in einer Tabelle.

Das sieht dann wie auf der "Abbildung 47 - nullte_normalform" oder auch einfach das nächste Bild.

Fahrzeug_ID	Modell	Auftragsdatum	Lagerstandort	Lager_ID	Auftrags_ID	Kundenname
125	KLKW	SEP	BER	14	555	Heinz Müller
10	LKW	SEP	BRE	24	447	Fabian Steinhauer
43	LKW	JUN	MUN	12	4125	Max Krämer

Abbildung 47 – nullte_normalform

1.2.2 Erste Normalform:

- Bei einer Tabelle in der ersten Normalform müssen alle Attribute atomar sein.

Ein Attribut ist eine Spalte wie "Fahrzeug_ID" oder "Modell".

Und atomar bedeutet das man die Attribute nicht weiter aufgliedern kann.

In der nullten Normalform ist das noch nicht gegeben, wie man es zum Beispiel am Kundenname sehen kann. Da dort Vor- und Nachname in einer Spalte stehen. Man kann nur später nur nach Vor- und Nachname filtern wenn diese getrennt sind.

Das Ergebnis sehen wir in der "Abbildung 48 - erste_normalform".

Fahrzeug_ID	Modell	Auftragsdatum	Lagerstandort	Lager_ID	Auftrags_ID	Kundenvorname	Kundennachname
125	KLKW	SEP	BER	14	555	Heinz	Müller
10	LKW	SEP	BRE	24	447	Fabian	Steinhauer
43	LKW	JUN	MUN	12	4125	Max	Krämer

Abbildung 48 - erste_normalform

1.2.3 Zweite Normalform:

- Eine Tabelle befindet sich in der zweiten Normalform, wenn diese sich in der ersten Normalform befindet und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist.

  Also wird bei der zweiten Normalform aus der einen Tabelle mehrere mit "ober" Kategorien (Die roten Spalten sind die Primärschlüssel.). Wie in der "Abbildung 49 - zweite_normalform" zu sehen ist. Ab hier würde man auch noch die Beziehungen hinzufügen.

Fahrzeug

Fahrzeug_ID	Modell
125	KLKW
10	LKW
43	LKW

Auftrag

Auftrags_ID	Auftragsdatum
555	SEP
447	SEP
4125	JUN

Lager

Lager_ID	Lagerstandort
14	BER
24	BRE
12	MUN

Kunde

Kunden_ID	Kundenvorname	Kundennachname
514	Heinz	Müller
12	Fabian	Steinhauer
224	Max	Krämer

Abbildung 49 - zweite_normalform

1.2.4 Dritte Normalform:

- Eine Tabelle befindet sich genau dann in der dritten Normalform, wenn sie sich in der zweiten Normalform befindet und kein Nichtschlüsselattribut transitiv von einem Schlüsselkandidaten abhängt.

  Das bedeutet, wenn 3 kleiner als 5 ist und 5 kleiner als 8 ist dann ergibt sich transitiv das 3 kleiner als 8 ist.

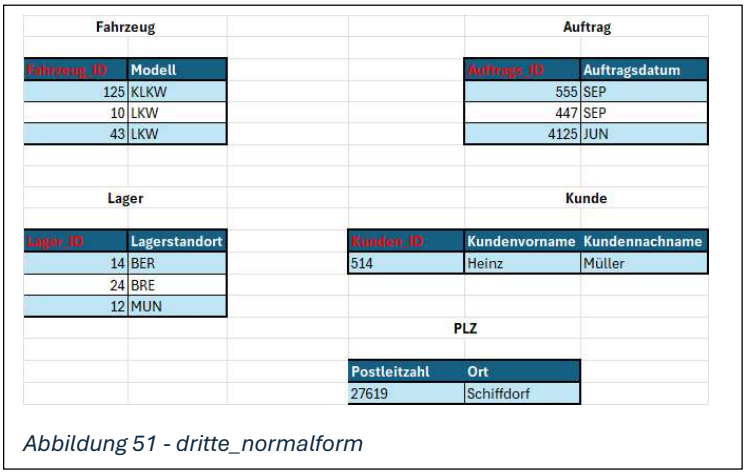
  Wenn man als Beispiel bei der Tabelle "Kunde" noch Postleitzahl und Ort hinzufügt, dann ist das Attribut "Ort" von der "Postleitzahl" abhängig und erst dann von dem Primärschlüssel. Die nicht Schlüsselattribute sind somit nicht voneinander unabhängig und das verstößt gegen die dritte Normalform. "Abbildung 50 - dritte_normalform_abhängigkeit".

Kunde

Kunden_ID	Kundenvorname	Kundennachname	Postleitzahl	Ort
514	Heinz	Müller	27619	Schiffdorf

Abbildung 50 - dritte_normalform_abhängigkeit

Die Transitiv abhängige Spalte "Ort" können wir somit in eine weitere Untertabelle ableiten da diese Spalte nur indirekt vom Schlüsselattribut abhängt. "Abbildung 51 - dritte_normalform".



Fahrzeug

Fahrzeug_ID	Modell
125	KLKW
10	LKW
43	LKW

Auftrag

Auftrags_ID	Auftragsdatum
555	SEP
447	SEP
4125	JUN

Lager

Lager_ID	Lagerstandort
14	BER
24	BRE
12	MUN

Kunde

Kunden_ID	Kundenvorname	Kundennachname
514	Heinz	Müller

PLZ

Postleitzahl	Ort
27619	Schiffdorf

Abbildung 51 - dritte_normalform

Damit ist die dritte Normalform erreicht.

Semesteraufgabe 9 [16.01.2025]:

1.1 Aufgabe:

Es sollten fünf virtuelle Views (Datensichten) erstellt werden. Dazu sind die entsprechenden Unterabfragen in SQL zu formulieren. Anschließend soll ein Screenshot hinzugefügt werden, der die Umsetzung der Abfragen zeigt.

1.2 Lösung:

Als erstes erstellen wir die fünf Views mit den Unterabfragen:

VIEW 1:

```
CREATE VIEW Waren_mit_Lagerstandort AS
SELECT
  W.WAREN_ID,
  W.BEZEICHNUNG,
  W.BESTAND,
  (SELECT L.LAGERSTANDORT FROM LAGER L WHERE L.LAGER_ID = W.LAGER_LAGER_ID) AS Lagerstandort
FROM WARE W;
```

VIEW 2:

```
CREATE VIEW Lagerbestand_Durchschnitt AS
SELECT
  L.LAGERSTANDORT,
  COUNT(W.WAREN_ID) AS Anzahl_Waren,
  AVG(CAST(W.GRÖSSE AS DECIMAL)) AS Durchschnittliche_Groesse
FROM WARE W
JOIN LAGER L ON W.LAGER_LAGER_ID = L.LAGER_ID
GROUP BY L.LAGERSTANDORT;
```

VIEW 3:

```
CREATE VIEW Fahrzeuge_mit_Auftrag AS
SELECT
  F.FAHRZEUG_ID,
  F.MODELL,
  F.HERSTELLER,
(SELECT T.ZIELORT FROM TRANSPORTAUFTRAG T WHERE T.AUFTRAGS_ID =
F.TRANSPORTAUFTRAG_AUFTRAGS_ID) AS Zielort
FROM FAHRZEUG F;
```

VIEW 4:

```
CREATE VIEW Waren_ueber_Durchschnitt AS
SELECT
  W.WAREN_ID,
  W.BEZEICHNUNG,
  W.BESTAND
FROM WARE W
WHERE W.BESTAND > (SELECT AVG(BESTAND) FROM WARE);
```

VIEW 5:

```
CREATE VIEW Offene_Lagerkapazitaet AS
SELECT
  L.LAGER_ID,
  L.LAGERSTANDORT,
  L.KAPAZITÄT - COALESCE(
      (SELECT SUM(CAST(W.VOLUMEN AS DECIMAL)) FROM WARE W WHERE W.LAGER_LAGER_ID =
L.LAGER_ID), 0) AS Freie_Kapazität
FROM LAGER L;
```

Eine kleine Erklärung zu den Views:

1. **Waren_mit_Lagerstandort** → Zeigt jede Ware zusammen mit dem Lagerstandort, in dem sie sich befindet.
2. **Lagerbestand_Durchschnitt** → Zeigt, wie viele Waren pro Lager sind und berechnet die durchschnittliche Größe der Waren.
3. **Fahrzeuge_mit_Auftrag** → Zeigt alle Fahrzeuge mit den Zielorten ihrer zugehörigen Transportaufträge.
4. **Waren_ueber_Durchschnitt** → Listet alle Waren auf, deren Bestand über dem durchschnittlichen Bestand aller Waren liegt.
5. **Offene_Lagerkapazität** → Berechnet, wie viel Platz in jedem Lager noch verfügbar ist, indem belegtes Volumen von der Kapazität abgezogen wird.

Nun können wir mit den dazugehörigen “SELECT“ die Daten ausgeben:

SELECT * FROM FAHRZEUGE_MIT_AUFTRAG ;

FAHRZEUG_ID	MODELL	HERSTELLER	ZIELORT
125	KLKW	Scania	FAM
10	LKW	Scania	BRE
43	LKW	MAN	HAM

(3 rows, 2 ms)

SELECT * FROM LAGERBESTAND_DURCHSCHNITT;

LAGERSTANDORT	ANZAHL_WAREN	DURCHSCHNITTLICHE_GROESSE
BER	2	28
BRE	1	55
HUM	1	79
MUN	1	51

(4 rows, 11 ms)

SELECT * FROM OFFENE_LAGERKAPAZITAET ;

LAGER_ID	LAGERSTANDORT	FREIE_KAPAZITÄT
14	BER	100
24	BRE	-527
12	MUN	354
2	HUM	473

(4 rows, 6 ms)

SELECT * FROM WAREN_MIT_LAGERSTANDORT;

WAREN_ID	BEZEICHNUNG	BESTAND	LAGERSTANDORT
147	E154	150	BER
25	E554	200	BER
47	D33	*null*	BRE
7	F14	100	MUN
258	A548	*null*	HUM

(5 rows, 5 ms)

SELECT * FROM WAREN_UEBER_DURCHSCHNITT;

WAREN_ID	BEZEICHNUNG	BESTAND
25	E554	200

(1 row, 3 ms)

Abbildung 52 - view_output