Handout
Themenfeld: Datenbanken und SQL
Abschnitt: 11.01.01. Abfragen über mehrere Tabellen - Fortsetzung

Autor: Thomas Krause
Stand: 14.11.2022 12:04:00

Inhalt

1 Vorbereitung und Übersicht

Vorbereitung:

Installieren Sie bei Bedarf die mitgelieferte Datenbank über folgendes Skript:

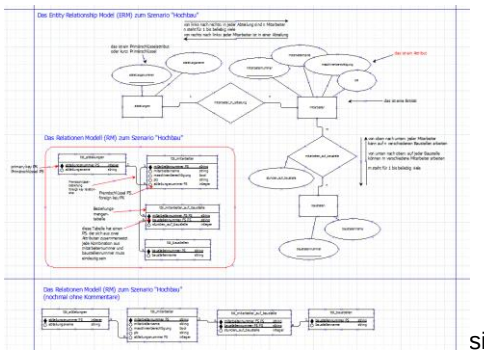
create_database_hochbau_MUSTERDATENBANK.sql

Szenario:

Der Vortrag bezieht sich auf das folgende, bereits gemeinsam bearbeitete Szenarion. Downloaden Sie bei Bedarf nochmals die Dateien bzw. installieren Sie sich die Datenbank über das mitgelieferte Skript.

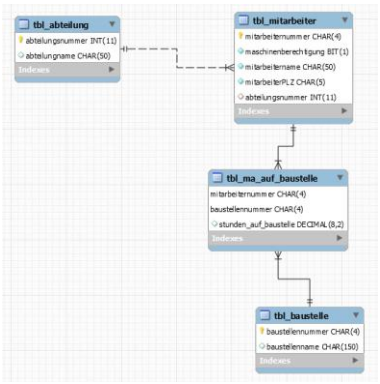
DIA:

siehe Dateien



siehe Datei:
database_hochbau_MUSTER_DB_ERM_RM.dia

Workbench:



Übersicht:

- Informationen werden in verschiedenen Tabellen gespeichert
- das Tabellen-Modell wird durch Anwendung der Normalisierungsregeln (NF1 … NF3) erstellt und enthält Tabellen mit Spalten, Primärschlüsseln und Fremdschlüsselbeziehungen
- die Normalisierung ist notwendig, um eine redundanzfreie/ redundanzarme und effiziente Speicherung zu erreichen
- die zusammengehörigen Informationen werden bei Abfragen und Verarbeitung wieder zusammengeführt, indem über mehrere Tabellen abgefragt wird

2 Abfragen über 2 Tabellen mit LEFT OUTER JOIN und RIGHT OUTER JOIN

Aufgabe/ Beispiel:

- die über Normalisierung auf mehrere Tabellen verteilten Informationen werden in einer SQL-Abfrage zusammengeführt
- im Beispiel: Liste mit Namen der Mitarbeiter und Namen der Abteilungen, in denen sie arbeiten; es sollen auch zusätzlich die Mitarbeiter angezeigt werden, die noch keiner Abteilung zugeordnet sind

Ausgangstabellen:

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
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
21	Vertrieb
22	Lager und Logistik

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungname
M009	Örtel	Hochbau
M010	Stein	Ausbau
M021	Hahn	Haustechnik
M024	Holzer	Hochbau
M113	Latte	NULL
M117	Fahrland	NULL
M118	Putz	NULL

Syntax:

```
SELECT *
FROM <name_tab1> LEFT [OUTER] JOIN <name_tab2> ON [<name_tab1>.]<feld_tab1> = [<name_tab2>.]<feld_tab2>
```

```
SELECT
        <feld>,
        <feld>,
        [<name_tab1>.]<feld>
        [<name_tab2>.]<feld>
FROM <name_tab1> LEFT [OUTER] JOIN <name_tab2> ON [<name_tab1>.]<feld_tab1> = [<name_tab2>.]<feld_tab2>
```

Ausgangstabellen: siehe oben

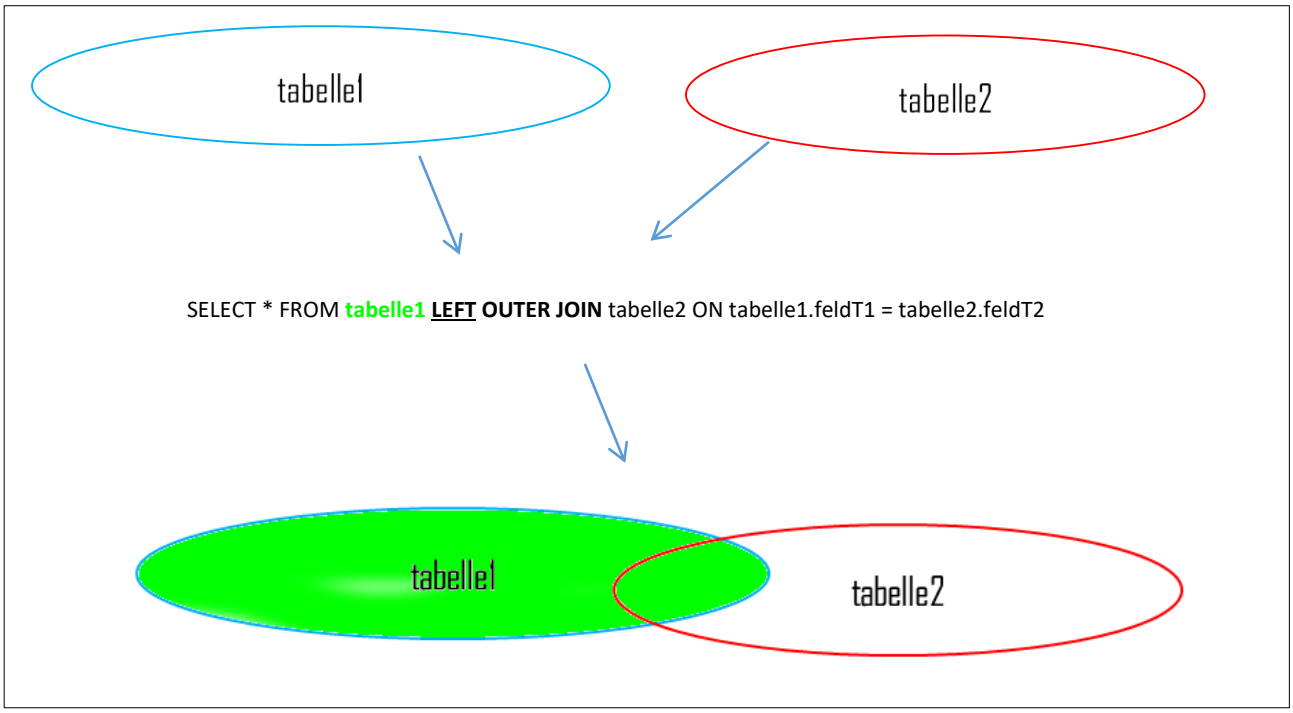
Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungname
M009	Örtel	Hochbau
M010	Stein	Ausbau
M021	Hahn	Haustechnik
M024	Holzer	Hochbau
M113	Latte	NULL
M117	Fahrland	NULL
M118	Putz	NULL

```
select
      mitarbeiternummer,
      mitarbeitername,
      abteilungname
from tbl_mitarbeiter left outer join tbl_abteilung
on tbl_mitarbeiter.abteilungsnummer = tbl_abteilung.abteilungsnummer;
```

BEACHTE:

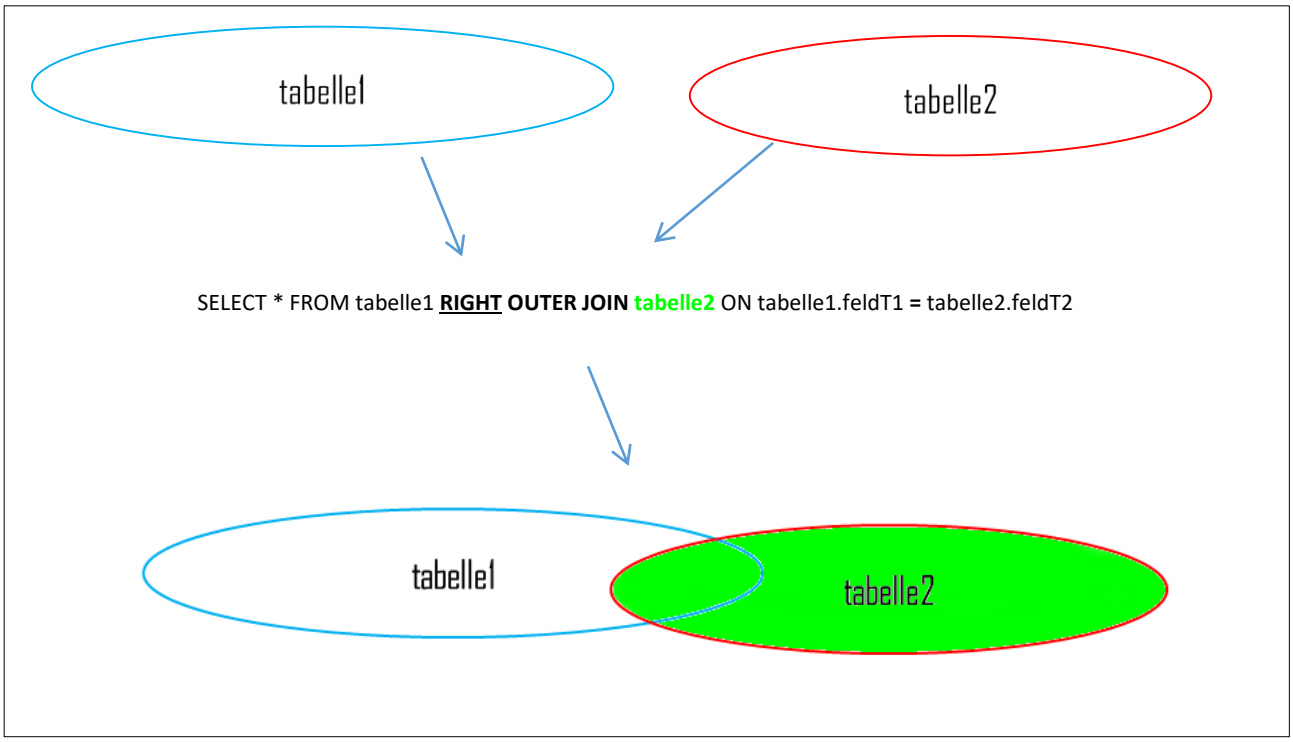
- bei vielen DBMS kann das Schlüsselwort OUTER entfallen. Wenn nur LEFT JOIN verwendet wird, wird das automatisch als LEFT OUTER JOIN interpretiert



weitere Erläuterungen:

- **Outer** Join es werden **alle** Datensätze zurückgeliefert, auch wenn es keinen verknüpften aus der zweiten Tabelle gibt (dort dann leerer Datensatz), Unterscheidung in LEFT OUTER JOIN und RIGHT OUTER JOIN, je nachdem, aus welcher Tabelle **alle** Datensätze geliefert werden sollen

- LEFT und RIGHT geben an, von welcher Tabelle alle Datensätze geliefert werden (Position der Tabellen zum Befehl JOIN in der Anweisung)



weitere Erläuterungen:

LEFT und RIGHT können wechselseitig ineinander umgewandelt werden
Anordnung der Tabellen zum JOIN-Befehl ändern --> Seiten wechseln

praktische Anwendung/ Wirkungsweise/ Üben:

Beispiel 1:

- im Beispiel: Liste mit Namen der Mitarbeiter und Namen der Abteilungen, in denen sie arbeiten; es sollen auch zusätzlich die Mitarbeiter angezeigt werden, die noch keiner Abteilung zugeordnet sind

Ausgangstabellen:

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
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
21	Vertrieb
22	Lager und Logistik

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungname
M009	Örtel	Hochbau
M010	Stein	Ausbau
M021	Hahn	Haustechnik
M024	Holzer	Hochbau
M113	Latte	NULL
M117	Fahrland	NULL
M118	Putz	NULL

```
select
	mitarbeiternummer,
	mitarbeitername,
	abteilungname
from tbl_mitarbeiter left outer join tbl_abteilung
on tbl_mitarbeiter.abteilungsnummer = tbl_abteilung.abteilungsnummer;
```

Beispiel 2:

- im Beispiel: Liste mit Namen der Mitarbeiter und Namen der Abteilungen, in denen sie arbeiten; es sollen auch zusätzlich die Mitarbeiter angezeigt werden, die noch keiner Abteilung zugeordnet sind

Ausgangstabellen:

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
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
21	Vertrieb
22	Lager und Logistik

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungname
M009	Örtel	Hochbau
M010	Stein	Ausbau
M021	Hahn	Haustechnik
M024	Holzer	Hochbau
M113	Latte	NULL
M117	Fahrland	NULL
M118	Putz	NULL

```
select
	mitarbeiternummer,
	mitarbeitername,
	abteilungname
from tbl_abteilung right join tbl_mitarbeiter
on tbl_mitarbeiter.abteilungsnummer = tbl_abteilung.abteilungsnummer;
```

BEACHTE:

- die Ergebnisliste ist die gleiche wie im vorangegangenen Beispiel
- es wurde jetzt ein RIGHT JOIN verwendet und die Tabellen haben ihre Positionen getauscht
- das Schlüsselwort OUTER kann entfallen

Beispiel 3:

- im Beispiel: Liste mit Namen der Mitarbeiter und Namen der Abteilungen, in denen sie arbeiten; es sollen auch zusätzlich die Abteilungen angezeigt werden, denen noch keine Mitarbeiter zugeordnet wurden

Ausgangstabellen:

- wie in Beispiel 2

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungsnummer	abteilungname
M009	Örtel	9	Hochbau
M024	Holzer	9	Hochbau
M021	Hahn	10	Haustechnik
M010	Stein	12	Ausbau
NULL	NULL	21	Vertrieb
NULL	NULL	22	Lager und Logistik

VARIANTE 1:

```
select
	mitarbeiternummer,
	mitarbeitername,
	tbl_abteilung.abteilungsnummer,
	abteilungname
from tbl_mitarbeiter right join tbl_abteilung
on tbl_mitarbeiter.abteilungsnummer = tbl_abteilung.abteilungsnummer;
```

VARIANTE 2:

```
select
	mitarbeiternummer,
	mitarbeitername,
	tbl_abteilung.abteilungsnummer,
	abteilungname
from tbl_abteilung left join tbl_mitarbeiter
on tbl_mitarbeiter.abteilungsnummer = tbl_abteilung.abteilungsnummer;
```

3 Abfragen über 2 Tabellen mit FULL OUTER JOIN

Aufgabe/ Zweck/ Beispiel/ Demonstrieren:

Liste mit Mitarbeitern und Abteilungen; dabei soll angezeigt werden, welcher Mitarbeiter in welcher Abeilung arbeitet; darüber hinaus sollen zusätzlich die Mitarbeiter ohne Abteilung und die Abteilungen ohne Mitarbeiter angezeigt werden

Ausgangstabellen:

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
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
21	Vertrieb
22	Lager und Logistik

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungsnummer	abteilungname
M009	Örtel	9	Hochbau
M010	Stein	12	Ausbau
M021	Hahn	10	Haustechnik
M024	Holzer	9	Hochbau
M113	Latte	NULL	NULL
M117	Fahrland	NULL	NULL
M118	Putz	NULL	NULL
NULL	NULL	21	Vertrieb
NULL	NULL	22	Lager und Logistik

(Achtung: screenshot ist NICHT aus MySQL)

mitarbeiternummer	mitarbeitername	abteilungsnummer	abteilungname
M009	Örtel	9	Hochbau
M010	Stein	12	Ausbau
M021	Hahn	10	Haustechnik
M024	Holzer	9	Hochbau
M113	Latte	NULL	NULL
M117	Fahrland	NULL	NULL
M118	Putz	NULL	NULL
NULL	NULL	21	Vertrieb
NULL	NULL	22	Lager und Logistik

(screenshot aus MySQL)

Syntax/ Aufbau/ Erläutern:

```
SELECT
       <feld>,
       <feld>,
       [<name_tab1>.]<feld>
       [<name_tab2>.]<feld>
FROM <name_tab1> FULL [OUTER] JOIN <name_tab2>
ON [<name_tab1>.]<feld_tab1> = [<name_tab2>.]<feld_tab2>;
```

BEACHTE:

- MySQL unterstützt kein FULL OUTER JOIN
- siehe auch u.a.:
  https://ubiq.co/database-blog/how-to-do-a-full-outer-join-in-mysql/ (abgerufen: 2022-01-17)
  https://www.peterkropff.de/site/mysql/full_outer_join.htm (abgerufen: 2022-01-17)
- in MySQL muss ein FULL OUTER JOIN zusammengesetzt werden aus einem LEFT JOIN und einem RIGHT JOIN

Syntax für MySQL:

```
SELECT
       <feld>,
       <feld>,
       [<name_tab1>.]<feld>
       [<name_tab2>.]<feld>
FROM <name_tab1> LEFT JOIN <name_tab2>
ON [<name_tab1>.]<feld_tab1> = [<name_tab2>.]<feld_tab2>
UNION
SELECT
       <feld>,
       <feld>,
       [<name_tab1>.]<feld>
       [<name_tab2>.]<feld>
FROM <name_tab1> RIGHT JOIN <name_tab2>
ON [<name_tab1>.]<feld_tab1> = [<name_tab2>.]<feld_tab2>;
```

Erläuterung:

- Liste mit Mitarbeitern und Abteilungen; dabei soll angezeigt werden, welcher Mitarbeiter in welcher Abeilung arbeitet; darüber hinaus sollen zusätzlich die Mitarbeiter ohne Abteilung und die Abteilungen ohne Mitarbeiter angezeigt werden
- da MySQL kein FULL OUTER JOIN unterstützt, wurde die Abfrage mit 2 Abfragen und einem UNION nachgebildet → Achtung: durch Weglassen der Option ALL (bei UNION) werden doppelte Datensätze in der Ergebnisliste unterdrückt

Ausgangstabellen:

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
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

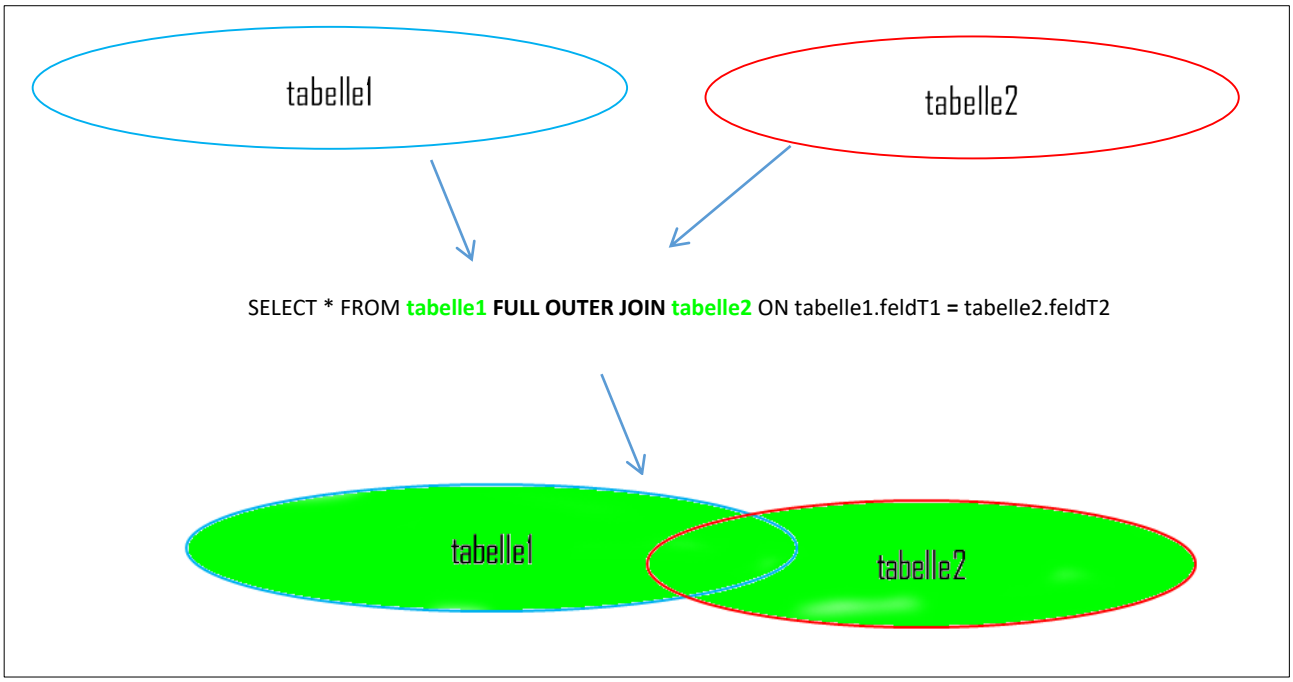
abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
21	Vertrieb
22	Lager und Logistik

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungsnummer	abteilungname
M009	Örtel	9	Hochbau
M010	Stein	12	Ausbau
M021	Hahn	10	Haustechnik
M024	Holzer	9	Hochbau
M113	Latte	NULL	NULL
M117	Fahrland	NULL	NULL
M118	Putz	NULL	NULL
NULL	NULL	21	Vertrieb
NULL	NULL	22	Lager und Logistik

(screenshot aus MySQL)

```
select
	mitarbeiternummer,
	mitarbeitername,
	tbl_abteilung.abteilungsnummer,
	abteilungname
from tbl_mitarbeiter left join tbl_abteilung
on tbl_mitarbeiter.abteilungsnummer = tbl_abteilung.abteilungsnummer
UNION
select
	mitarbeiternummer,
	mitarbeitername,
	tbl_abteilung.abteilungsnummer,
	abteilungname
from tbl_mitarbeiter right join tbl_abteilung
on tbl_mitarbeiter.abteilungsnummer = tbl_abteilung.abteilungsnummer;
```



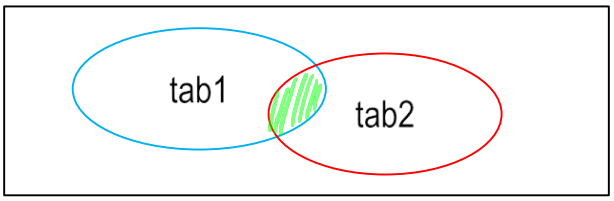
weitere Erläuterungen:

- es werden alle Datensätze beider Tabellen gezeigt UND die Datensätze, für die die Verknüpfungsbedingung WAHR ist, werden miteinander verbunden angezeigt
- heißt auch „Vereinigungsmenge“
- es können auch verwendet werden: WHERE, ORDER BY, GROUP BY
- die verknüpften Felder müssen den gleichen Datentyp haben (eventuell anpassen über Befehl CAST)

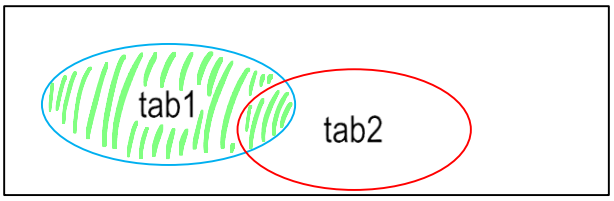
praktische Anwendung/ Wirkungsweise/ Üben:

N/A

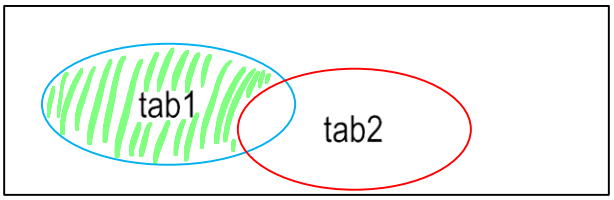
4 Zusammenfassung bis hierin



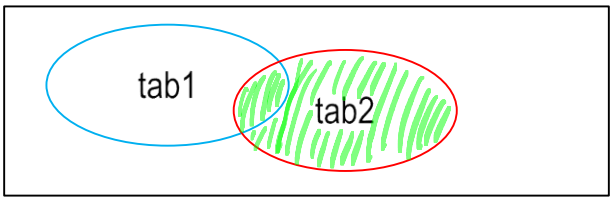

```
SELECT <spaltenliste> FROM <tab1> INNER JOIN <tab2> ON <tab1>.<key> = <tab2>.<key>
SELECT <spaltenliste> FROM <tab1>, <tab2> WHERE <tab1>.<key> = <tab2>.<key>
```



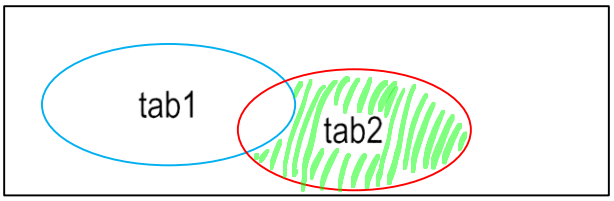

```
SELECT <spaltenliste> FROM <tab1> LEFT JOIN <tab2> ON <tab1>.<key> = <tab2>.<key>
```



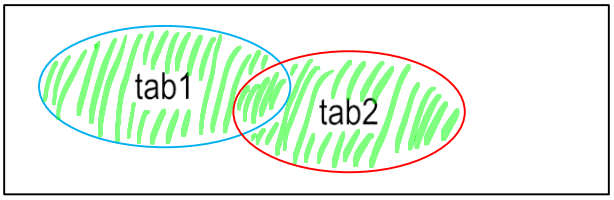

```
SELECT <spaltenliste> FROM <tab1> LEFT JOIN <tab2> ON <tab1>.<key> = <tab2>.<key>
WHERE <tab2>.<key> IS NULL
```



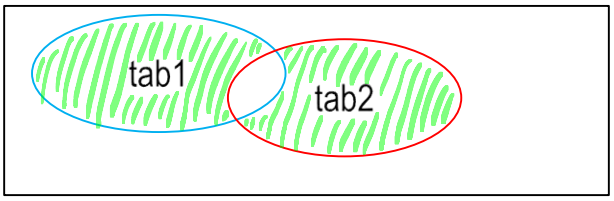

```
SELECT <spaltenliste> FROM <tab1> RIGHT JOIN <tab2> ON <tab1>.<key> = <tab2>.<key>
```




```
SELECT <spaltenliste> FROM <tab1> RIGHT JOIN <tab2> ON <tab1>.<key> = <tab2>.<key>
WHERE <tab1>.<key> IS NULL
```




```
SELECT <spaltenliste> FROM <tab1> FULL OUTER JOIN <tab2> ON <tab1>.<key> = <tab2>.<key>
```




```
SELECT <spaltenliste> FROM <tab1> FULL OUTER JOIN <tab2> ON <tab1>.<key> = <tab2>.<key>
WHERE <tab1>.<key> IS NULL OR <tab2>.<key> IS NULL
```

5 Abfragen über 2 Tabellen mit Kartesischem Produkt

Aufgabe/ Zweck/ Beispiel/ Demonstrieren:

Erzeugung einer Ergebnisliste, bei der alle Datensätze der einen Tabelle mit allen Datensätzen der anderen Tabellle kombiniert werden.

Ausgangstabellen:

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
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
21	Vertrieb
22	Lager und Logistik

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungsnummer	abteilungsnummer	abteilungname
M009	Örtel	9	9	Hochbau
M009	Örtel	9	10	Haustechnik
M009	Örtel	9	12	Ausbau
M009	Örtel	9	21	Vertrieb
M009	Örtel	9	22	Lager und Logistik
M010	Stein	12	9	Hochbau
M010	Stein	12	10	Haustechnik
M010	Stein	12	12	Ausbau
M010	Stein	12	21	Vertrieb
M010	Stein	12	22	Lager und Logistik
M021	Hahn	10	9	Hochbau
M021	Hahn	10	10	Haustechnik
M021	Hahn	10	12	Ausbau
M021	Hahn	10	21	Vertrieb
M021	Hahn	10	22	Lager und Logistik
M024	Holzer	9	9	Hochbau
M024	Holzer	9	10	Haustechnik
M024	Holzer	9	12	Ausbau
M024	Holzer	9	21	Vertrieb
M024	Holzer	9	22	Lager und Logistik

(… die Liste geht noch weiter, hat insgesamt 35 Zeilen)

Syntax/ Aufbau/ Erläutern:

Möglichkeit 1:

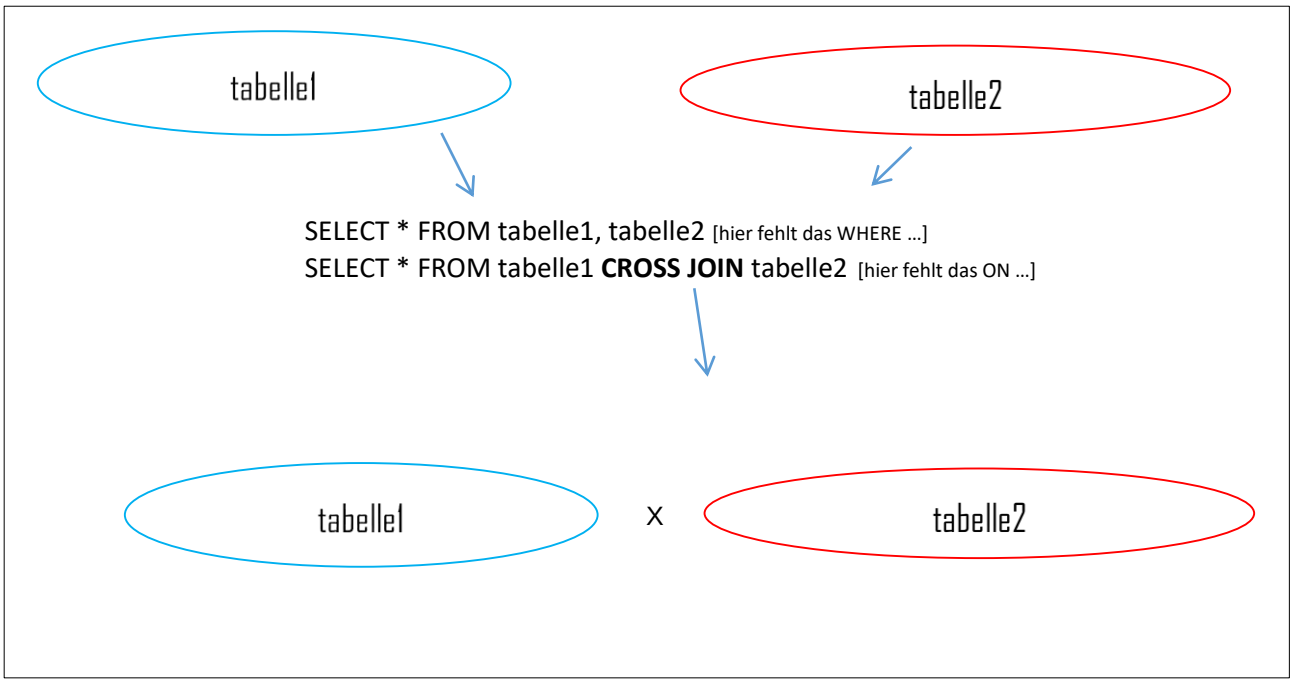
```
SELECT
       <feld>,
       <feld>,
       [<name_tab1>.]<feld>
       [<name_tab2>.]<feld>
FROM <name_tab1> CROSS JOIN <name_tab2>;
```

Möglichkeit 2:

```
SELECT
       <feld>,
       <feld>,
       [<name_tab1>.]<feld>
       [<name_tab2>.]<feld>
FROM <name_tab1>, <name_tab2>;
```

BEACHTE:

- die Anzahl der Zeilen in der Ergebnisliste ist das Produkt aus Zeilenanzahl tab1 und Zeilenanzahl tab2
- Möglichkeit 2 kann häufig ungewollt durch eine unvollständige SQL-Anweisung entstehen (fehlendes WHERE …)



SELECT * FROM tabelle1, tabelle2 [hier fehlt das WHERE …]
SELECT * FROM tabelle1 **CROSS JOIN** tabelle2 [hier fehlt das ON …]

Erläuterung:

- jedes Tabellenfeld der tabelle1 wird mit jedem Tabellenfeld der tabelle2 kombiniert aka: Kreuz-Produkt, kartesisches Produkt
- die Anzahl der Datensätze in der Ergebnismenge ist: anzahl_datensätze$_{tabelle1}$ * anzahl_datensätze$_{tablelle2}$
- es können auch verwendet werden:
  WHERE, ORDER BY
- Anwendung: ?

praktische Anwendung/ Wirkungsweise/ Üben:

Beispiel 1:

Ausgangstabellen:

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
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

abteilungsnummer	abteilungname
9	Hochbau
10	Haustechnik
12	Ausbau
21	Vertrieb
22	Lager und Logistik

Ergebnisliste:

mitarbeiternummer	mitarbeitername	abteilungsnummer	abteilungsnummer	abteilungname
M009	Örtel	9	9	Hochbau
M009	Örtel	9	10	Haustechnik
M009	Örtel	9	12	Ausbau
M009	Örtel	9	21	Vertrieb
M009	Örtel	9	22	Lager und Logistik
M010	Stein	12	9	Hochbau
M010	Stein	12	10	Haustechnik
M010	Stein	12	12	Ausbau
M010	Stein	12	21	Vertrieb
M010	Stein	12	22	Lager und Logistik
M021	Hahn	10	9	Hochbau
M021	Hahn	10	10	Haustechnik
M021	Hahn	10	12	Ausbau
M021	Hahn	10	21	Vertrieb
M021	Hahn	10	22	Lager und Logistik
M024	Holzer	9	9	Hochbau
M024	Holzer	9	10	Haustechnik
M024	Holzer	9	12	Ausbau
M024	Holzer	9	21	Vertrieb
M024	Holzer	9	22	Lager und Logistik

(… die Liste geht noch weiter, hat insgesamt 35 Zeilen)

Möglichkeit 1:

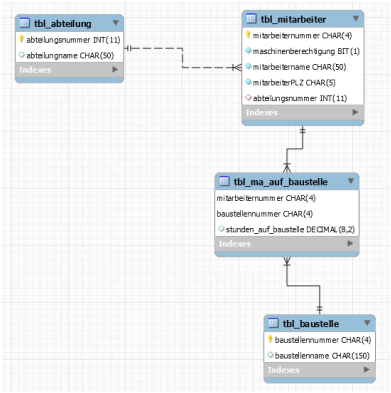
```
select
	mitarbeiternummer,
	mitarbeitername,
	tbl_mitarbeiter.abteilungsnummer,
	tbl_abteilung.abteilungsnummer,
	abteilungname
from tbl_mitarbeiter cross join tbl_abteilung;
```

Möglichkeit 2:

```
select
	mitarbeiternummer,
	mitarbeitername,
	tbl_mitarbeiter.abteilungsnummer,
	tbl_abteilung.abteilungsnummer,
	abteilungname
from tbl_mitarbeiter, tbl_abteilung;
```

6 Abfragen über mehr als 2 Tabellen

Hier nochmal die Ausgangslage:



Syntax:

Möglichkeit 1:

```
SELECT <liste_der_spaltennamen>
        FROM <tabelle1>
                JOIN <tabelle2> ON  <tabelle1.fremdschlüsselfeld> = <tabelle2.primärschlüsselfeld>
                JOIN <tabelle3> ON  <tabelle2.fremdschlüsselfeld> = <tabelle3.primärschlüsselfeld>
                [JOIN <tabelleN> ON  <tabelleM.fremdschlüsselfeld> = <tabelleN.primärschlüsselfeld>]
```

Möglichkeit 2:

```
SELECT <liste_der_spaltennamen>
        FROM <tabelle1>, <tabelle2>, <tabelle3>[, <tabelleN>]
        WHERE    <tabelle1.fremdschlüsselfeld> = <tabelle2.primärschlüsselfeld> AND
                 <tabelle2.fremdschlüsselfeld> = <tabelle3.primärschlüsselfeld> [AND
                 <tabelleM.fremdschlüsselfeld> = <tabelleN.primärschlüsselfeld>]
```

praktische Anwendung:

Liste mit allen Mitarbeitern, Stunden auf Baustellen und Baustellen. Dabei sollen die Namen der Mitarbeiter und die Namen der Baustellen angezeigt werden:

mitarbeiternummer	mitarbeitername	stunden_auf_baustelle	baustellennummer	baustellenname
M009	Örtel	37.00	B253	GaleriaX
M010	Stein	12.00	B021	MIDL
M010	Stein	23.00	B112	Kaufstadt
M021	Hahn	21.00	B056	Brutto
M021	Hahn	24.00	B112	Kaufstadt
M021	Hahn	34.00	B253	GaleriaX
M024	Holzer	8.00	B056	Brutto
M024	Holzer	24.00	B253	GaleriaX

Möglichkeit 1:

```
-- Variante 1
select
          tbl_mitarbeiter.mitarbeiternummer,
          mitarbeitername,
          stunden_auf_baustelle,
          tbl_baustelle.baustellennummer,
          baustellenname
from tbl_mitarbeiter
          join tbl_ma_auf_baustelle on tbl_mitarbeiter.mitarbeiternummer = tbl_ma_auf_baustelle.mitarbeiternummer
          join tbl_baustelle on tbl_ma_auf_baustelle.baustellennummer = tbl_baustelle.baustellennummer;


-- Variante 2 mit Aliassen für die Tabellen
select
          m.mitarbeiternummer,
          mitarbeitername,
          stunden_auf_baustelle,
          b.baustellennummer,
          baustellenname
from tbl_mitarbeiter m
          join tbl_ma_auf_baustelle mb on m.mitarbeiternummer = mb.mitarbeiternummer
          join tbl_baustelle b on mb.baustellennummer = b.baustellennummer;
```

Möglichkeit 2:

```
-- Variante 1
select
        tbl_mitarbeiter.mitarbeiternummer,
        mitarbeitername,
        stunden_auf_baustelle,
        tbl_baustelle.baustellennummer,
        baustellenname
from tbl_mitarbeiter, tbl_ma_auf_baustelle, tbl_baustelle
where   tbl_mitarbeiter.mitarbeiternummer = tbl_ma_auf_baustelle.mitarbeiternummer AND
        tbl_ma_auf_baustelle.baustellennummer = tbl_baustelle.baustellennummer;


-- Variante 2 mit Aliassen für die Tabellen
select
        m.mitarbeiternummer,
        mitarbeitername,
        stunden_auf_baustelle,
        b.baustellennummer,
        baustellenname
from tbl_mitarbeiter m, tbl_ma_auf_baustelle mb, tbl_baustelle b
where   m.mitarbeiternummer = mb.mitarbeiternummer AND
        mb.baustellennummer = b.baustellennummer;
```