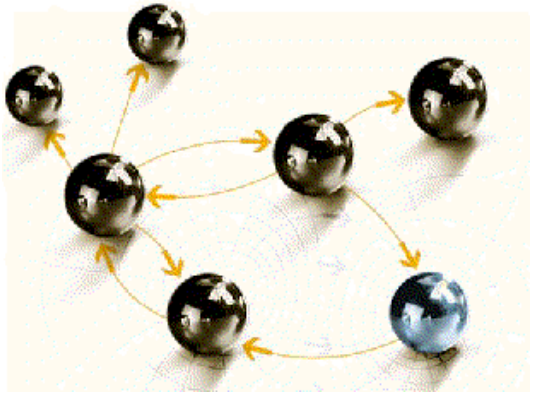


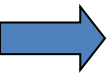
cps4it

DB2 for z/OS

Teil 2 – SQL

cps4it
consulting, projektmanagement und seminare für die informationstechnologie
Ralf Seidler, Stromberger Straße 36A, 55411 Bingen
Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: ralf.seidler@cps4it.de
Internet: http://www.cps4it.de

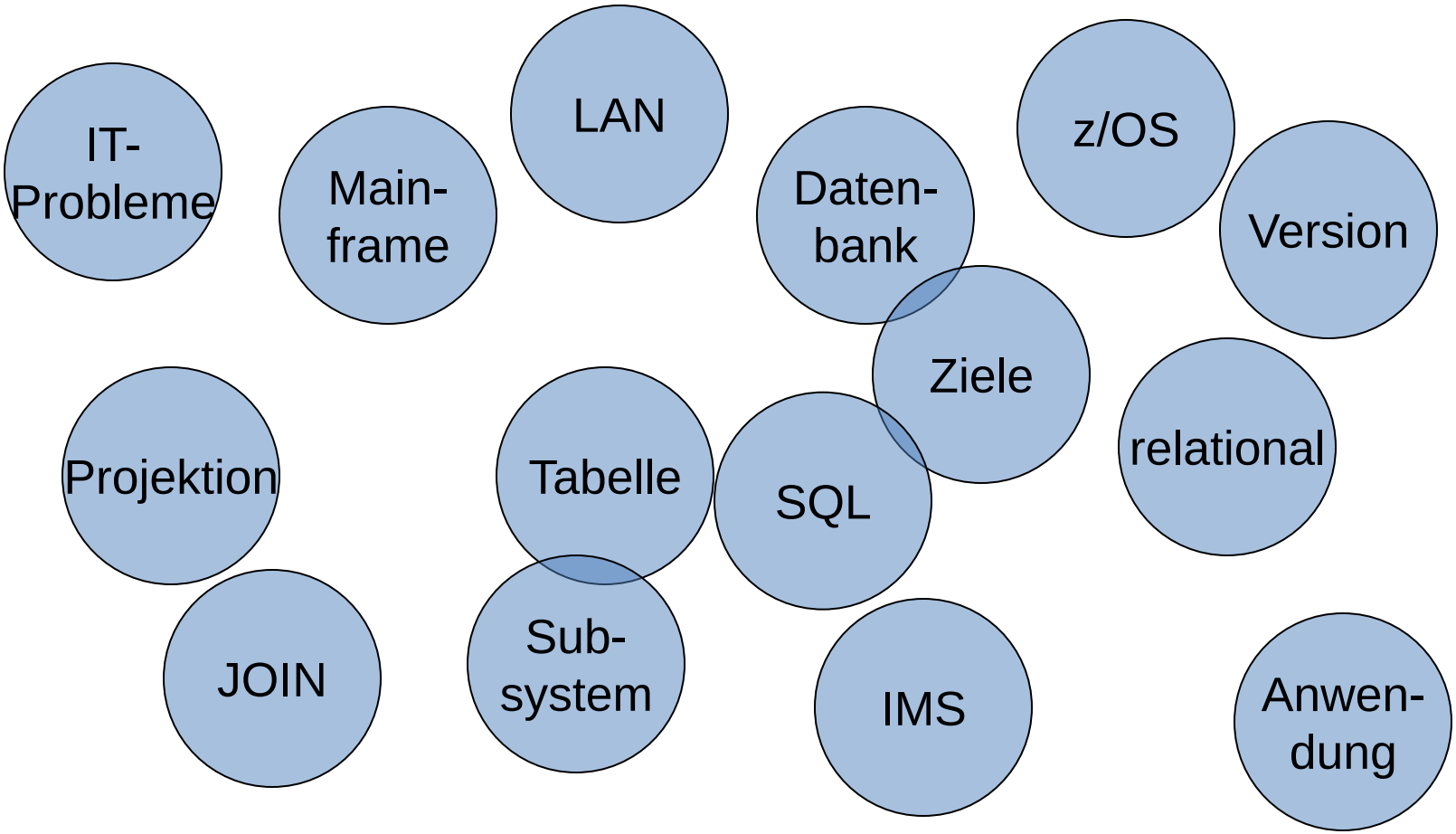
Inhalt



- Überblick zum Teil 1
- Abfragen auf 1 Tabelle
- Verknüpfung von Abfragen
- Verschachtelung und Funktionen
- Ändern von Tabellen
- Benutzersicht – View
- Abfrageparameter in Auswahl

Überblick zum Teil 1

Begriffe



Überblick zum Teil 1

Literaturhinweise

- Bookmanager im Internet
- pdf-Dokumente im Internet
  - teilweise in Deutsch
- Bücher
- Bookmanager im Intranet
- pdf-Dokumente im Intranet

Überblick zum Teil 1

Datenbank-Design

- gutes Design
- schlechtes Design
- Relationenmodell
- Primärschlüssel
- Beziehungen zwischen Tabellen
- Fremdschlüssel
- Normalisierung und Konsistenzregeln

Überblick zum Teil 1

Beispieldatenbank

- Definition
- Inhalte

Überblick zum Teil 1

Datendefinitionen

- die Tabelle und ihre Datenformate
- erstellen einer Tabelle
- Integritätsprüfungen
- NULL bzw. NOT NULL
- erweitern und löschen
- Synonym
- Index

Überblick zum Teil 1

Speicherstruktur

- DB2-Objekte
- Tablespace
- Database
- Storagegroup

Überblick zum Teil 1

interaktives Arbeiten mit DB2 – DB2I

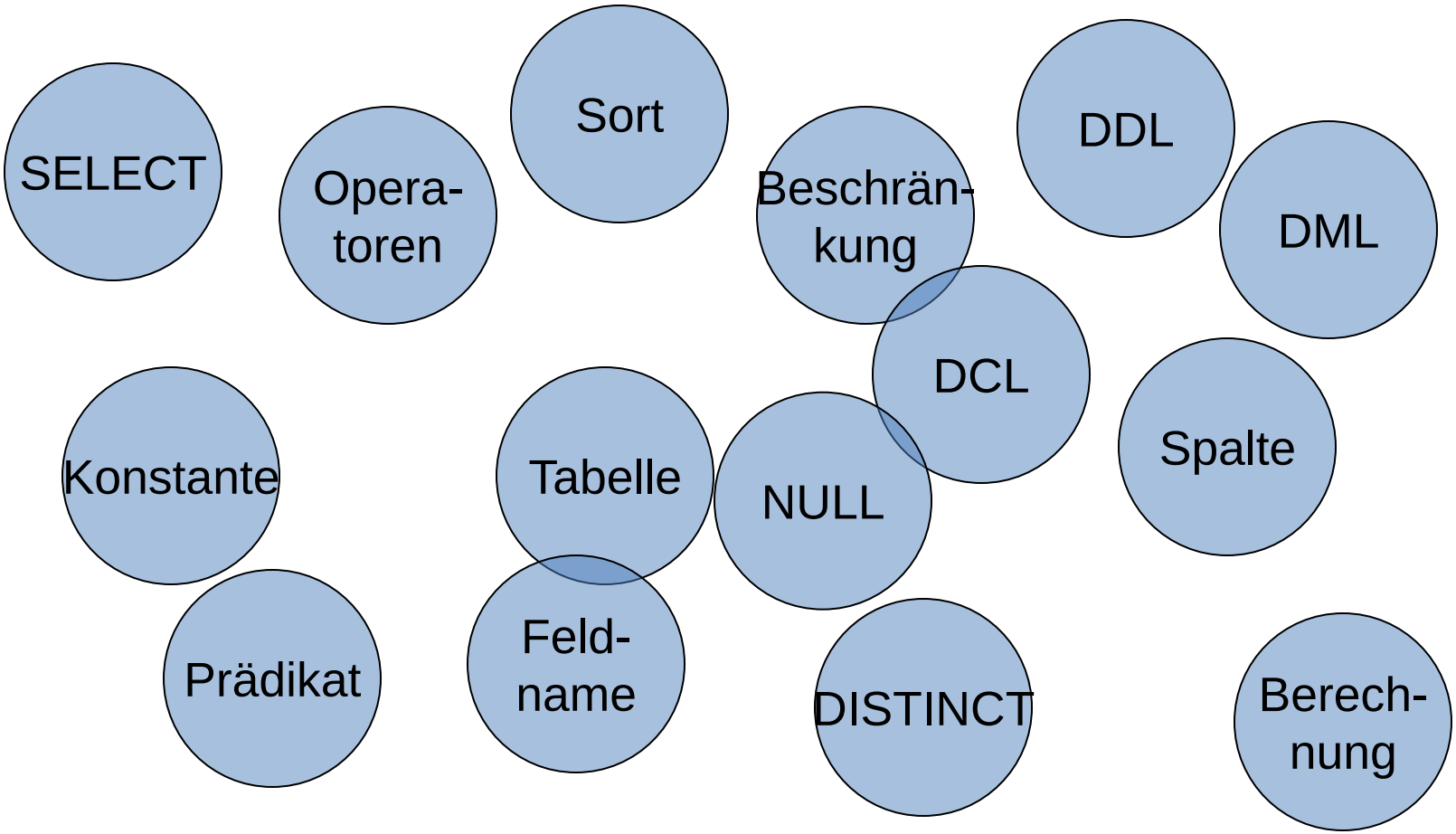
- Kommunikation mit DB2
- DB2I – Überblick
- DB2I – das Hauptmenü
- SPUFI und seine Möglickeiten

Inhalt

- Überblick zum Teil 1
- Abfragen auf 1 Tabelle
- Verknüpfung von Abfragen
- Verschachtelung und Funktionen
- Ändern von Tabellen
- Benutzersicht – View
- Abfrageparameter in Auswahl

Abfragen auf 1 Tabelle

Begriffe



Abfragen auf 1 Tabelle

Datenmanipulation

- Teile der SQL-Sprache
  - DDL
    Data Definition Language
    Datendefinition
  - DML
    Data Modifikation Language
    Datenmanipulation
  - DCL
    Data Control Language
    Datenkontrolle

Abfragen auf 1 Tabelle

die ersten SQL-Befehle

- SELECT
  UPDATE
  DELETE
  INSERT
  - Beispiele basieren auf Materialbeschaffungs-DB
  - Beispiele werden “interaktiv” bearbeitet
  - SQL-Befehle im Programm: später
  - SQLs sind teilweise komplex!

Abfragen auf 1 Tabelle

Beispiel-Tabellen

Lieferant (L)	LNR	LNAME	STATUS	ORT

TEILE (T)	TNR	TNAME	FARBE	GEWICHT	ORT

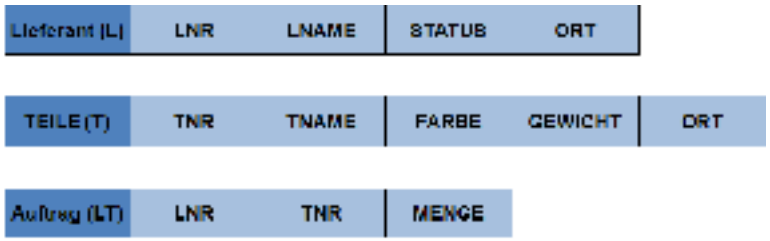
Auftrag (LT)	LNR	TNR	MENGE

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 1

- Aufgabe
  - auswählen von Lieferanten-Nummer und Status der Lieferanten aus Berlin
- Befehl

```
SELECT LNR, LSTATUS
  FROM L
 WHERE ORT = 'BERLIN'
```



Abfragen auf 1 Tabelle

einfacher SELECT – Ergebnis

- Ergebnis

LNR	LSTATUS
L1	30
L4	10

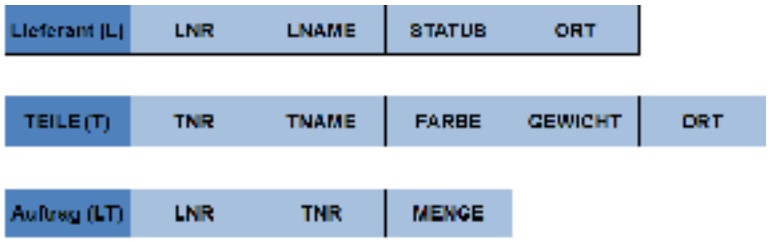
- Das Ergebnis der Abfrage ist wieder eine Relation, d.h. eine Tabelle. Wird das Ergebnis gespeichert, kann dieses Ergebnis mit einem weiteren SELECT abgefragt werden.

Abfragen auf 1 Tabelle

einfacher SELECT – qualifizieren

- Die Abfrage kann (manchmal muss) qualifiziert werden. Das Ergebnis ändert sich in unserem Fall nicht.
  Der SQL sieht dann wie folgt aus:

```
SELECT L.LNR, L.LSTATUS
  FROM L
 WHERE L.ORT = 'BERLIN'
```



Abfragen auf 1 Tabelle

Übung(en)

- Kapitel 1.4.5.1 Beispiel 1

Abfragen auf 1 Tabelle

SELECT – allgemeine Syntax, einfache Form

```
SELECT [DISTINCT] select-liste
  FROM tabelle(n)
[WHERE auswahl-bedingung]
[GROUP BY spaltenname(n)
       [HAVING auswahl-bedingung]]
[ORDER BY spaltenname(n)]
```

Abfragen auf 1 Tabelle

SELECT – allgemeine Syntax, einfache Form – Beschreibung 1

```
SELECT [DISTINCT] select-liste
  FROM tabelle(n)
[WHERE auswahl-bedingung]
[GROUP BY spaltenname(n)
       [HAVING auswahl-bedingung]]
[ORDER BY spaltenname(n)]
```

- select-liste
  - Spaltenname(n) oder *
  - Konstante
  - Kombination aus Spaltenname(n) und Konstante(n) (Ausdruck)
  - Funktion (built-in-function)
- auswahlbedingung (Prädikat)
  - eine oder mehrere Bedingungen
  - Bedingung ist ein Vergleich zwischen 2 Angaben
  - eine Angabe kann einen Spaltennamen, eine Konstante oder einen Ausdruck darstellen

Abfragen auf 1 Tabelle

SELECT – allgemeine Syntax, einfache Form – Beschreibung 2

- GROUP BY
  - erzeugen eines Gruppenwechsels bei Änderung der angegebenen Spalten
  - kann bei komplexen SELECTs weiter verwendet werden wie ORDER BY oder SUBSELECT
- HAVING (Unterparameter zu GROUP BY)
  - Bedingung für Gruppenwechsel
- ORDER BY
  - sortieren der Ergebnismenge
- DISTINCT
  - Duplikate eliminieren

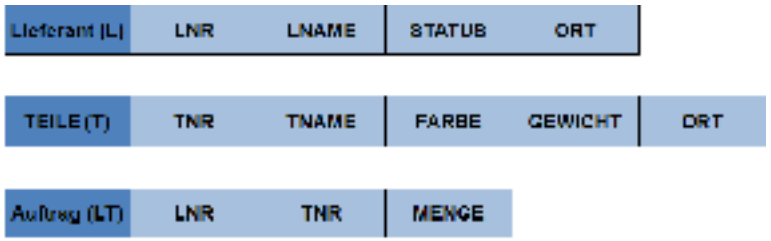
```
SELECT [DISTINCT] select-liste
  FROM tabelle(n)
[WHERE auswahl-bedingung]
[GROUP BY spaltenname(n)
       [HAVING auswahl-bedingung]]
[ORDER BY spaltenname(n)]
```

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 2 – ein Feld auswählen

- Aufgabe
  - auswählen aller Teilenummern der bestellten Materialien
- Befehl

```
SELECT TNR
   FROM LT
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 2 – Ergebnis

- Ergebnis

```
TNR
---
T1
T2
T3
T4
T5
...
```

```
...
T6
T1
T2
T2
T2
...
```

```
...
T4
T5
```

Abfragen auf 1 Tabelle

Übung(en)

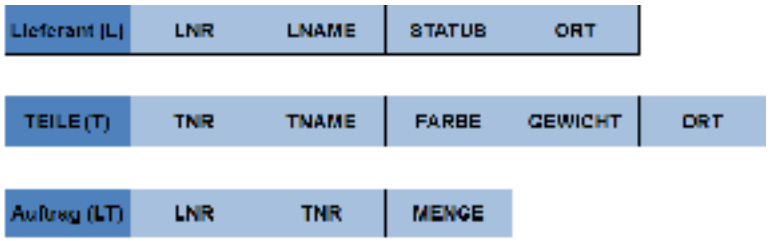
- Kapitel 1.4.5.2 Beispiel 2

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 3 – ohne Duplikate

- Aufgabe
  - auswählen aller Teilenummern der bestellten Materialien ohne Duplikate
- Befehl

```
SELECT DISTINCT TNR
  FROM LT
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 3 – Ergebnis

- Ergebnis

```
TNR
---
T1
T2
T3
T4
T5
T6
```

Abfragen auf 1 Tabelle

Übung(en)

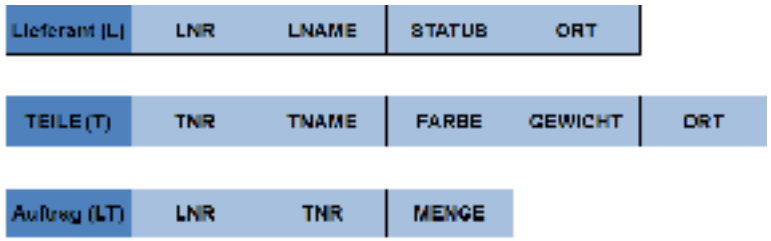
- Kapitel 1.4.5.3 Beispiel 3

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 4 – Berechnung und feste Werte

- Aufgabe
  - Für alle Materialien ist die Teilenummer und das aus dem Nettogewicht und einem konstanten Gewichtsfaktor errechnete Bruttogewicht aufzulisten. Außerdem soll die Formel angezeigt werden.
- Befehl

```
SELECT TNR, 'Nettogewicht x 1,25 =',
            GEWICHT * 1.25
  FROM T
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 4 – Ergebnis

- Ergebnis

```
TNR
--- -------------------- -----
T1  Nettogewicht x 1,25 = 23.75
T2  Nettogewicht x 1,25 = 15.00
T3  Nettogewicht x 1,25 = 17.50
T4  Nettogewicht x 1,25 = 21.25
T5  Nettogewicht x 1,25 = 21.25
T6  Nettogewicht x 1,25 = 15.00
```

Abfragen auf 1 Tabelle

Übung(en)

- Kapitel 1.4.5.4 Beispiel 4

Abfragen auf 1 Tabelle

Arithmetik

- Operatoren
  - addieren +
  - subtrahieren -
  - multiplizieren *
  - dividieren /
- NULL-Werte
  - NULL wird bei Berechnung nicht berücksichtigt
  - ist nur 1 Operand NULL so auch das Ergebnis

Abfragen auf 1 Tabelle

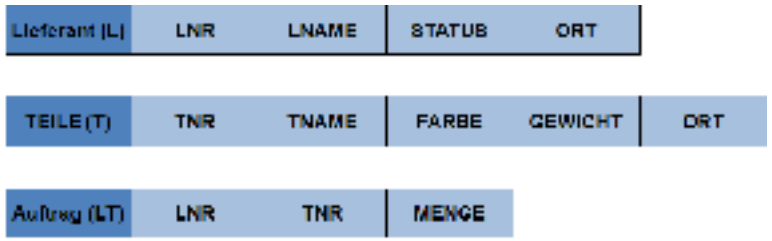
einfacher SELECT – Beispiel 5 – alle Daten

- Aufgabe
  - Auflisten aller Daten der Lieferanten-Tabelle
- Befehl

```
SELECT *
  FROM L
```

- oder

```
SELECT LNR, LNAME, LSTATUS, ORT
  FROM L
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 5 – Ergebnis

- Ergebnis

```
LNR  LNAME     LSTATUS  ORT
---  --------  -------  --------------
L1   NEUMANN         30  BERLIN
L2   SCHMIDT         20  HAMBURG
L3   KRAUSE          30  HAMBURG
L4   MEIER           10  BERLIN
L5   SCHULZ          20  FRANKFURT
```

Abfragen auf 1 Tabelle

Übung(en)

- Kapitel 1.4.5.5 Beispiel 5

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 5 – Bewertung

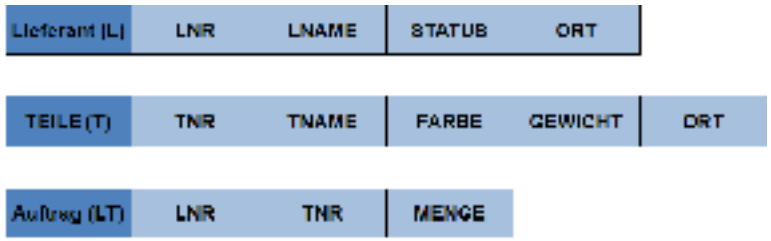
- beide SQLs bringen das gleiche Ergebnis
- * spart Schreibarbeit
- Aber was passiert, wenn diese Abfrage in einem Programm kodiert ist und die Tabelle erweitert wird?
- Also:
  - * gut bei Tests / interaktivem Arbeiten
  - * (fast) niemals im Programm kodieren

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 6 – Bedingung

- Aufgabe
  - Auflisten der LNR der Lieferanten in Hamburg mit einem Status größer als 20.
- Befehl

```
SELECT LNR
  FROM L
 WHERE ORT     =  'Hamburg'
   AND LSTATUS >  20
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 6 – Ergebnis

- Ergebnis

```
LNR
---
L3
```

Abfragen auf 1 Tabelle

Übung(en)

- Kapitel 1.4.5.6 Beispiel 6

Abfragen auf 1 Tabelle

Bedingungen (Prädikate) – Operatoren / Wertigkeit

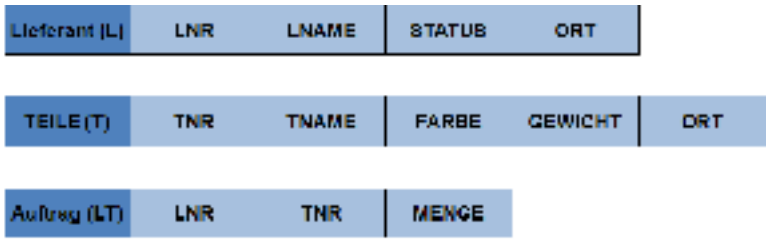
- Vergleichsoperatoren
  = ^= <> > >= ^> < <= ^<
- bool'sche Operatoren
  NOT AND OR
- Klammern
  ( )
- Reihenfolge
  - arithmetische Ausdrücke … Klammern … Vergleichs-operatoren … NOT … AND … OR

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 7 – Sortierung

- Aufgabe
  - Aufsuchen der Lieferantennummer der Lieferanten in Hamburg und Anzeige absteigend sortiert nach dem Lieferantenstatus.
- Befehl

```
SELECT     LNR, LSTATUS
  FROM     L
 WHERE     ORT      =   'Hamburg'
 ORDER BY LSTATUS DESC
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 7 – Ergebnis

- Ergebnis

```
LNR LSTATUS
--- -------
L3       30
L2       20
```

Abfragen auf 1 Tabelle

Übung(en)

- Kapitel 1.4.5.7 Beispiel 7

Abfragen auf 1 Tabelle

Beschreibung ORDER BY

- Syntax:

```
ORDER BY spaltenname [ASC|DESC]
        [,spaltenname [ASC|DESC] …]
```

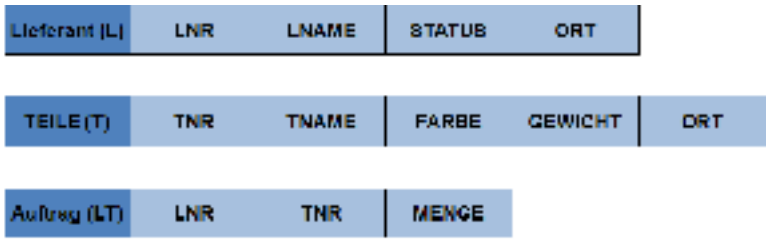
- ohne ORDER BY ist die Ergebnistabelle ohne bestimmte Reihenfolge
- Reihenfolge der Spalten im SELECT
- Reihenfolge der Sortierung im ORDER BY
- ~~Jeder Spaltenname im ORDER BY muss im SELECT spezifiziert werden.~~
- Angabe Spaltennummer erlaubt (Reihenfolge!)

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 8 – Begrenzung – 1

- Aufgabe
  - Liste die Teile auf, deren Gewicht zwischen 16 und 19 kg liegt.
- Befehl

```
SELECT     *
  FROM     T
 WHERE     GEWICHT BETWEEN 16 AND 19
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 8 – Ergebnis

- Ergebnis

```
TNR  TNAME      FARBE  GEWICHT  ORT
---  ---------  -----  -------  ----------
T1   C          BLAU        19  BERLIN
T4   S          BLAU        17  BERLIN
T5   B          ROT         17  HAMBURG
```

Abfragen auf 1 Tabelle

Übung(en)

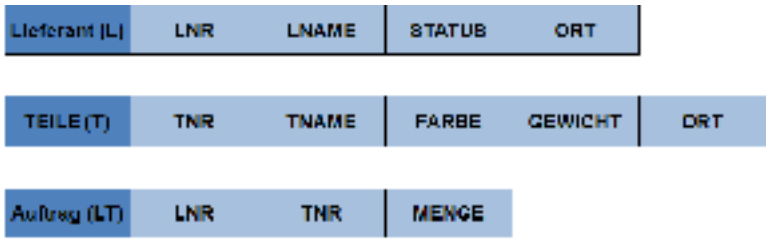
- Kapitel 1.4.5.8 Beispiel 8

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 9 – Begrenzung – 2

- Aufgabe
  - Liste die Teile auf, deren Gewicht * nicht* zwischen 16 und 19 kg liegt.
- Befehl

```
SELECT    *
  FROM    T
 WHERE    GEWICHT NOT BETWEEN 16 AND 19
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 9 – Ergebnis

- Ergebnis

```
TNR TNAME      FARBE GEWICHT ORT
--- ---------- ----- ------- ----------
T2  D          GELB       12 HAMBURG
T3  S          ROT        14 STUTTGART
T6  N          BLAU       12 BERLIN
```

Abfragen auf 1 Tabelle

Übung(en)

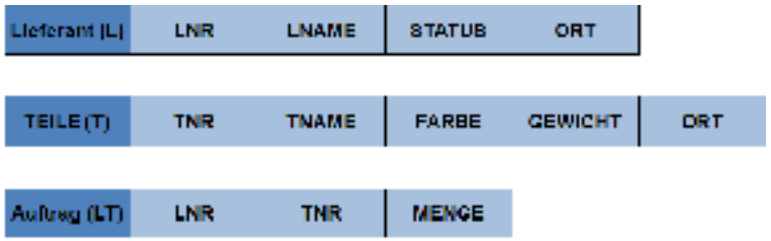
- Kapitel 1.4.5.9 Beispiel 9

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 10 – Wertetabelle – 1

- Aufgabe
  - Liste der Teile mit dem Gewicht 12, 13 oder 17 kg.
- Befehl

```
SELECT     *
  FROM     T
 WHERE     GEWICHT IN (12, 13, 17)
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 10 – Ergebnis

- Ergebnis

```
TNR TNAME      FARBE GEWICHT ORT
--- ---------- ----- ------- ----------
T2  D          GELB       12 HAMBURG
T4  S          BLAU       17 BERLIN
T5  B          ROT        17 HAMBURG
T6  N          BLAU       12 BERLIN
```

Abfragen auf 1 Tabelle

Übung(en)

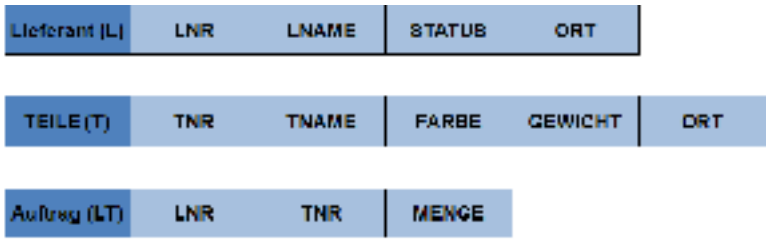
- Kapitel 1.4.5.10 Beispiel 10

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 11 – Wertetabelle – 2

- Aufgabe
  - Liste der Teile mit dem Gewicht 12, 13 oder 17 kg.
- Befehl

```
SELECT    *
  FROM    T
 WHERE    GEWICHT = 12
    OR    GEWICHT = 13
    OR    GEWICHT = 17
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 11 – Ergebnis

- Ergebnis

```
TNR TNAME      FARBE GEWICHT ORT
--- ---------- ----- ------- ----------
T2  D          GELB       12 HAMBURG
T4  S          BLAU       17 BERLIN
T5  B          ROT        17 HAMBURG
T6  N          BLAU       12 BERLIN
```

Abfragen auf 1 Tabelle

Übung(en)

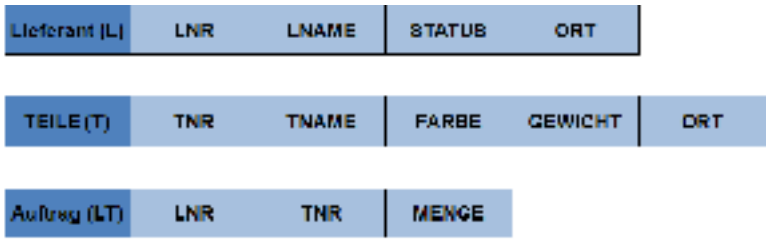
- Kapitel 1.4.5.11 Beispiel 11

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 12 – Wertetabelle – 3

- Aufgabe
  - Liste der Teile, die nicht das Gewicht 12, 13 oder 17 kg haben.
- Befehl

```
SELECT    *
  FROM    T
 WHERE    GEWICHT NOT IN (12, 13, 17)
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 12 – Ergebnis

- Ergebnis

```
TNR  TNAME      FARBE  GEWICHT  ORT
---  ---------  -----  -------  ----------
T1   C          BLAU        19  BERLIN
T3   S          ROT         14  STUTTGART
```

Abfragen auf 1 Tabelle

Übung(en)

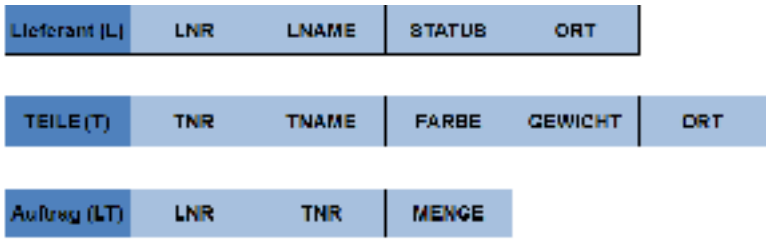
- Kapitel 1.4.5.12 Beispiel 12

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 13 – Wertetabelle – 4

- Aufgabe
  - Liste der Teile, die nicht das Gewicht 12, 13 oder 17 kg haben.
- Befehl

```
SELECT    *
  FROM    T
 WHERE    GEWICHT ^= 12
   AND    GEWICHT ^= 13
   AND    GEWICHT ^= 17
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 13 – Ergebnis

- Ergebnis

```
TNR  TNAME      FARBE  GEWICHT  ORT
---  ---------  -----  -------  ----------
T1   C          BLAU        19  BERLIN
T3   S          ROT         14  STUTTGART
```

Abfragen auf 1 Tabelle

Übung(en)

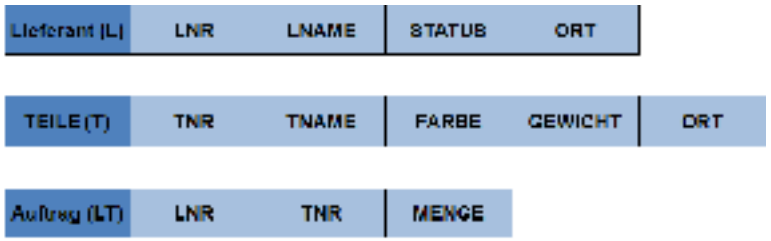
- Kapitel 1.4.5.13 Beispiel 13

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 14 – NULL – 1

- Annahme: Lieferant L3 hat NULL statt ‘30’
- Aufgabe
  - Liste der Lieferantennummern der Lieferanten mit einem Status > ’25’
- Befehl

```
SELECT    LNR
  FROM    L
 WHERE    LSTATUS > 25
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 14 – Ergebnis

- Ergebnis

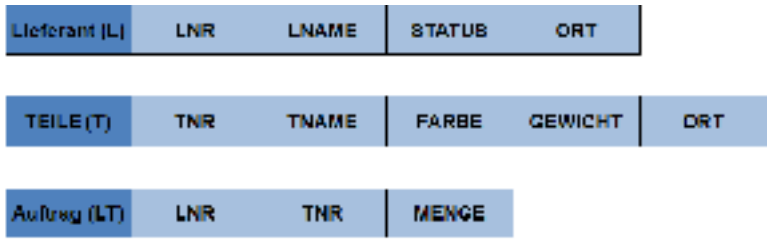
```
LNR
---
L1
```

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 14 – NULL – 2

- Annahme: Lieferant L3 hat NULL statt ‘30’
- Aufgabe
  - Liste die Lieferantennummer der Lieferanten mit einem Status <= ’25’
- Befehl

```
SELECT     LNR
  FROM     L
 WHERE     LSTATUS <= 25
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 14 – Ergebnis

- Ergebnis

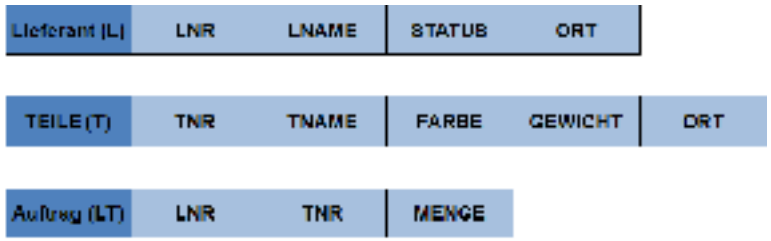
```
LNR
---
L2
L4
L5
```

Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 14 – NULL – 3

- Annahme: Lieferant L3 hat NULL statt ‘30’
- Aufgabe
  - Liste die Lieferantennummer der Lieferanten mit einem Status NULL
- Befehl

```
SELECT     LNR
  FROM     L
 WHERE     LSTATUS IS NULL
```



Abfragen auf 1 Tabelle

einfacher SELECT – Beispiel 14 – Ergebnis

- Ergebnis

```
LNR
---
L3
```

Abfragen auf 1 Tabelle

Übung(en)

- Kapitel 1.4.5.14 Beispiel 14

Abfragen auf 1 Tabelle

NULL – Bewertung

- NULL ist weder größer noch kleiner, er ist nicht ungleich gegenüber einem anderen Wert, auch nicht gegenüber einem NULL-Wert.
- SYNTAX: spaltenname IS [NOT] NULL
- Achtung:
  - DISTINCT: Duplikate werden erkannt
  - UNIQUE INDEX: lässt nur 1 NULL-Wert zu
  - ORDER BY: NULL > alle Nicht-NULL-Werte

Abfragen auf 1 Tabelle

Übung(en)

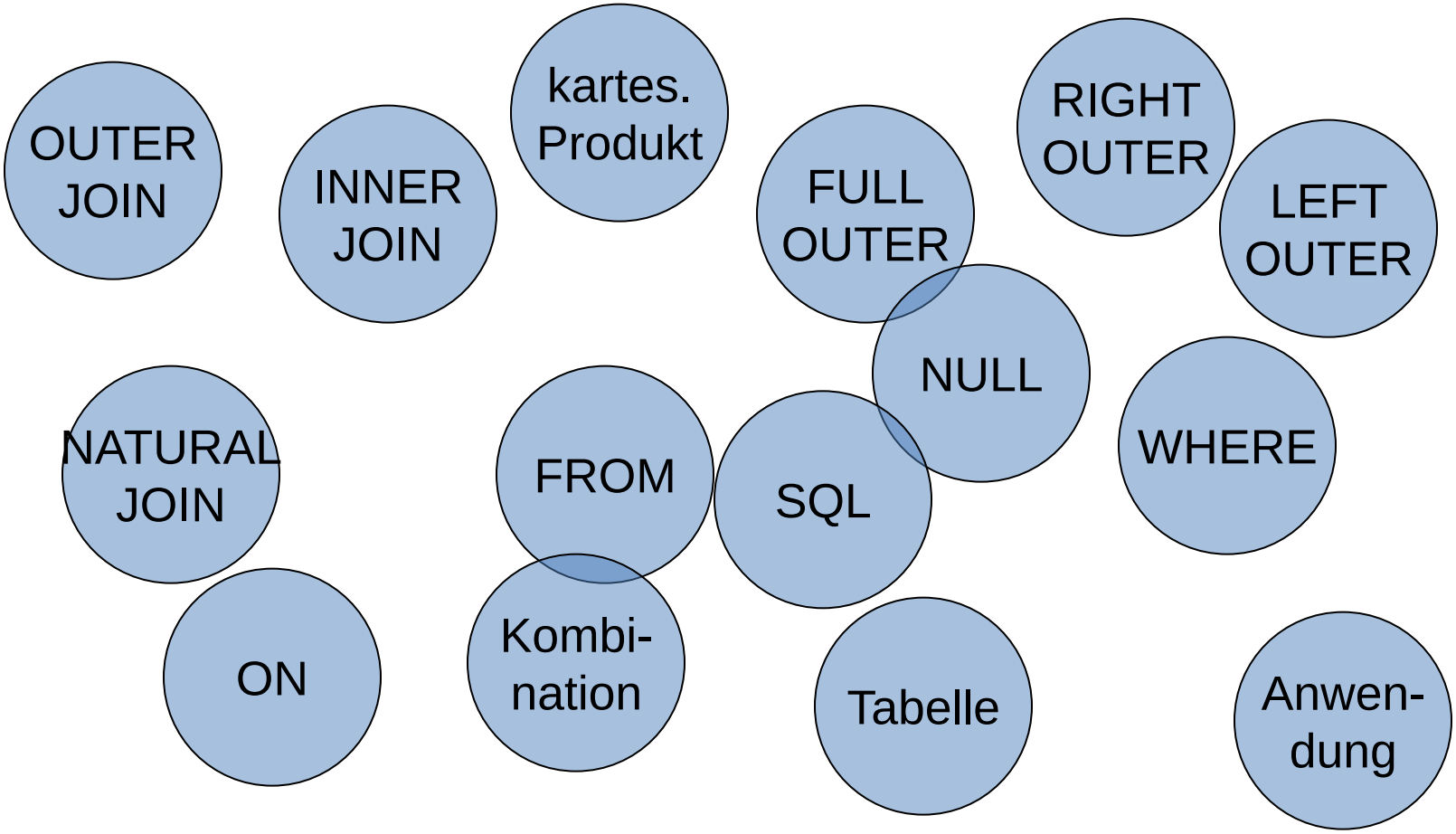
- Kapitel 3.1 Projektdaten aller Projekte
- Kapitel 3.2 Projektdaten einer Lokation
- Kapitel 3.3 Sortieren
- Kapitel 3.4 Auswahl von Daten

Inhalt

- Überblick zum Teil 1
- Abfragen auf 1 Tabelle
- Verknüpfung von Abfragen
- Verschachtelung und Funktionen
- Ändern von Tabellen
- Benutzersicht – View
- Abfrageparameter in Auswahl

Verknüpfung von Abfragen

Begriffe

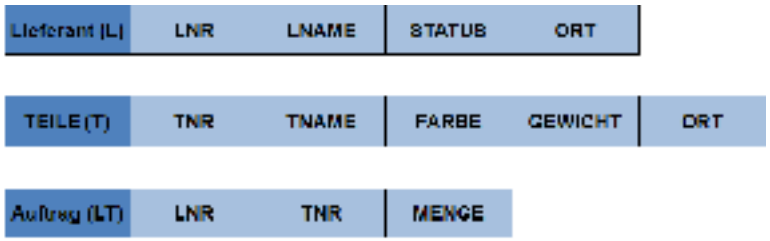


Verknüpfung von Abfragen

Join

- Aufgabe
  - Kombiniere die Daten aus Tabellen “L” und “T”, bei denen der Ort des Lieferanten gleich dem Ort des Teilelagers ist.
- Befehl

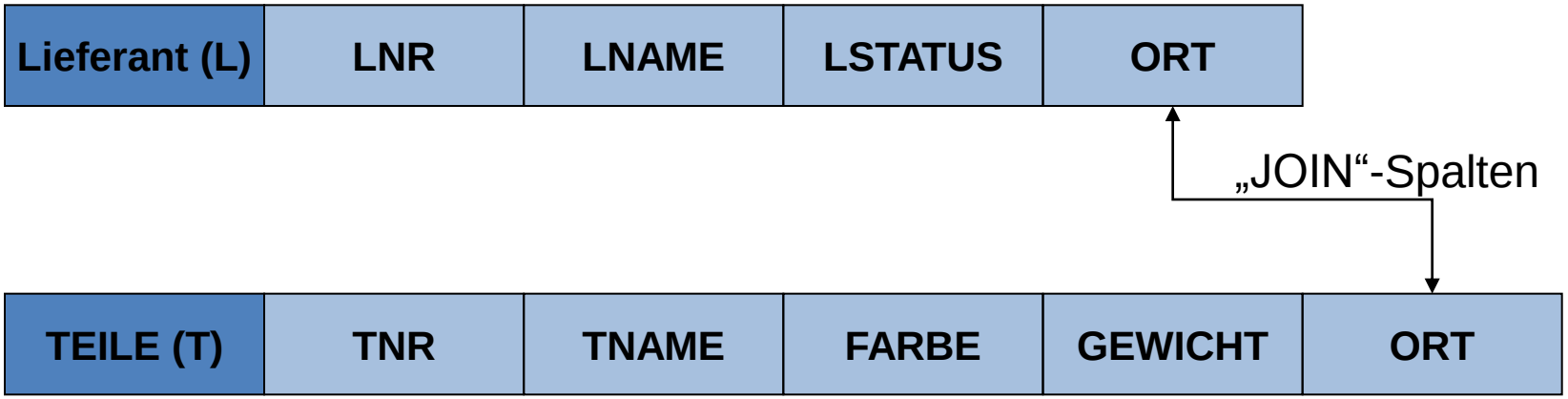
```
SELECT    *
  FROM    L, T
 WHERE    L.ORT = T.ORT
```



Verknüpfung von Abfragen

Join – Beschreibung

- zu beachten:
  - Die Spaltennamen in der Bedingung werden durch die Tabellennamen qualifiziert.



Verknüpfung von Abfragen

Join – Anweisung

- Über Feldinhalte werden Beziehungen zwischen 2 oder mehr Tabellen hergestellt.
- Im FROM-Teil werden 2 oder mehr Tabellen genannt.
- Der WHERE-Teil enthält die “JOIN-Bedingung”. Eine Angabe der Bedingung bezieht sich auf eine Spalte einer der zu “joinenden” Tabelle, die andere auf eine Spalte der anderen Tabelle.
- Der WHERE-Teil kann erweitert werden.

Verknüpfung von Abfragen

INNER JOIN – Ergebnis – Beschreibung

- Für alle laut JOIN-Bedingung möglichen Kombinationen werden Ergebniszeilen gebildet.
- Nur Zeilen mit übereinstimmenden Daten der JOIN-Bedingung werden verbunden.
- Achtung: Gibt es irgendwo Daten mit Frankfurt oder Stuttgart? Warum nicht?
- Der beschriebene JOIN wird INNER-JOIN oder NATURAL-JOIN genannt.

Verknüpfung von Abfragen

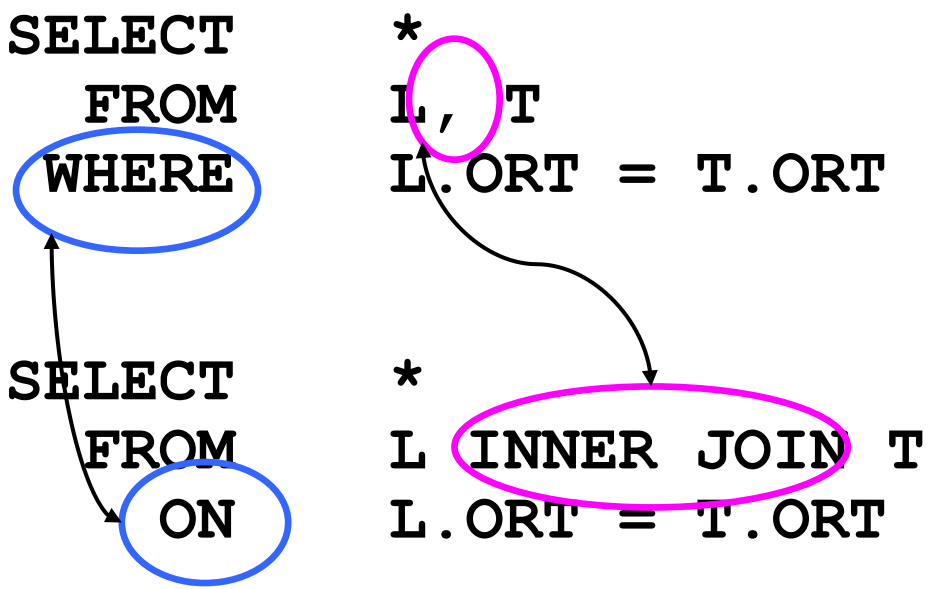
INNER JOIN – Ergebnis

LNR	LNAME	LSTATUS	ORT	TNR	TNAME	FARBE	GEWICHT	ORT
L1	NEUMANN	30	BERLIN	T1	C	BLAU	19	BERLIN
L1	NEUMANN	30	BERLIN	T4	S	BLAU	17	BERLIN
L1	NEUMANN	30	BERLIN	T6	N	BLAU	12	BERLIN
L2	SCHMIDT	20	HAMBURG	T2	D	GELB	12	HAMBURG
L2	SCHMIDT	20	HAMBURG	T5	B	ROT	17	HAMBURG
L3	KRAUSE	30	HAMBURG	T2	D	GELB	12	HAMBURG
L3	KRAUSE	30	HAMBURG	T5	B	ROT	17	HAMBURG
L4	MEIER	10	BERLIN	T1	C	BLAU	19	BERLIN
L4	MEIER	10	BERLIN	T4	S	BLAU	17	BERLIN
L4	MEIER	10	BERLIN	T6	N	BLAU	12	BERLIN

Verknüpfung von Abfragen

JOIN – alternative Schreibweisen

- Befehl:



Verknüpfung von Abfragen

Übung(en)

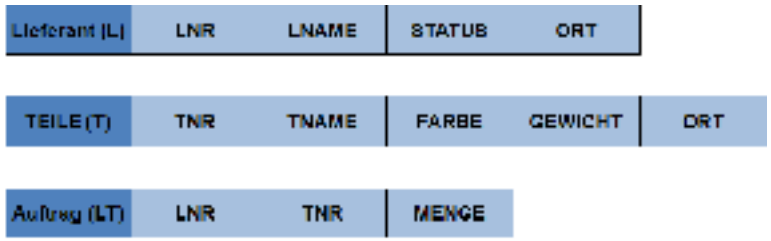
- Kapitel 1.4.5.15 Beispiel 15

Verknüpfung von Abfragen

Kartesisches Produkt

- Aufgabe
  - Anzeige aller Daten aus den Tabellen Lieferant und Teil. Was ist wirklich gewollt?
- Befehl

```
SELECT    *
  FROM    L, T
```



Verknüpfung von Abfragen

Kartesisches Produkt – Ergebnis

- Ergebnis

LNR	LNAME	LSTATUS	ORT	TNR	TNAME	FARBE	GEWICHT	ORT
L1	NEUMANN	30	BERLIN	T1	C	BLAU	19	BERLIN
L1	NEUMANN	30	BERLIN	T2	D	GELB	12	HAMBURG
L1	NEUMANN	30	BERLIN	T3	S	ROT	14	STUTTGART
L1	NEUMANN	30	BERLIN	T4	S	BLAU	17	BERLIN
L1	NEUMANN	30	BERLIN	T5	B	ROT	17	HAMBURG
L1	NEUMANN	30	BERLIN	T6	N	BLAU	12	BERLIN
L2	SCHMIDT	20	HAMBURG	T1	C	BLAU	19	BERLIN
L2	SCHMIDT	20	HAMBURG	T2	D	GELB	12	HAMBURG
L2	SCHMIDT	20	HAMBURG	T3	S	ROT	14	STUTTGART
L2	SCHMIDT	20	HAMBURG	T4	S	BLAU	17	BERLIN
L2	SCHMIDT	20	HAMBURG	T5	B	ROT	17	HAMBURG
L2	SCHMIDT	20	HAMBURG	T6	N	BLAU	12	BERLIN
L3	KRAUSE	30	HAMBURG	T1	C	BLAU	19	BERLIN
L3	KRAUSE	30	HAMBURG	T2	D	GELB	12	HAMBURG
L3	KRAUSE	30	HAMBURG	T3	S	ROT	14	STUTTGART
L3	KRAUSE	30	HAMBURG	T4	S	BLAU	17	BERLIN
L3	KRAUSE	30	HAMBURG	T5	B	ROT	17	HAMBURG
L3	KRAUSE	30	HAMBURG	T6	N	BLAU	12	BERLIN
L4	MEIER	10	BERLIN	T1	C	BLAU	19	BERLIN
L4	MEIER	10	BERLIN	T2	D	GELB	12	HAMBURG
L4	MEIER	10	BERLIN	T3	S	ROT	14	STUTTGART
L4	MEIER	10	BERLIN	T4	S	BLAU	17	BERLIN
L4	MEIER	10	BERLIN	T5	B	ROT	17	HAMBURG
L4	MEIER	10	BERLIN	T6	N	BLAU	12	BERLIN
L5	SCHULZ	20	FRANKFURT	T1	C	BLAU	19	BERLIN
L5	SCHULZ	20	FRANKFURT	T2	D	GELB	12	HAMBURG
L5	SCHULZ	20	FRANKFURT	T3	S	ROT	14	STUTTGART
L5	SCHULZ	20	FRANKFURT	T4	S	BLAU	17	BERLIN
L5	SCHULZ	20	FRANKFURT	T5	B	ROT	17	HAMBURG
L5	SCHULZ	20	FRANKFURT	T6	N	BLAU	12	BERLIN

Verknüpfung von Abfragen

Übung(en)

- Kapitel 1.4.5.16 Beispiel 16

Verknüpfung von Abfragen

Kartesisches Produkt – Erklärung

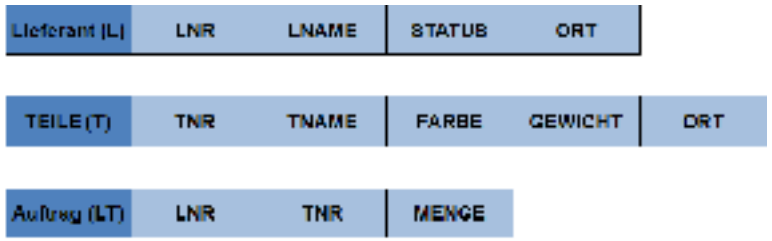
- Das Weglassen der JOIN-Bedingung ergibt das kartesische Produkt.
  - Herkunft: Decartes, Vektor-Produkt
- Ergebnistabelle enthält alle möglichen Kombinationen der Zeilen der Tabellen.
- Anzahl Zeilen = Anz-Tab-1 * Anz-Tab-2
- Mit JOIN-Bedingung fallen alle Zeilen weg, die der Bedingung nicht genügen.

Verknüpfung von Abfragen

FULL OUTER JOIN

- Aufgabe
  - Kombiniere Daten aus den Tabellen L und T, bei denen der Ort des Lieferanten gleich dem Ort des Teilelagers ist. Zusätzlich sollen die Zeilen ausgegeben werden, die keine Übereinstimmungen haben.
- Befehl

```
SELECT    *
  FROM    L FULL OUTER JOIN T
    ON    L.ORT = T.ORT
```



Verknüpfung von Abfragen

FULL OUTER JOIN – Ergebnis

- Ergebnis

LNR	LNAME	LSTATUS	ORT	TNR	TNAME	FARBE	GEWICHT	ORT
L1	NEUMANN	30	BERLIN	T1	C	BLAU	19	BERLIN
L1	NEUMANN	30	BERLIN	T4	S	BLAU	17	BERLIN
L1	NEUMANN	30	BERLIN	T6	N	BLAU	12	BERLIN
L2	SCHMIDT	20	HAMBURG	T2	D	GELB	12	HAMBURG
L2	SCHMIDT	20	HAMBURG	T5	B	ROT	17	HAMBURG
L3	KRAUSE	30	HAMBURG	T2	D	GELB	12	HAMBURG
L3	KRAUSE	30	HAMBURG	T5	B	ROT	17	HAMBURG
L4	MEIER	10	BERLIN	T1	C	BLAU	19	BERLIN
L4	MEIER	10	BERLIN	T4	S	BLAU	17	BERLIN
L4	MEIER	10	BERLIN	T6	N	BLAU	12	BERLIN
L5	SCHULZ	20	FRANKFURT					
				T3	S	ROT	14	STUTTGART

Verknüpfung von Abfragen

Übung(en)

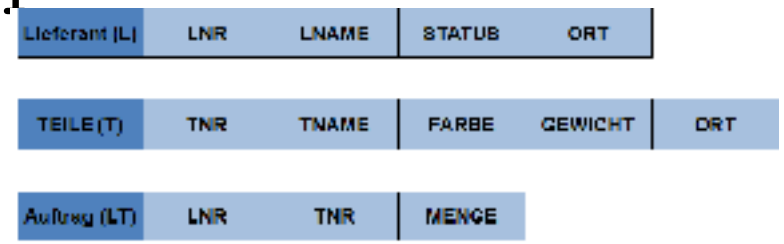
- Kapitel 1.4.5.17 Beispiel 17

Verknüpfung von Abfragen

LEFT OUTER JOIN

- Aufgabe
  - Kombiniere Daten aus den Tabellen L und T, bei denen der Ort des Lieferanten gleich dem Ort des Teilelagers ist. Zusätzlich sollen die Zeilen aus L ausgegeben werden, die keine Übereinstimmung in T haben.
- Befehl

```
SELECT    *
  FROM    L LEFT OUTER JOIN T
    ON    L.ORT = T.ORT
```



Verknüpfung von Abfragen

LEFT OUTER JOIN – Ergebnis

- Ergebnis

LNR	LNAME	LSTATUS	ORT	TNR	TNAME	FARBE	GEWICHT	ORT
L1	NEUMANN	30	BERLIN	T1	C	BLAU	19	BERLIN
L1	NEUMANN	30	BERLIN	T4	S	BLAU	17	BERLIN
L1	NEUMANN	30	BERLIN	T6	N	BLAU	12	BERLIN
L2	SCHMIDT	20	HAMBURG	T2	D	GELB	12	HAMBURG
L2	SCHMIDT	20	HAMBURG	T5	B	ROT	17	HAMBURG
L3	KRAUSE	30	HAMBURG	T2	D	GELB	12	HAMBURG
L3	KRAUSE	30	HAMBURG	T5	B	ROT	17	HAMBURG
L4	MEIER	10	BERLIN	T1	C	BLAU	19	BERLIN
L4	MEIER	10	BERLIN	T4	S	BLAU	17	BERLIN
L4	MEIER	10	BERLIN	T6	N	BLAU	12	BERLIN
L5	SCHULZ	20	FRANKFURT	-	-	-	-	-

Verknüpfung von Abfragen

Übung(en)

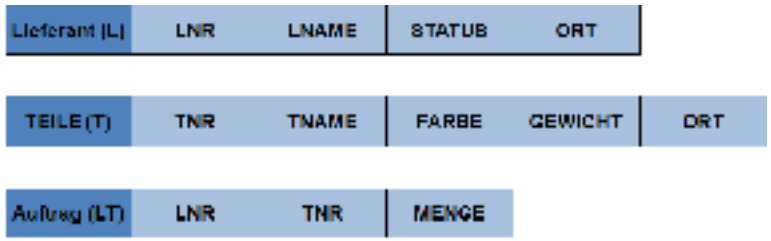
- Kapitel 1.4.5.18 Beispiel 18

Verknüpfung von Abfragen

RIGHT OUTER JOIN

- Aufgabe
  - Kombiniere Daten aus den Tabellen L und T, bei denen der Ort des Lieferanten gleich dem Ort des Teilelagers ist. Zusätzlich sollen die Zeilen aus T ausgegeben werden, die keine Übereinstimmung in L haben.
- Befehl

```
SELECT    *
  FROM    L RIGHT OUTER JOIN T
    ON    L.ORT = T.ORT
```



Verknüpfung von Abfragen

RIGHT OUTER JOIN – Ergebnis

- Ergebnis

LNR	LNAME	LSTATUS	ORT	TNR	TNAME	FARBE	GEWICHT	ORT
L1	NEUMANN	30	BERLIN	T1	C	BLAU	19	BERLIN
L1	NEUMANN	30	BERLIN	T4	S	BLAU	17	BERLIN
L1	NEUMANN	30	BERLIN	T6	N	BLAU	12	BERLIN
L2	SCHMIDT	20	HAMBURG	T2	D	GELB	12	HAMBURG
L2	SCHMIDT	20	HAMBURG	T5	B	ROT	17	HAMBURG
L3	KRAUSE	30	HAMBURG	T2	D	GELB	12	HAMBURG
L3	KRAUSE	30	HAMBURG	T5	B	ROT	17	HAMBURG
L4	MEIER	10	BERLIN	T1	C	BLAU	19	BERLIN
L4	MEIER	10	BERLIN	T4	S	BLAU	17	BERLIN
L4	MEIER	10	BERLIN	T6	N	BLAU	12	BERLIN
				T3	S	ROT	14	STUTTGART

Verknüpfung von Abfragen

Übung(en)

- Kapitel 1.4.5.19 Beispiel 19

Verknüpfung von Abfragen

cps4it

Übung(en)

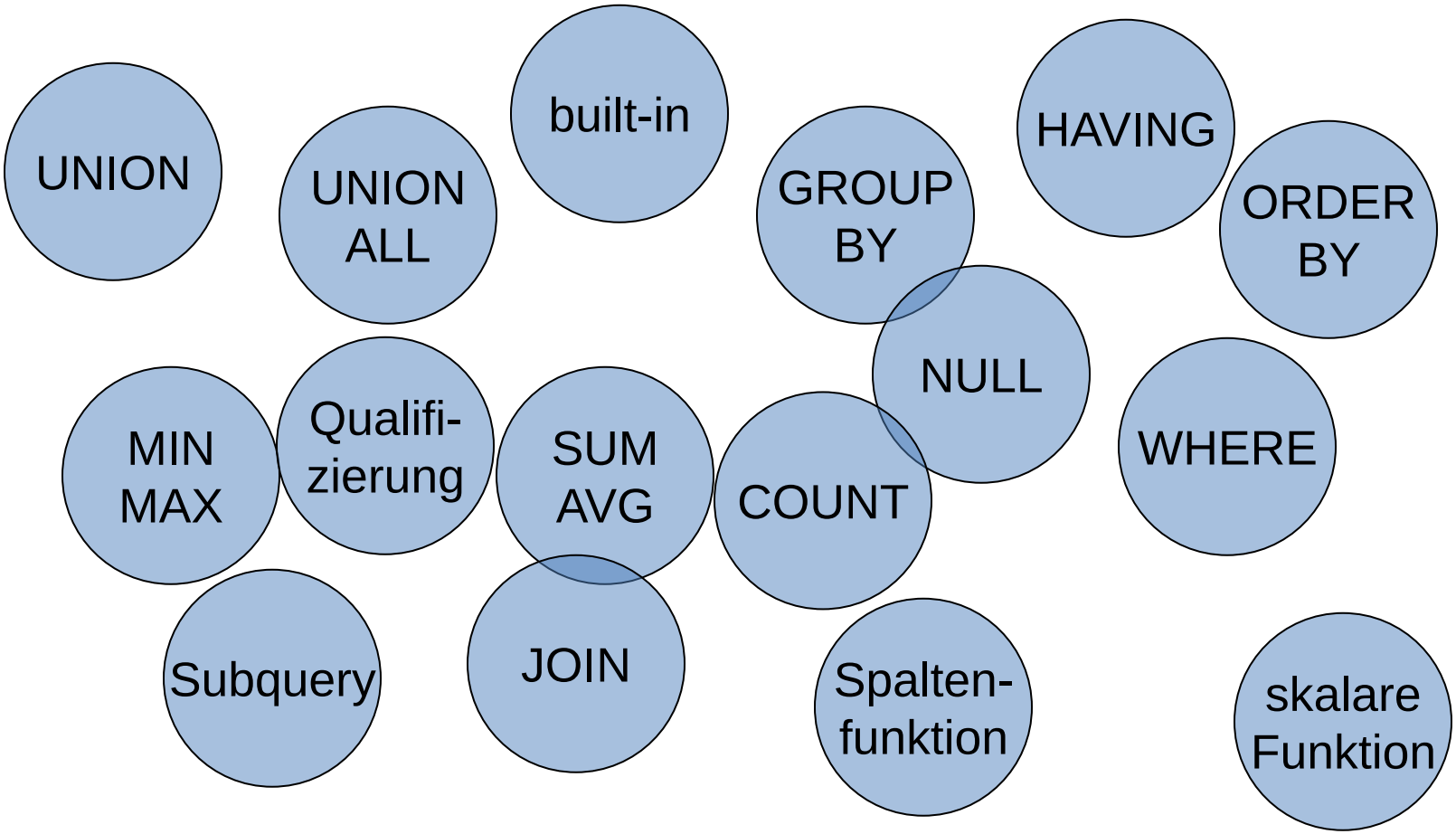
- Kapitel 4.1 Aufträge mit Name Lieferant
- Kapitel 4.2 Teile von bestimmten Lieferanten
- Kapitel 4.3 Aufträge Proj.daten / k. Aufträge

Inhalt

- Überblick zum Teil 1
- Abfragen auf 1 Tabelle
- Verknüpfung von Abfragen
- Verschachtelung und Funktionen
- Ändern von Tabellen
- Benutzersicht – View
- Abfrageparameter in Auswahl

Verschachtelung und Funktionen

Begriffe

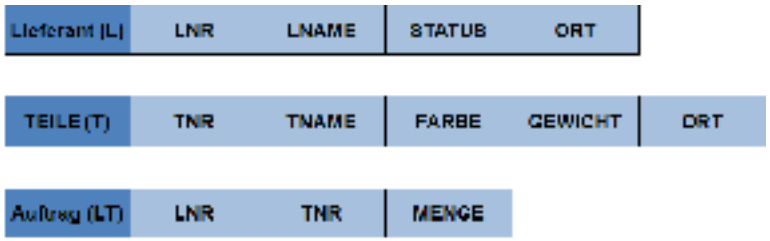


Verschachtelung und Funktionen

Unterabfrage (Subquery)

- Aufgabe
  - Liste der Namen aller Lieferanten, die Teil T2 liefern.
- Befehl

```
SELECT     LNAME
  FROM     L
 WHERE     LNR IN (SELECT LNR
                    FROM LT
                   WHERE TNR = 'T3')
```



Verschachtelung und Funktionen

Unterabfrage (Subquery) – Ergebnis / Teilergebnis

- Ergebnis:

```
LNAME
---------------
SCHMIDT
KRAUSE
SCHULZ
```

- die geschachtelte Unterabfrage

```
SELECT LNR
  FROM LT
 WHERE TNR = 'T3'
```

- liefert:

```
L2, L3, L5
```

Verschachtelung und Funktionen

Unterabfrage (Subquery) – Qualifizierung von Spaltennamen

- Was ist, wenn Felder gleich heißen?
- Wird vom Benutzer keine Qualifizierung vorgenommen, geht DB2 von bestimmten Annahmen aus:
  - Es nimmt den Tabellennamen des FROM-Teils, der unmittelbar Bestandteil der jeweiligen Unter- oder Hauptabfrage ist.
  - Gibt es für den Tabellennamen einen Alias, wird dieser benutzt.

Verschachtelung und Funktionen

Übung(en)

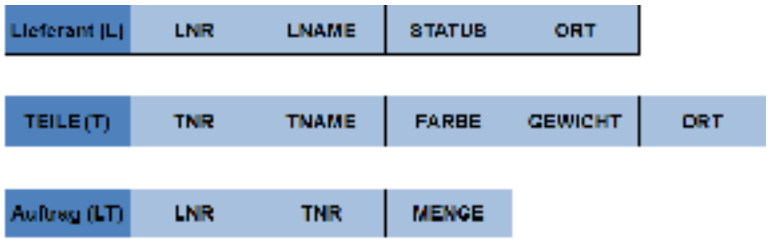
- Kapitel 1.4.5.20 Beispiel 20

Verschachtelung und Funktionen

Unterabfrage (Subquery) – Qualifizierung von Spaltennamen

- Beispiel mit Qualifizierung

```
SELECT    L.LNAME
  FROM    L
 WHERE    L.LNR IN (SELECT LT.LNR
                      FROM LT
                     WHERE LT.TNR = 'T3')
```



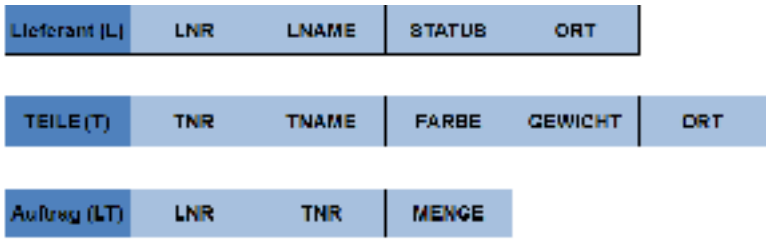
Verschachtelung und Funktionen

Unterabfrage (Subquery) – Subquery oder JOIN

- Bitte JOIN nutzen statt Subquery

```
SELECT     L.LNAME
  FROM     L INNER JOIN LT
           ON L.LNR = LT.LNR
 WHERE     LT.TNR = 'T2'
```

- denn:
  - leichter lesbar
  - (etwas) schneller



Verschachtelung und Funktionen

mehrfache Verschachtelung

- Aufgabe:
  - Benötigt wird eine Liste der Lieferanten, welchen mindestens ein Teil mit der Farbe ROT liefern.
- Befehl

```
SELECT    LNAME
  FROM    L
 WHERE    LNR IN
                ( SELECT LNR
                    FROM LT
                   WHERE TNR IN
                                ( SELECT TNR
                                    FROM T
                                   WHERE FARBE = 'ROT')
                 )
```

Verschachtelung und Funktionen

mehrfache Verschachtelung – Ergebnis

- Ergebnis:

```
LNAME
---------------
NEUMANN
MEIER
```

Verschachtelung und Funktionen

mehrfache Verschachtelung – geht das auch anders?

- Aufgabe:
  - Benötigt wird eine Liste der Lieferanten, welchen mindestens ein Teil mit der Farbe ROT liefern.
- Befehl

```
SELECT ...
JOIN ...
???
```

Verschachtelung und Funktionen

mehrfache Verschachtelung – Hinweise

- … wenn es tatsächlich nicht ohne Subqueries geht …
- Werden einfache Vergleichsoperatoren (=,> etc.) verwendet, muss sichergestellt sein, dass die Unterabfrage nur 1 Wert liefert.
- Es führt zu keinem Fehler, wenn die Unterabfrage keinen Wert liefert. Dieser Fall wird wie das Ergebnis NULL behandelt.
- Die Unterabfrage muss direkt nach dem Vergleichsoperator stehen.

Verschachtelung und Funktionen

Übung(en)

- Kapitel 1.4.5.21 Beispiel 21

Verschachtelung und Funktionen

Übung(en)

- Kapitel 4.4 Join statt Subquery

Verschachtelung und Funktionen

Built-In-Funktionen – Spaltenfunktionen

- aggregate functions
- 1 Ergebnis aus mehreren ausgewählten Zeilen einer Spalte oder Gruppe
- generelle Syntax:
  - funktion(argument)
- Spaltenfunktionen sind bei WHERE nicht erlaubt
  - COUNT, SUM, AVG, MAX, MIN, STDDEV, VARIANCE

Verschachtelung und Funktionen

Built-In-Funktionen – Tabellenfunktionen

- table functions
- nur im FROM-Statement
- im Zusammenhang mit CREATE TABLE
- Funktionen
  - MQREADALL, MQREADALLCLOB, MQREADALLXML, MQRECEIVEALL, MQRECEIVEALLCLOB, MQRECEIVEALLXML

Verschachtelung und Funktionen

Built-In-Funktionen – skalare Funktionen

- scalar functions
- 1-n Werte liefert/n 1 Ergebnis
- keine Gruppe möglich
- Beispiele:
  - Konvertierung
  - Stringmanipulation
- siehe Schulungsunterlagen
- siehe DB2 UDB for z/OS SQL-Reference

Link Built-in-Funktionen

Verschachtelung und Funktionen

Spaltenfunktionen – einfache Beispiele – 1

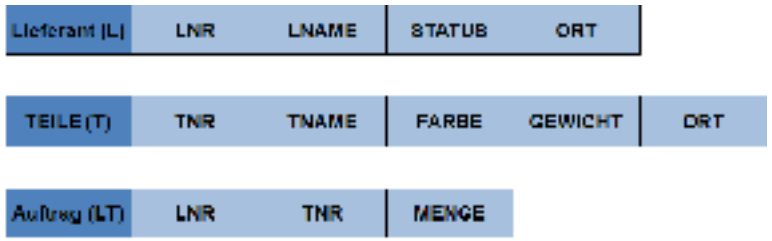
- COUNT (*)
  - gibt eine Zahl mit der Anzahl der Zeilen zurück, die die Suchbedingung erfüllen
- Beispiele:

```
SELECT     COUNT(*)
  FROM     L

-> 5
```

```
SELECT     COUNT(*)
  FROM     LT
 WHERE     TNR = 'T2'

-> 2
```



Verschachtelung und Funktionen

Spaltenfunktionen – einfache Beispiele – 2

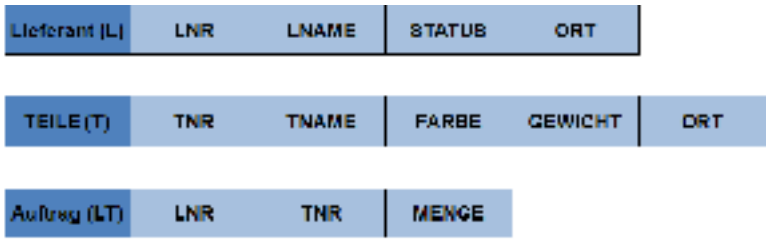
- COUNT (DISTINCT spaltenname)
  - gibt eine Zahl mit der Anzahl der unterschiedlichen Zeilen zurück, die die Suchbedingung erfüllen
- Beispiel:

```
SELECT COUNT
       (DISTINCT LNR)
  FROM LT

-> 5
```

```
SELECT COUNT(LNR)
  FROM LT

-> 24
```



Verschachtelung und Funktionen

Übung(en)

- Kapitel 1.4.5.22 Beispiel 22

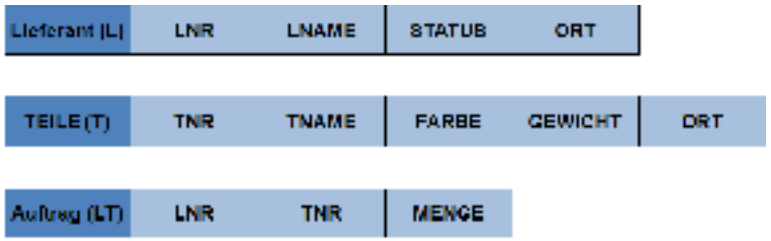
Verschachtelung und Funktionen

Spaltenfunktionen – einfache Beispiele – 3

- SUM
  - errechnet den Gesamtwert der Spalte
  - nur bei nummerischen Daten möglich
  - DISTINCT kann benutzt werden; dadurch werden nur unterschiedliche Werte addiert
  - NULL wird bei Summierung nicht berücksichtigt

```
SELECT     SUM (MENGE)
  FROM     LT
 WHERE     TNR = 'T4'

-> 1400
```



Verschachtelung und Funktionen

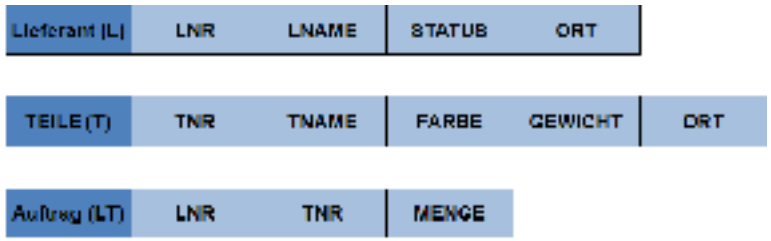
Übung(en)

- Kapitel 1.4.5.23 Beispiel 23

Verschachtelung und Funktionen

Spaltenfunktionen – einfache Beispiele – 4

- AVG
  - errechnet den Durchschnittswert der Spalte
  - sinngemäßige Logik wie SUM
  - NULL wird bei Berechnung nicht berücksichtigt
- MIN / MAX
  - findet den kleinsten bzw. größten Wert der Spalte
  - ist für *alle* Datentypen möglich
  - NULL wird bei Berechnung nicht berücksichtigt

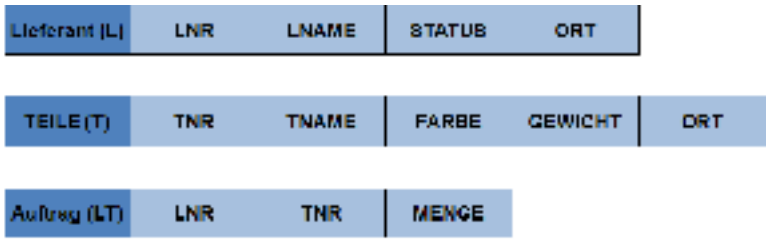


Verschachtelung und Funktionen

Gruppierung – Aufgabe

- Aufgabe
  - Zeige je Teil die Summe der im Auftrag befindlichen Menge an.
- Befehl

```
SELECT     TNR, SUM(MENGE)
  FROM     LT
 GROUP BY TNR
```



Verschachtelung und Funktionen

Gruppierung – Ergebnis

- Ergebnis:

```
TNR
--- -----
T1   1200
T2    400
T3   4300
T4   1400
T5   1200
T6   1500
```

Verschachtelung und Funktionen

Gruppierung – Erläuterung

- Alle Spalten nach dem SELECT (außer den Spalten in GROUP BY) müssen sich auf eine built-in-Funktion beziehen, weil je Gruppe nur 1 Wert ausgewiesen wird.
- Enthalten irgendwelche Zeilen in der GROUP-BY-Spalte NULL-Werte, so wird jede dieser Zeilen als eine Gruppe behandelt.
- GROUP BY hat nichts mit ORDER BY zu tun!

Verschachtelung und Funktionen

Übung(en)

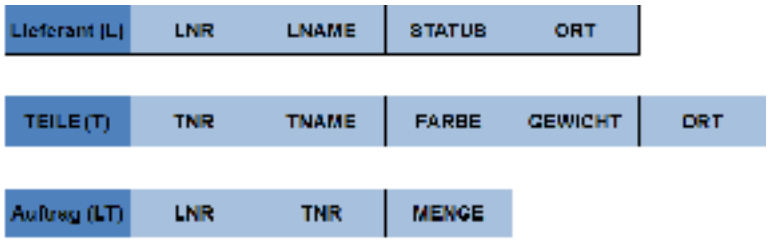
- Kapitel 1.4.5.24 Beispiel 24

Verschachtelung und Funktionen

Gruppen mit Eigenschaften – Aufgabe

- Aufgabe
  - Liste die Teilenummern der Teile auf, die von mehr als 1 Lieferanten geliefert werden.
- Befehl

```
SELECT     TNR
  FROM     LT
GROUP BY   TNR
HAVING     COUNT(*) > 2
```



Verschachtelung und Funktionen

Gruppen mit Eigenschaften – Ergebnis

- Ergebnis:

```
TNR
---
T1
T3
T5
T6
```

Verschachtelung und Funktionen

Übung(en)

- Kapitel 1.4.5.25 Beispiel 25

Verschachtelung und Funktionen

Gruppen mit Eigenschaften – Erläuterung

- WHERE wählt Zeile aus
- HAVING wählt Gruppen aus
- HAVING darf nur Ausdrücke enthalten, die nur einen Wert je Gruppe enthalten.
- Falls kein GROUP BY kodiert worden ist, wird die gesamte Tabelle als Gruppe angesehen

Verschachtelung und Funktionen

Gruppen mit Eigenschaften – Befehlsablauf

1. FROM — Auswahl der Tabelle
2. WHERE — Auswählen der Zeile(n)
3. GROUP BY — Gruppen bilden
4. HAVING — Auswahl(en) der Gruppe(n)
5. SELECT — Ergebnis bilden (*)
6. ORDER BY — Sortieren Ergebnis

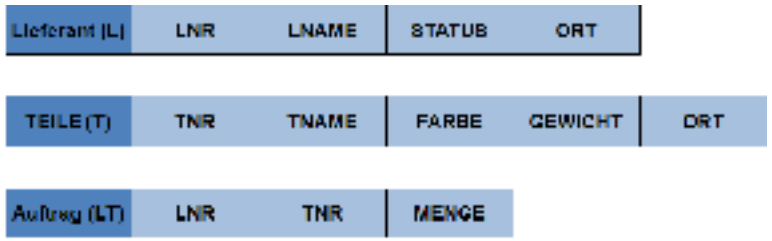
(*) nur Spalten, die in GROUP BY vorkommen oder nur mit COUNT, COUNT DISTINCT, AVG, SUM, MAX, MIN

Verschachtelung und Funktionen

Kombination von Abfragen – Aufgabe

- Aufgabe
  - Liste die Teilenummern der Teile auf, die entweder ein Gewicht über 16 kg haben oder vom Lieferanten L1 geliefert werden.
- Befehl

```
SELECT    TNR
  FROM    T
 WHERE    GEWICHT > 16
 UNION
SELECT    TNR
  FROM    LT
 WHERE    LNR = 'L1'
```



Verschachtelung und Funktionen

Kombination von Abfragen – Ergebnis

- Ergebnis:

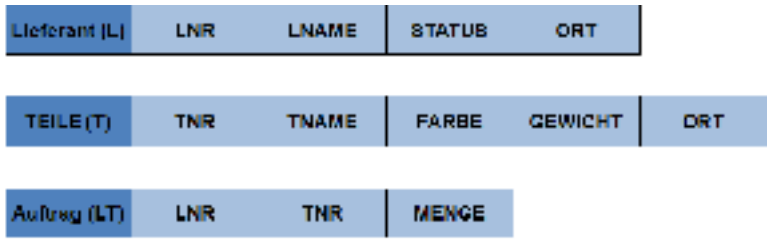
```
TNR
---
T1
T4
T5
```

Verschachtelung und Funktionen

Kombination von Abfragen – Aufgabe

- Aufgabe
  - Liste die Teilenummern der Teile auf, die entweder ein Gewicht über 16 kg haben oder vom Lieferanten L1 geliefert werden.
- Befehl

```
SELECT     TNR
  FROM     T
 WHERE     GEWICHT > 16
 UNION     ALL
SELECT     TNR
  FROM     LT
 WHERE     LNR = 'L1'
```



Verschachtelung und Funktionen

Kombination von Abfragen – Ergebnis

- Ergebnis:

```
TNR
---
T1
T4
T5
T1
T1
```

Verschachtelung und Funktionen

Kombination von Abfragen – Erläuterung UNION / UNION ALL – 1

- Vereinigung der Ergebnisse mehrerer SELECTS
- UNION ALL mit Duplikaten
- Die SELECT-Listen in den verwendeten SELECT-Befehlen müssen die gleiche Anzahl von Elementen enthalten.
- Die n-ten Spalten müssen den gleichen Datentyp haben (char, num, time etc.)
- Haben nummerische Werte unterschiedliche Datenformate, findet eine Konvertierung statt (es gibt Regeln).

Verschachtelung und Funktionen

Kombination von Abfragen – Erläuterung UNION / UNION ALL – 2

- Bei Character-Feldern unterschiedlicher Länge werden die kürzeren mit Blanks aufgefüllt.
- In den SELECT-Listen können auch Konstanten vorkommen, um z.B. die Herkunft der Zeilen erkennbar zu machen.
- Eine ORDER BY Anweisung darf nur im letzten SELECT eingesetzt werden. Das Element mit der Sortierfolge kann nur durch die Position in der SELECT-Liste angegeben werden.

Verschachtelung und Funktionen

Übung(en)

- Kapitel 1.4.5.26 Beispiel 26

Verschachtelung und Funktionen

cps4it

Übung(en)

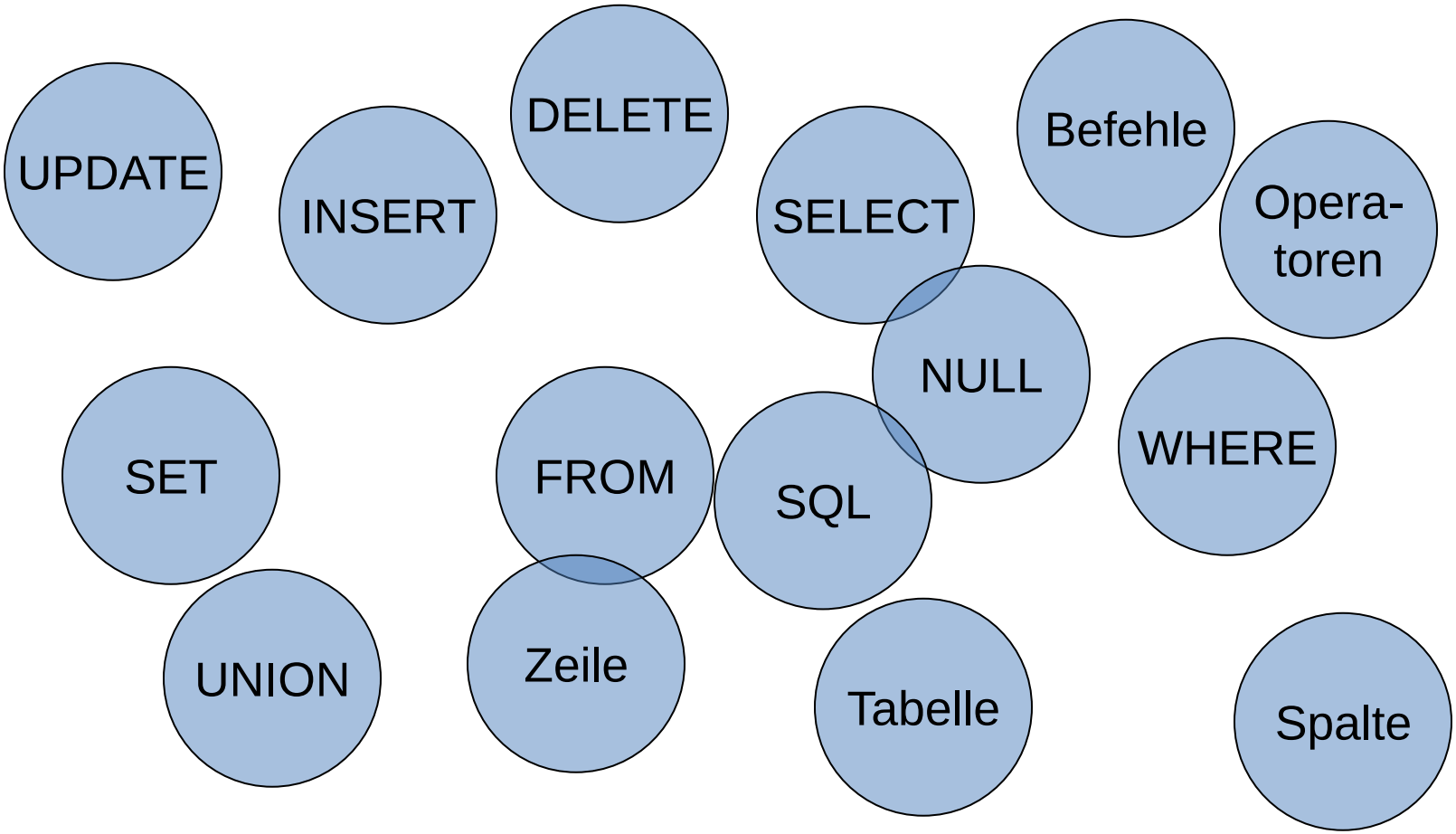
- Kapitel 5.1 Lieferanten mit Status
- Kapitel 5.2 von L1 belieferte Projekte
- Kapitel 5.3 Liste von Teilen mit Daten
- Kapitel 5.4 Namen von Orten > 1 Mal da

Inhalt

- Überblick zum Teil 1
- Abfragen auf 1 Tabelle
- Verknüpfung von Abfragen
- Verschachtelung und Funktionen
- Ändern von Tabellen
- Benutzersicht – View
- Abfrageparameter in Auswahl

Ändern von Tabellen

Begriffe



Ändern von Tabellen

Einfügen von Zeilen

- 1. Variante: einfügen einzelne Zeile

```
INSERT
  INTO tabellenname [(spalte [, spalte] …)]
VALUES (konstante [, konstante ] …)
```

- 2. Variante: einfügen mehrere Zeile

```
INSERT
  INTO tabellenname [(spalte [, spalte] …)]
  unterabfrage
```

Ändern von Tabellen

Einfügen von Zeilen – 1. Variante – Erläuterung

- Für jede Spalte nach dem INSERT muss im VALUES-Teil ein Wert vorhanden sein.
- Beziehung Spalte zu Wert ist 1:1
- Reihenfolge der Spalten ist beliebig.
- Für alle Spalte, die NOT NULL definiert sind, muss ein Wert vorhanden sein.
- Fehlende Werte müssen mit NULL beschrieben werden.
- **Es ist syntaktisch korrekt, den INSERT ohne Spaltennamen zu schreiben. Dies sollte aber ebenso wie ein SELECT * nicht kodiert werden.**

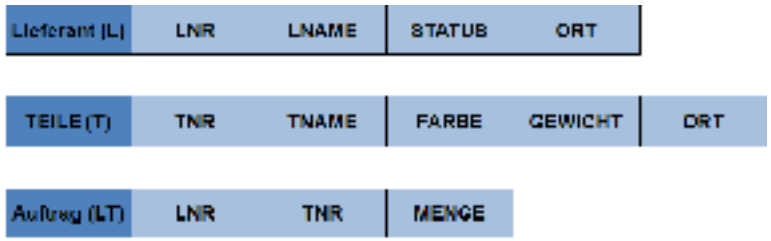
Ändern von Tabellen

Einfügen von Zeilen – 1. Variante – Beispiel

- Befehl:

```
INSERT
  INTO T (TNR, ORT, GEWICHT)
VALUES   ('T7', 'DORTMUND', 21)
```

- Ergebnis:
  - Es gibt eine neue Zeile mit den angegebenen Werten für Teilenummer, Lagerort und Gewicht.
  - Die Spalten TNAME und FARBE erhalten NULL.
- Hinweis: Die Reihenfolge der Felder ist egal.



Ändern von Tabellen

Übung(en)

- Kapitel 1.4.5.27 Beispiel 27

Ändern von Tabellen

Einfügen von Zeilen – 2. Variante

- Die Anzahl der im SELECT-Teil ausgewählten Elemente der SELECT-Liste muss gleich der Anzahl der Spalten sein, die im INTO-Teil angegeben sind.
- Die ausgewählten Elemente der SELECT-Liste müssen den gleichen Datentyp haben wie die Spalten, in die sie eingefügt werden sollen.

```
INSERT
  INTO tabellenname [(spalte [, spalte] …)]
  unterabfrage
```

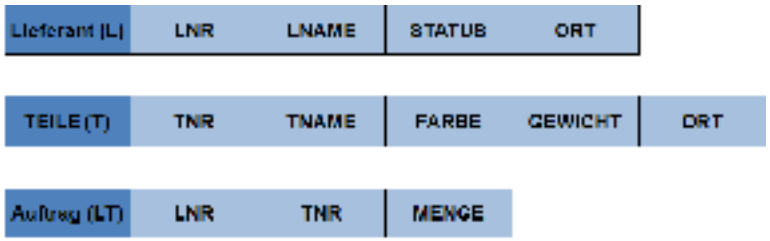
Ändern von Tabellen

Einfügen von Zeilen – 2. Variante – Beispiel

- Befehl:

```
CREATE TABLE TEMP (TNR CHAR(5)
                  , GESMENGE INTEGER);
INSERT
  INTO TEMP (TNR, GESMENGE)
       SELECT TNR, SUM(MENGE)
         FROM LT
        GROUP BY TNR;
```

- Ergebnis:
  - Es gibt eine neue Tabelle mit dem Ergebnis des SELECT.



Ändern von Tabellen

Einfügen von Zeilen – 2. Variante – Erläuterungen

- Die Zwischentabelle TEMP kann vom Benutzer beliebig weiter verwendet werden.
- Die Zwischentabelle kann verändert werden ohne Einfluss auf die Originaldaten (hier LT).
- Die Tabelle existiert bis zu einem Commit.
- Die Tabelle kann durch einen SQL-Befehl wieder gelöscht werden:

```
DROP TABLE TEMP
```

Ändern von Tabellen

Übung(en)

- Kapitel 1.4.5.28 Beispiel 28

Ändern von Tabellen

Ändern von Zeilen

- Alle Zeilen, die die Bedingung(en) im WHERE-Teil erfüllen, werden entsprechend den Angaben im SET-Teil geändert.
- Syntax:

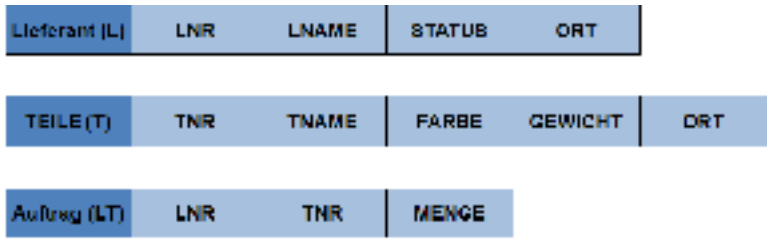
```
UPDATE tabellenname
  SET  spalte = ausdruck
      [spalte = ausdruck ] …
[WHERE bedingungen ]
```

Ändern von Tabellen

Ändern von Zeilen – Beispiel 1

- Aufgabe
  - Ändern der Zeile mit der Teilenummer T6 in der Spalte FARBE nach BLAU, das Gewicht soll um 2 kg erhöht werden und der Ort soll NULL sein.
- Befehl

```
UPDATE     T
  SET      FARBE    = 'BLAU',
           GEWICHT  = GEWICHT + 2,
           ORT      = NULL
 WHERE     TNR      = 'T7'
```



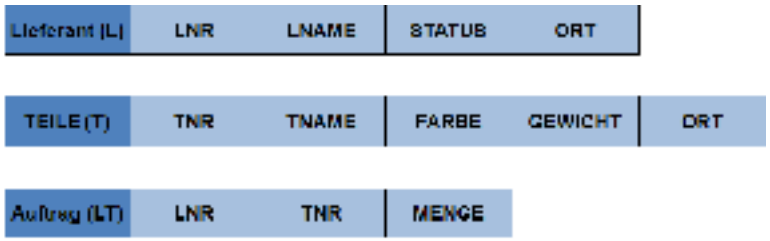
Ändern von Tabellen

Ändern von Zeilen – Beispiel 2

- Aufgabe
  - Verdopple den Status aller Lieferanten in BERLIN.
- Befehl

```
UPDATE     L
  SET      LSTATUS   = 2 * LSTATUS
 WHERE     ORT       = 'BERLIN'
```

- Frage: Geht es auch ohne WHERE-Clause?



Ändern von Tabellen

Übung(en)

- Kapitel 1.4.5.29 Beispiel 29

Ändern von Tabellen

Löschen von Zeilen

- Alle Zeilen, die den angegebenen Bedingungen genügen, werden gelöscht.
- Syntax:

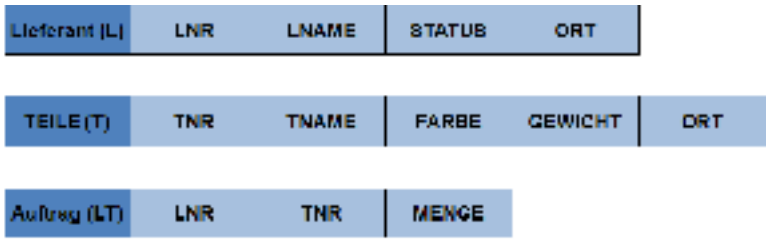
```
DELETE
  FROM tabellenname
 [WHERE bedingungen ]
```

Ändern von Tabellen

Löschen von Zeilen – Beispiel

- Aufgabe
  - Alle Zeilen aus T löschen mit der TNR ‘ T7‘.
- Befehl

```
DELETE
  FROM T
 WHERE TNR = 'T7'
```



Ändern von Tabellen

Übung(en)

- Kapitel 1.4.5.30 Beispiel 30

Ändern von Tabellen

Sprachumfang (in Auswahl) – 1

- aufsuchen
  - SELECT
- modifizieren
  - INSERT, DELETE, UPDATE
- gruppieren
  - GROUP BY, HAVING
- sortieren
  - ORDER BY

Ändern von Tabellen

Sprachumfang (in Auswahl) – 2

- Vergleichsoperatoren
  - = ^= > ^> >= < ^< <=
- Bool’sche Operatoren
  - AND, OR, NOT
- andere Operatoren
  - LIKE, DISTINCT, ANY, ALL, IN, BETWEEN, UNION, EXISTS

Ändern von Tabellen

Sprachumfang (in Auswahl) – 3

- arithmetische Operatoren
  - + - * /
- Verkettungsoperator
  - ||
- built-in-Funktionen
  - Spaltenfunktionen AVG, MAX, MIN, SUM, COUNT
  - skalare Funktionen CHAR, YEAR, LENGTH …

Ändern von Tabellen

Übung(en)

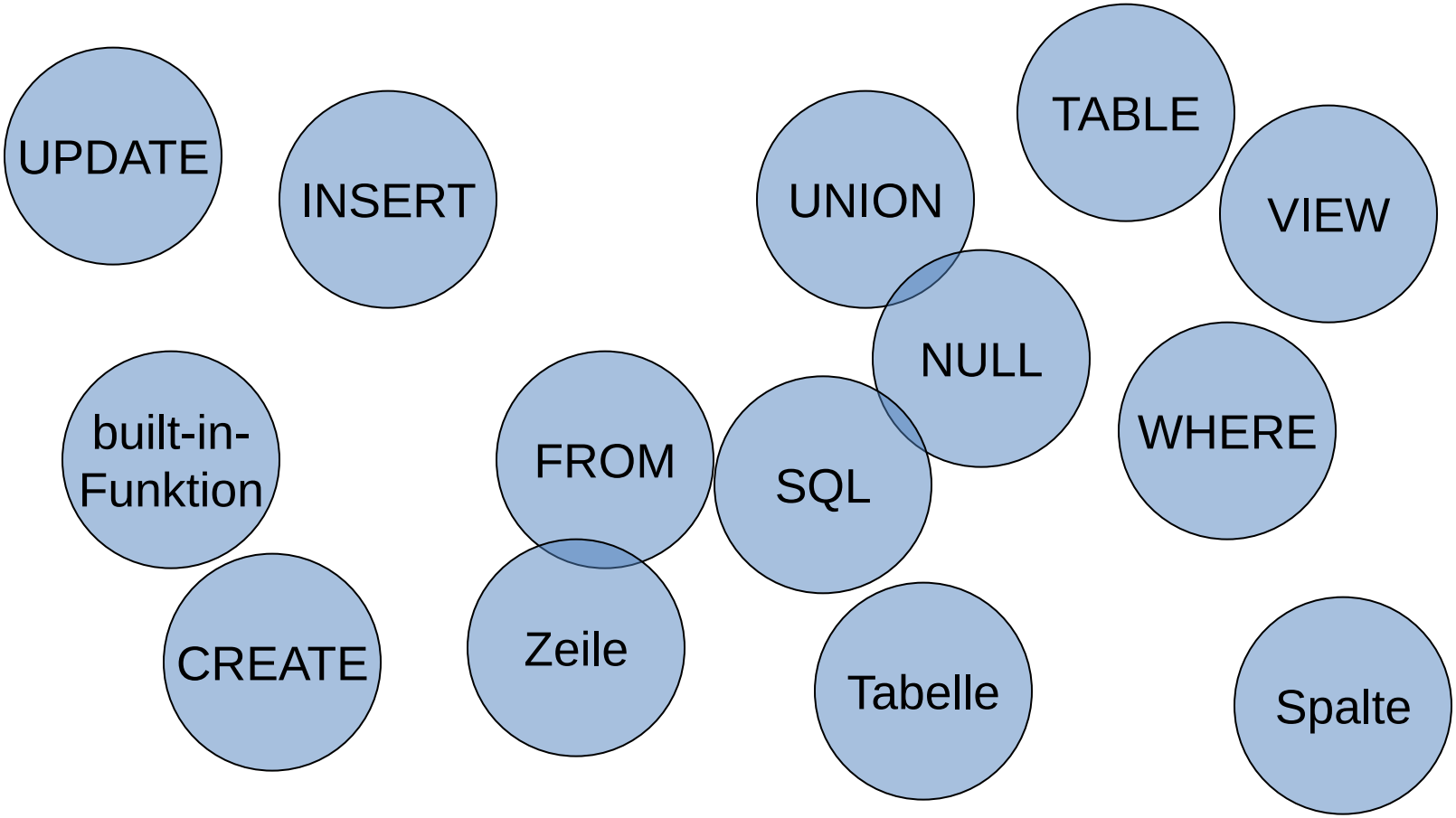
- Kapitel 6.1 Tabelleninhalte erzeugen
- Kapitel 6.2 Zeilen verändern
- Kapitel 6.3 Zeilen löschen
- Kapitel 6.4 Tabelleninhalte löschen

Inhalt

- Überblick zum Teil 1
- Abfragen auf 1 Tabelle
- Verknüpfung von Abfragen
- Verschachtelung und Funktionen
- Ändern von Tabellen
- Benutzersicht – View
- Abfrageparameter in Auswahl

Benutzersicht – View

Begriffe



Benutzersicht – View

Sinn einer View

LNR	LNAME	LSTATUS	ORT
L1	NEUMANN	30	BERLIN
L2	SCHMIDT	20	HAMBURG
L3	KRAUSE	30	HAMBURG
L4	MEIER	10	BERLIN
L5	SCHULZ	20	FRANKFURT

LNR	LSTATUS	ORT
L1	30	BERLIN
L3	30	HAMBURG

„Gute Lieferanten“

Benutzersicht – View

Eigenschaften einer View

- Zusätzliche Darstellungsmöglichkeit der Tabelle(n), aus der sie abgeleitet wird.
- Es entsteht eine “virtuelle” Tabelle.
- Die Definition wird im DB2-Katalog gespeichert.
- Für den Benutzer erscheint die VIEW wie eine normale Tabelle.

Benutzersicht – View

Syntax

```
CREATE VIEW viewname
      [ (spalte [, spalte] …) ]
   AS  unterabfrage
 [WITH CHECK OPTION]
```

- Die Unterabfrage darf weder ORDER BY noch UNION beinhalten.
- Mit der Angabe WITH CHECK OPTION wird bei INSERT und UPDATE geprüft, ob die Werte der WHERE-Bedingung entsprechen.

Benutzersicht – View

Beispiel

```
CREATE VIEW GUTE_L
    AS SELECT LNR, LSTATUS, ORT
         FROM L
        WHERE LSTATUS > 25
```

- Sind keine Spaltennamen angegeben, werden diejenigen aus der Unterabfrage übernommen.
- Spaltennamen sind erforderlich wenn
  - doppelte Namen auftreten (JOIN)
  - built-in-Funktionen genutzt werden
  - arithmetische Ausdrücke genutzt werden

Benutzersicht – View

Beispiel

- Spaltennamen sollten immer benutzt werden wegen
  - Verbesserung der Lesbarkeit
  - Erhöhen der Datenunabhängigkeit bei Namensänderungen in der Originaltabelle
    (In einem solchen Fall muss nur die Definition der View verändert werden und kein einziger SQL.)

Benutzersicht – View

Vorteile von Views

- Datenschutz
  - Nur eine Untermenge der Spalten und/oder Zeilen ist für den Benutzer sichtbar.
- Datenunabhängigkeit
  - Bei Namensänderungen muss nur die Definiton der View verändert werden. Redesign einfacher .
- Vereinfachung für den Benutzer
  - Jeder Benutzer sieht, was er braucht.
- Individuelle Spaltennamen

Benutzersicht – View

Nachteile von Views

- keine Veränderungen sind möglich
  - beim UPDATE, INSERT , DELETE, wenn im CREATE VIEW SUM, MAX, MIN, AVG, DISTINCT, GROUP BY, FROM mit mehr als 1 Tabellennamen (JOIN) enthalten sind.
  - beim UPDATE einer Spalte, wenn im CREATE VIEW diese Spalte einen arithmetischen Ausdruck enthält
- keine Veränderungen sind möglich bei Insert
  - falls im CREATE VIEW ein arithmetischer Ausdruck enthalten ist
  - falls eine Spalte der Ausgangstabelle, die im CREATE VIEW nicht vorkommt, mit NOT NULL definiert ist.

Benutzersicht – View

Übung(en)

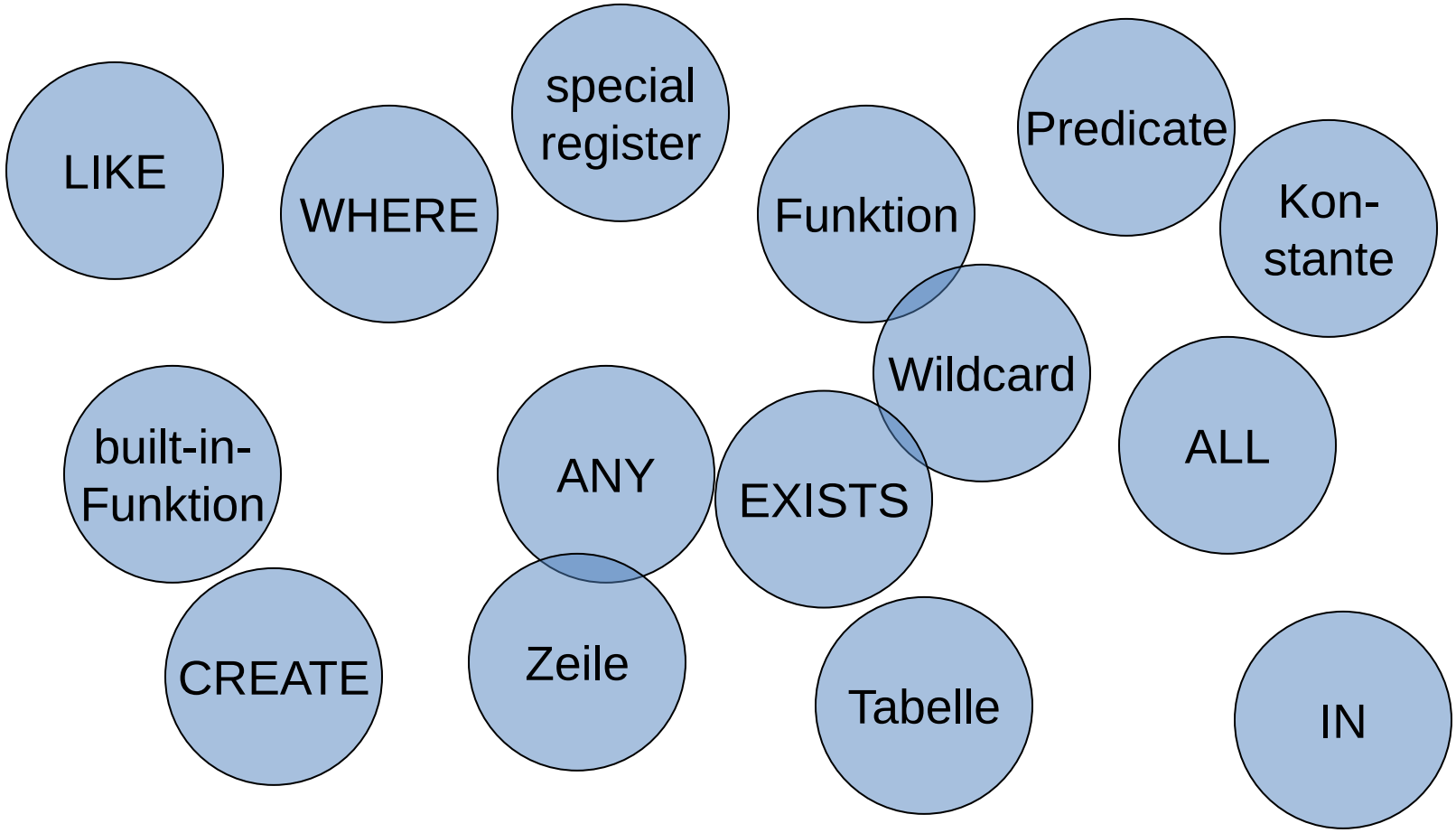
- Kapitel 7.1 Tabelleninhalte erzeugen
- Kapitel 7.2 Spalten verändern
- Kapitel 7.3 Zeilen löschen
- Kapitel 7.4 Zeilen einfügen
- Kapitel 7.5 View erstellen

Inhalt

- Überblick zum Teil 1
- Abfragen auf 1 Tabelle
- Verknüpfung von Abfragen
- Verschachtelung und Funktionen
- Ändern von Tabellen
- Benutzersicht – View
- Abfrageparameter in Auswahl

Abfrageparameter in Auswahl

Begriffe



Abfrageparameter in Auswahl

weitere Begriffe – LIKE

- Benutzung als Predicate (WHERE)
- Konstante
- special register
- skalare Funktion (mit Argumenten von “oben”)
- Ausdruck (mit Argumenten von “oben”)
- Wildcards (%_)

Abfrageparameter in Auswahl

weitere Begriffe – ANY / EXISTS

- ANY
  ausdruck IN (irgendwas)
  ist identisch zu
  ausdruck = ANY (irgendwas)

- EXISTS
  Benutzung als Predicate
  WHERE EXISTS (subselect)
  WHERE NOT EXISTS (subselect)

Abfrageparameter in Auswahl

weitere Begriffe – ALL

- ausdruck NOT IN (irgendwas)
  ist identisch zu
  ausdruck <> ALL (irgendwas)
- Beispiele
  - COUNT (HUGO) = COUNT (ALL HUGO)
  - SUM (ALL HUGO) <> SUM (DISTINCT HUGO)
  - SELECT ALL <> SELECT DISTINCT
  - UNION ALL

Abfrageparameter in Auswahl

special register

- CURRENT DATE
- CURRENT SQLID
- CURRENT TIME
- CURRENT TIMESTAMP
- CURRENT TIMEZONE
- etc.

Abfrageparameter in Auswahl

und vieles mehr …

- CALL – Aufruf Stored Procedure
- CASE / WHEN – komplexe Bedingungen bei WHERE
- Schema – Menge von Objekten
  - distinct types, Funktionen, Stored Procedures, Trigger
- Trigger – automatische Aktion
- Stored Procedure – Programm mit SQLs

Abfrageparameter in Auswahl

cps4it

weiter gehende Informationen

- Stored Procedure
  - DB2 UDB z/OS V10 Application Programming and SQL Guide
    Kapitel 10 An example of a simple stored procedure

    http://publib.boulder.ibm.com/epubs/pdf/dsnapm03.pdf

    http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.apsg/src/tpc/db2z_storedprocedure.htm

Abfrageparameter in Auswahl

cps4it

weiter gehende Informationen

- Trigger
  - DB2 for z/OS V10 Application Programming and SQL Guide
    Kapitel 10 Example of creating and using a trigger

    http://publib.boulder.ibm.com/epubs/pdf/dsnapm03.pdf

    http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.apsg/src/tpc/db2z_createmodifydb2objects.htm

Abfrageparameter in Auswahl

weiter gehende Informationen

- Isolation level
  - DB2 for z/OS V10 Performance Paper
    Kapitel zum Thema concurrency

    http://publib.boulder.ibm.com/epubs/pdf/dsnapm03.pdf

    http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.dshare/src/tpc/db2z_improveconcurrencyds.htm

Abfrageparameter in Auswahl

weiter gehende Informationen

- Hinweise für Performance
  - DB2 for z/OS Managing Performance

    http://publib.boulder.ibm.com/epubs/pdf/dsnpgm03.pdf

    http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.perf/src/perf/db2z_perf.htm

    (ein Paper von vielen)