Einführung in die Wirtschaftsinformatik

Teil 11 – SQL Fragestunde

Wintersemester 2023/2024

Lehrstuhl für Wirtschaftsinformatik
Prozesse und Systeme
Universität Potsdam

Chair of Business Informatics
Processes and Systems
University of Potsdam

Univ.-Prof. Dr.–Ing. habil. Norbert Gronau
Lehrstuhlinhaber | *Chairholder*

Mail August-Bebel-Str. 89 | 14482 Potsdam | Germany
Visitors Digitalvilla am Hedy-Lamarr-Platz, 14482 Potsdam
Tel +49 331 977 3322

E-Mail ngronau@lswi.de
Web lswi.de

NULL Werte

Multiple Row Funktionen

Having

In Ausdrücken rechnen

Self JOIN

Wiederholung Vorlesung 7: NULL-Werte in Feldern

- In der letzten VL Probleme mit NULL Werten:
- Auch beim Rechnen mit Nullwerten treten Probleme auf

```
SELECT name, vorname, akad_titel
FROM mitarbeiter;
```

NAME	VORNAME	AKAD_TITEL
Walker	John A.	
Melzer	Thomas	
Bormann	Samira	Dr.
Hofmann	Katja	
Würz	Hannah	
…		
Petersen	Helmut	
Schulz-	Paul	Dr. Ing.
Plenk	Karl	
Engel	Lothar	
Roth	Katharina	
…		

Leerwerte:

Felder ohne Inhalte

Dieses Problem hatten wir in einer der vorigen VL!

```
SELECT name, Vorname, gehalt*12*provision Jahresgehalt
FROM mitarbeiter;
```

NAME	VORNAME	JAHRESGEHALT
Christ	Adrien	2880
Mehmedovic	Ahmad	-
Winter	Tanja	-
…	…	…
Malossek	Benjamin	1886,4

NULL*gehalt*12
= NULL

Greenberg 2002, S. 64

IS NULL für den Test auf Nullwerte

Zellen ohne Werte mit IS NULL

```
SELECT name, position, abt_nr
FROM mitarbeiter
WHERE abt_nr IS NULL;
```

NAME	POSITION	ABT_NR
Riekhoff	Chefsekretärin	-
Johansson	Chief Operations	
Metz	Chefsekretärin	…
…	…	-
Walker	Chief Information	-

Zellen ohne Werte mit anderen Operatoren

```
SELECT name, position, abt_nr
FROM mitarbeiter
WHERE abt_nr  = '';
```

```
No data found
```

Behandlung von NULL-Werten

Angabe konkreter Werte in `ausdruck`

- NVL (ausdruck1, ausdruck2|wert)
- NVL2 (ausdruck1, ausdruck2|wert1, ausdruck3|wert2)
- COALESCE (ausdruck1, ausdruck2, ..., ausdruckn)

Vergleich und Ausgabe von `ausdruck` oder NULL-Wert

- NULLIF (ausdruck1, ausdruck2)

Diese Funktionen können für alle Datentypen eingesetzt werden.

Quelle: Greenberg 2002, S. 167

Funktion NVL

- Konvertierung von NULL-Werten in konkrete Werte bei DATE, CHARACTER, NUMBER
- Forderung – Übereinstimmung der Datentypen

```
NVL(ausdruck1, ausdruck2)
```

Datentyp NUMBER

```
Beispiel 1: NVL(gehalt,3300)
```

```
Beispiel 2: NVL(proj_kosten,0)
```

Datentyp CHAR oder VARCHAR2

```
Beispiel 3: NVL(proj_name,'Nicht verfügbar')
```

```
Beispiel 4: NVL(position,'Transportarbeiter')
```

Quelle: Greenberg 2002, S. 168f.

Funktion NVL mit numerischem Rückgabewert

```
SELECT name, gehalt, NVL(provision,0)provision,
(gehalt*12*(1+NVL(provision,0))) Jahresgehalt
FROM mitarbeiter;
```

1

2

- Berechnung Jahresgehalt aller Angestellten
- Multiplikation von Gehalt, Anzahl Monate und Provisionssatz
- Problemstellung: Provisionssatz nur für Verkäufer --> alle anderen Felder der Spalte sind leer

NAME	GEHALT	PROVISION	JAHRESGEHALT
Büchner	10430	0	125160
Martens	2400	0	28800
Dost	3100	0	37200
Fuchs	3600	0	43200
...	...	...	...
Johnson	1540	0,15	21252
Poderni	1380	0,14	18878,4
Pommer	1460	0,13	19797,6
...	...		...
Altmann	1830	1 0	2 21960

Anwendung der Funktion NVL2

Festlegung des Rückgabewerts durch Inhalt des ersten Ausdrucks

- Rückgabe eines NULL-Wertes – Ausgabe des dritten Ausdrucks von NVL2
- Rückgabe von Werten – Ausgabe des zweiten Ausdrucks von NVL2

```
NVL2(ausdruck1, ausdruck2, ausdruck3)
```

```
SELECT name, gehalt, abt_nr, NVL(provision,0),
NVL2(provision,'Gehalt + Provision','Gehalt') Einkommen
FROM mitarbeiter WHERE abt_nr IN ('410V','107R');
```

Ersetzt bei einer Provision NULL den NVL Value durch 0

Wird ausgegeben wenn Provision NULL

Wird ausgegeben wenn Provision nicht NULL

NAME	GEHALT	ABT_NR	NVL(PROVISION,0)	EINKOMMEN
Probst	4510	410V	0	Gehalt
Peplinski	3050	410V	0	Gehalt
Petrova	2100	410V	0	Gehalt
De Ridder	5890	410V	0	Gehalt
Schöneck	1380	410V	0,13	Gehalt +
Thyssen	1480	410V	0,13	Gehalt +
…	…	…	…	…

Funktion NULLIF

Vergleich von zwei Ausdrücke

- Bei Gleichheit – Ausgabe NULL-Wert
- Bei Ungleichheit – Ausgabe ausdruck1

```
NULLIF(ausdruck1, ausdruck2)
```

Beispiel

```
SELECT name, NULLIF(position,'Abteilungsleiter') "Nicht
leitende Angestellte" FROM mitarbeiter;
```

Wenn position = 'Abteilungsleiter' wird Nullwert zurückgegeben

NAME	Nicht leitende Angestellte
...	...
Dost	Einkäufer
Fuchs	Sekretärin
Rösch	-
Beyer	-
Kettler	-
Schneider	Schleifer
...	...

NULL-Werte

Beyer's Position = 'Abteilungsleiter' —> Rückgabewert ist NULL

Funktion COALESCE

- Dient der Vermeidung von NULL-Werten
- Liefert aus Parameterliste den Wert eines Parameters zurück, der nicht NULL ist
- Wenn erster Ausdruck kein NULL-Wert – Rückgabe dieses Ausdrucks
- Rückgabe von ausdruck2,...,n dann, wenn vorhergehender Ausdruck NULL-Wert enthält

```
COALESCE(ausdruck1, ausdruck2, ... ausdruckn)
```

Liefert aus Parameterliste den Wert eines Parameters zurück, der nicht NULL ist

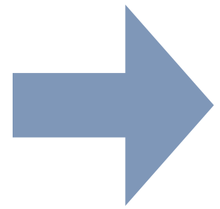
Wird zurückgegeben wenn ausdruck1 NULL ist

Wird zurückgegeben wenn Ausdrücke 1 bis n-1 alle NULL sind

Vorteil der Funktion COALESCE gegenüber der Funktion NVL ist die Angabe von mehr als zwei alternativen Werten.

Anwendung der Funktion COALESCE

GEHALT	PROVISION
3600	
9000	0,065
2100	
1300	0,065



Jahreseinkommen
43.200
433.000
25.200
340.600

- Verschiedene Mitarbeiter beziehen ihr Jahresgehalt aus unterschiedlichen Quellen (reines Gehalt vs. Gehalt + Provision)
- Problem: Wir können keine einheitliche Spalte nutzen, um das Jahresgehalt auszurechnen - je nach Einkommensart müssen wir unterschiedliche Spalten nutzen

```
SELECT ... COALESCE(12*gehalt + 5000000*provision,
12*gehalt) "Jahreseinkommen" FROM mitarbeiter;
```

➡ Hinweis: Wir gehen davon aus, dass die Firma einen gesamten Provisionsumsatz von 5.000.000 hat, auf welchen auch die Provision berechnet wird

COALESCE() nimmt eine beliebig lange Liste von Werten an und gibt den ersten Wert ungleich NULL zurück.

Funktionen verschachteln

- Beliebige Verschachtelungstiefe der Single Row-Funktionen
- Auswertung der Funktionen erfolgt von innen nach außen

```
Fkt2(Fkt1(spalte,argument1),argument2)
```

Schritt 1 = Ergebnis 1

Schritt 2 = Ergebnis 2

```
SELECT name, NVL(TO_CHAR(leiter), 'Ohne Vorgesetzten')Vorgesetzter
FROM mitarbeiter WHERE leiter IS NULL;
```

NAME	VORGESETZTER
Kellner	Ohne Vorgesetzten
Reinhard	Ohne Vorgesetzten

Verschachtelte Funktionen werden grundsätzlich durch runde Klammern getrennt.

Quelle: Greenberg 2002, S. 165

NULL Werte

Multiple Row Funktionen

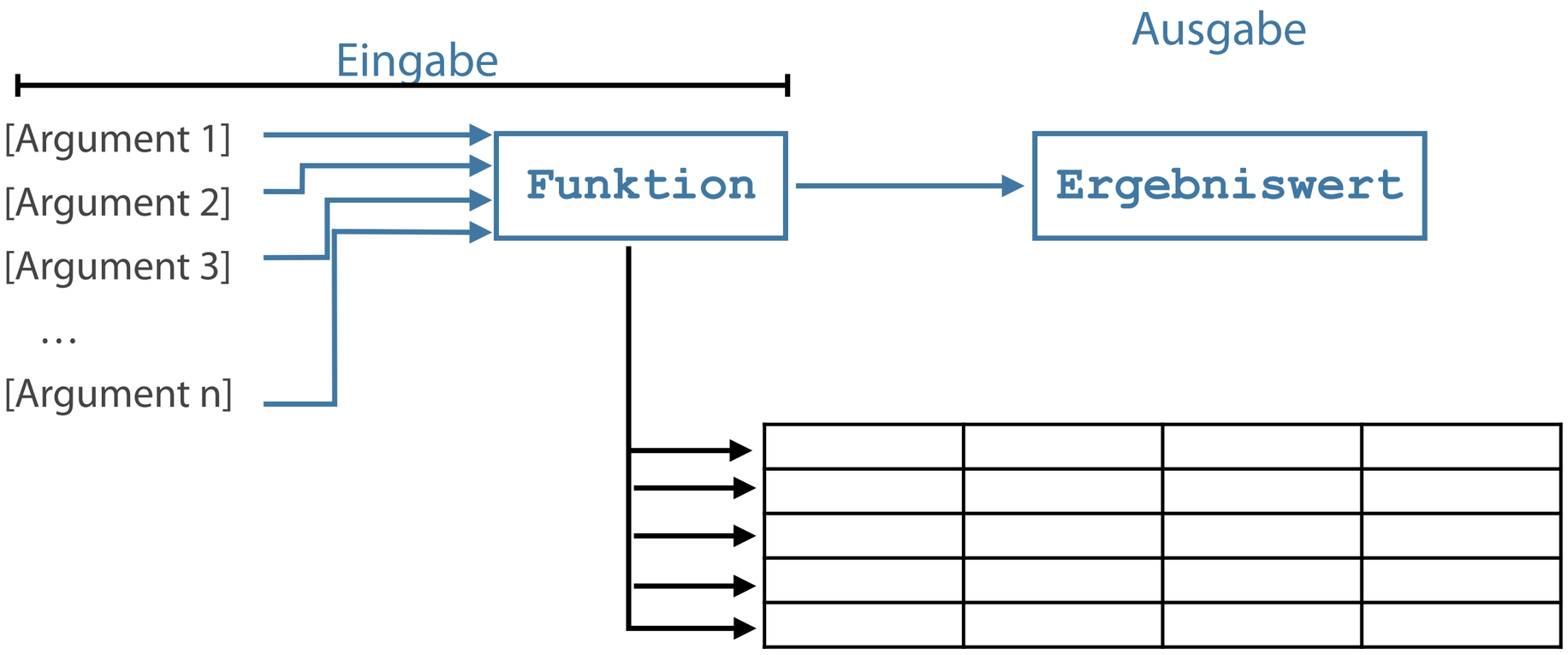
Having

In Ausdrücken rechnen

Self JOIN

SQL-Funktionen

- Bearbeitung von Zeilen und Ausgabe von Ergebnissen dieser Bearbeitung



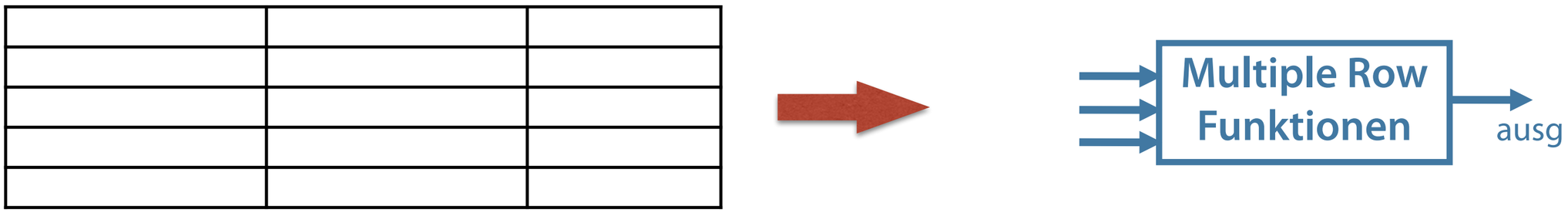
SQL-Funktionen enthalten manchmal Argumente und geben immer einen Wert zurück.

Quelle: Greenberg 2002, S. 125

Unterschied zwischen Single-Row und Multiple-Row-Funktionen

Single Row Funktion – Rückgabe – ein Ergebnis pro Zeile

Multiple Row Funktion – Rückgabe – ein Ergebnis pro Zeilengruppe



arg – Argument
ausg – Ausgabe

Multiple-Row-Funktionen werden bei verschachtelten SQL-Abfragen benötigt, um Funktionen auf eine Zeilengruppe anwenden zu können.

Quelle: Greenberg 2002, S. 126

Merkmale von Single Row Funktionen

Datenelemente

- Bearbeitung jeder zurückgegebenen Zeile aus einer Hauptabfrage

Argumente und Werte

- Spalten oder Ausdrücke – als Argumente akzeptiert
- Rückgabe eines Ergebnisses als Wert je Zeile

Erläuterung der Syntax

- funktionsname – Name der Funktion
- argument1, argument2 – die von der Funktion verwendeten Argumente (Spaltenname, Ausdruck)

```
funktionsname [(argument1, argument2,...)];
```

Quelle: Greenberg 2002, S. 127

Übersicht Single Row Funktionen

Zeichenfunktionen

- Rückgabe von Zeichen- oder numerischen Werten

Konvertierungsfunktionen

- Konvertierung eines Wertes von einem Datentyp in einen anderen

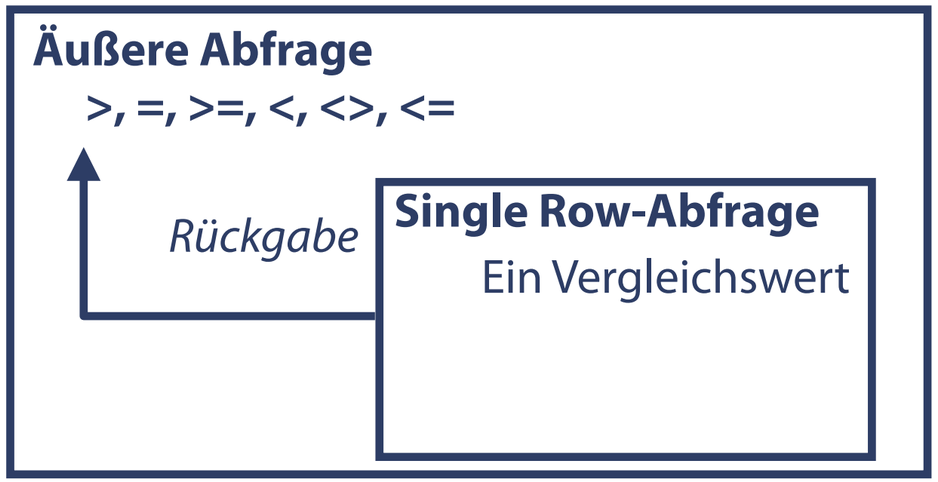
Numerische Funktionen

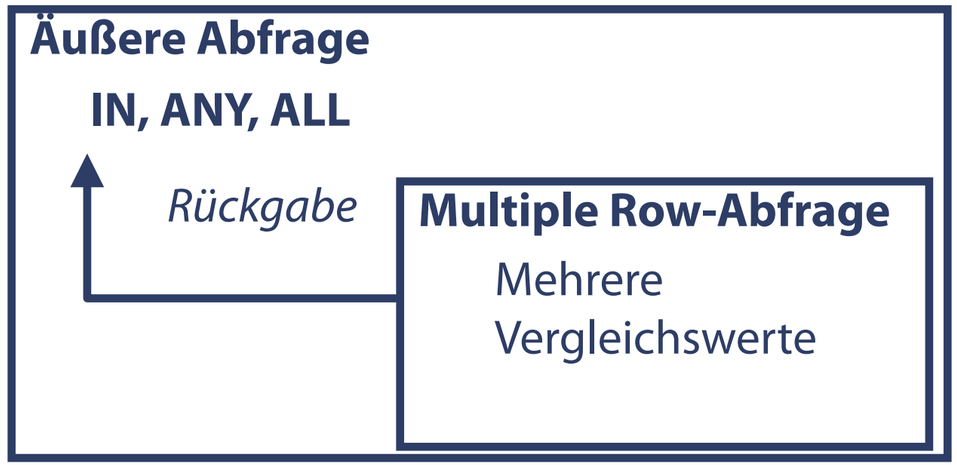
- Rückgabe numerischer Werte

Datumsfunktionen

- Rückgabe eines Wertes vom Datentyp DATE

Operatoren in Unterabfragen





Syntax

- Unterabfragen grundsätzlich in Klammern
- Vergleichsoperator vor (links von) Unterabfrage
- Anwendung ORDER BY-Klausel in Unterabfrage bei Realisierung einer Top-N-Analyse (Abfrage mit Ranking)

Vergleichsoperatoren

Single Row-Unterabfragen

größer als	kleiner als	gleich	ungleich	größer oder gleich	kleiner oder gleich
>	<	=	<>	>=	<=

Multiple Row-Unterabfragen

Gleich einem Element aus der Liste	Vergleich mit jedem von Unterabfrage zurück-gegebenen Wert	Vergleich mit allen von Unterabfrage zurück-gegebenen Werten
IN	ANY	ALL

Multiple-Row Anfragen

Beispiel – Rückgabe mehrerer Werte

- Wie heißen die Mitarbeiter in den Abteilungen, in denen Personen mit dem Namen Grimm arbeiten?

```
SELECT name, vorname, abt_nr
FROM mitarbeiter
WHERE abt_nr IN
      (SELECT abt_nr FROM mitarbeiter
      WHERE name = 'Grimm')
ORDER BY abt_nr, name;
```

NAME	VORNAME	ABT_NR
Adler	Jana	100V
Grimm	Bernd	100V
Hofmann	Katja	100V
Melzer	Thomas	100V
Sonntag	Christof	100V
Walther	Stefanie	100V
Grimm	Alexander	260Z
…	…	…

Kombination mehrerer Single-Row Unterabfragen

```
SELECT name, position, gehalt
FROM mitarbeiter
WHERE position =  (SELECT position
                   FROM mitarbeiter
                   WHERE pers_nr = 101003)
AND gehalt >      (SELECT gehalt
                   FROM mitarbeiter
                   WHERE pers_nr = 101107)
;
```

Einkäufer

2800

NAME	POSITION	GEHALT
Dost	Einkäufer	3100
Petersen	Einkäufer	2890

In der WHERE-Klausel können auch mehrere innere Abfragen nacheinander verwendet werden.

NULL-Werte in einer Unterabfrage – Ungeeignete Multiple Row-Funktion

Problem

- Abgefragte Spalte in Unterabfrage enthält mindestens einen NULL-Wert -->
- Hauptabfrage kann keine Ergebnistabelle erzeugen

Bedingungen, die einen NULL-Wert vergleichen…

- …liefern einen NULL-Wert zurück
- Entspricht der Wirkung von “<> ALL“

```
SELECT name
FROM mitarbeiter
WHERE pers_nr NOT IN
(SELECT leiter
FROM mitarbeiter);
```

```
No data found
```

NULL Werte

Multiple Row Funktionen

Having

In Ausdrücken rechnen

Self JOIN

Syntax der HAVING-Klausel

Formulierung von Bedingungen für Gruppen – Abarbeitungsreihenfolge

1. Prüfung der WHERE-Bedingung für jeden einzelnen Datensatz
2. Gefilterte Datensätze werden gruppiert
3. Anzeige der Gruppendatensätze entsprechend der HAVING-Klausel

```
SELECT spalte, gruppenfunktion
FROM tabelle
[WHERE bedingung]
[GROUP BY group_by_ausdruck]
[HAVING gruppenbedingung]
[ORDER BY spalte];
```

Mögliche Funktionen innerhalb der Bedingung:
MIN, MAX, SUM, COUNT, AVG

HAVING wirkt ausschließlich bei Einschränkungen nach Gruppenfunktionen.

Quelle: Elmazri 2002, S. 302

Einschränkungen mit Hilfe der HAVING-Klausel

Funktion mit numerischem Argument

- Abfrage des Durchschnitts aller Gehälter innerhalb jeder einzelnen Abteilung, das zwischen 4000 und 5000 Euro liegt.

```
SELECT abt_nr, AVG(gehalt) Durchschnittsgehalt
FROM mitarbeiter
GROUP BY abt_nr
HAVING AVG(gehalt) BETWEEN 4000 AND 5000;
```

ABT_NR	DURCHSCHNITTSGEHALT
420F	4900
620W	4296
250A	4262,5
320M	4473,333333333333333333333
310T	4607,5
610A	4322,5
...	...

Aggregation und Filterung

GEHALT	ABT_NR
4490	310V
1310	110V
1380	210V
4515	210V
1310	110V
7330	10VL
4810	410V
1580	110V
1400	210V
2104	10VL
...	...
6300	630E
3400	100V
3100	240P
4400	240P
2740	260F
...	...

1 Suche in der Tabelle "MITARBEITER" das minimale Gehalt innerhalb jeder Abteilung.

1 Aggregation (GROUP BY)

GEHALT	ABT_NR
1310	310V
1360	210V
3000	-
2400	260M
1310	110V
2570	210E
2500	520G
1380	410V
2060	260D
...	...

2 Gib nur die aus, deren Wert unter 2800 Euro liegt.

Ergebnisse durch Selektion gruppierter Werte ausgeblendet

3 Filterung (HAVING)

GEHALT	ABT_NR
1310	310V
1360	210V
2400	260M
1310	110V
2570	210E
2500	520G
...	...

Die HAVING-Klausel filtert die anzuzeigenden Tabellenzeilen in der Gruppierung.

NULL Werte

Multiple Row Funktionen

Having

In Ausdrücken rechnen

Self JOIN

Arithmetische Operatoren

- Zusätzliche Spalte mit Berechnungsergebnis
- Existiert nicht in der Tabelle "mitarbeiter"

```
SELECT name, gehalt, gehalt * 1.02 Gehaltserhöhung
FROM mitarbeiter;
```

NAME	GEHALT	GEHALTSERHÖHUNG
Büchner	10430	10638,6
Martens	2400	2448
Dost	3100	3162
Fuchs	3600	3672
Rösch	6590	6721,8
…	…	…

Zur besseren Lesbarkeit können Leerzeichen vor und nach dem arithmetischen Operator eingefügt werden.

Operatorpriorität

- Punkt- vor Strichrechnung: Multiplikationen und Divisionen vor Additionen und Subtraktionen
- Auswertung von Operatoren derselben Priorität von links nach rechts
- Einsatz von Klammern zur Priorisierung der Auswertung
- Bessere Lesbarkeit von Anweisungen

*** / + -**

a+b+c

a + b * c <> (a + b) * c

Greenberg 2002, S. 61

Berechnung mit priorisierten Operatoren

- Berechnung des Jahresgehaltes plus Einmalzahlung

```
SELECT name, gehalt, 50 + gehalt * 12
FROM mitarbeiter;
```

oder alternativ

```
SELECT name, gehalt, 50 + (gehalt * 12 )
FROM mitarbeiter;
```

NAME	GEHALT	50+GEHALT*12
Büchner	10430	125210
Martens	2400	28850
Dost	3100	37250
Fuchs	3600	43250
Rösch	6590	79130
…	…	…

Gruppenfunktion und NULL-Wert

Auswahl aller Werte ohne NULL-Werte

- Berechnung durchschnittliches Einkommen in Abhängigkeit von der Provision

```
SELECT AVG(gehalt * (1 + provision))
Durchschnittseinkommen
FROM mitarbeiter;
```

DURCHSCHNITTSEINKOMMEN
1747,5962962962962

Auswahl aller Werte inklusive von NULL-Werten – Funktion NVL

- Ersetzen leerer Werte durch Vorgabewerte
- Funktion liefert für alle Zeilen ein brauchbares Ergebnis

```
SELECT AVG(gehalt * (1 + NVL(provision,0)))
Durchschnittseinkommen
FROM mitarbeiter;
```

DURCHSCHNITTSEINKOMMEN
4373,1105

Quelle: Greenberg 2002, S. 243

NULL Werte

Multiple Row Funktionen

Having

In Ausdrücken rechnen

Self JOIN

Self-Join

Verknüpfung einer Tabelle mit sich selbst

- Zusätzliche Bezeichner/Alias-Namen in der FROM-Klausel zur Unterscheidung

```
SELECT a.pers_nr AS "Pers_Nr (Angestellte)", a.name AS
"Angestelltenname", a.anrede AS "Anrede", a.leiter AS
"Leiter(Pers_nr)", b.name AS "Name des Leiters", b.anrede
AS "Anrede"
FROM mitarbeiter a, mitarbeiter b
WHERE a.leiter = b.pers_nr
ORDER BY a.pers_nr;
```

Pers_Nr (Angestellte)	Angestelltenname	Anrede	Leiter (Pers_Nr)	Name des Leiters	Anrede
101001	Büchner	Herr	101060	Köhler	Herr
101002	Martens	Herr	101060	Köhler	Herr
101003	Dost	Herr	101059	Ernst	Herr
101004	Fuchs	Frau	101060	Köhler	Herr
101005	Rösch	Herr	101047	Klemm	Frau
…	…	…	…	…	…

Ein Self-Join ermöglicht die Herstellung von Verbindungen innerhalb einer Tabelle mit einer einzigen Abfrage.

Syntax für Self-Joins

Formale Betrachtung auf zwei Seiten einer Tabelle

```
SELECT tab1.spalte, ..., tab2.spalte
FROM tabelle1 tab1, tabelle1 tab2
WHERE tab1.spalte1 = tab2.spalte2;
```

Wer ist der/die Vorgesetzte jedes Mitarbeiters?

```
SELECT leiter.anrede || ' ' || leiter.name || ' ist
Vorgesetze/r  von ' || angestellte.anrede || ' ' ||
angestellte.name
FROM mitarbeiter angestellte, mitarbeiter leiter
WHERE angestellte.leiter = leiter.pers_nr;
```

LEITER.NAME||'ISTVORGESETZE/RVON'||ANGESTELLTE.NAME
…
Herr Schmiedel ist Vorgesetze/r von Herr Beyerke
…

Kontrollfragen

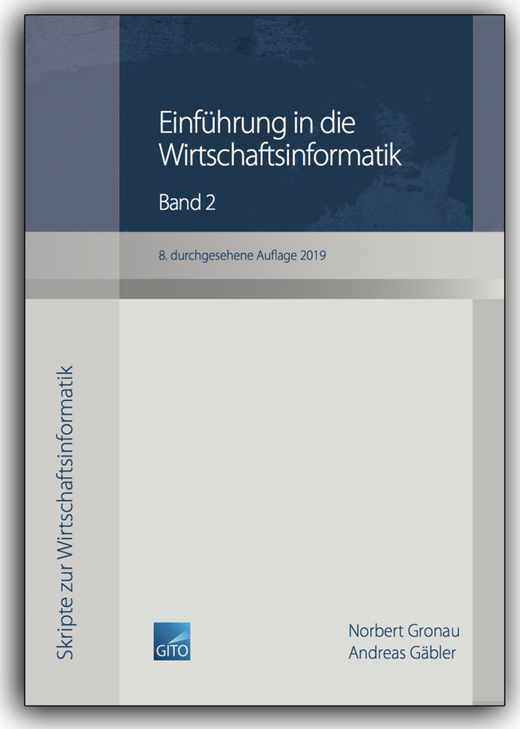
- Wann werden CASE-Ausdrücke in SQL-Anweisungen verwendet?
- Nach der Normalisierung sind ursprüngliche Tabellen oftmals in mehrere neue Tabellen aufgeteilt. Mit welcher Funktion können die Daten daraus wieder miteinander verbunden werden?
- Unter welchen Bedingungen wird ein kartesisches Produkt erzeugt?
- Was bewirkt die Verwendung eines LEFT OUTER JOIN in einer Anweisung?

Literatur

Elmazri, R./Navathe, S. B.: Grundlagen von Datenbanksystemen; 3. Auflage, 2002, Addison-Wesley

Greenberg. N./Nathan, P: Professioneller Einstieg in Oracle9i SQL - Band 1; 2002, Oracle

Zum Nachlesen



Gronau, N., Gäbler, A.:
Einführung in die Wirtschaftsinformatik, Band 2

8. überarbeitete Auflage
GITO Verlag Berlin 2019, ISBN 978-3-95545-285-8

Kontakt

Univ.-Prof. Dr.-Ing. Norbert Gronau

Universität Potsdam
Karl-Marx-Str. 67 | 14482 Potsdam
Germany

Tel. +49 331 977 3322
E-Mail ngronau@lswi.de