Einführung in die Wirtschaftsinformatik

Teil 9 – SQL-Gruppenfunktionen

Wintersemester 2020/2021

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*

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

Tel +49 331 977 3322
Fax +49 331 977 3406

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

Aggregierung von Daten

Filterung von Gruppenergebnissen

Unterabfragen (Subqueries)

Gruppenfunktionen

Erzeugung aggregierter Informationen

- Aggregationsfunktionen bezogen auf Gruppen von Zeilen
- Rückgabe – NUR EIN Ergebnis pro Gruppe

```
SELECT MAX(gehalt)
FROM mitarbeiter;
```

PERS_NR	GEHALT
101001	10430
101002	2400
101003	3100
101004	3600
101005	6590
101006	7770
101007	8080
101008	3800
101009	4100
101010	4000
...	...
101064	3400
101065	56000
...	...
101135	2910
101136	2550

Suche das höchste Gehalt in der Tabelle "MITARBEITER"

MAX(GEHALT)
56000

Typen von Gruppenfunktionen

Gruppenfunktion	**Wirkung**
AVG([Filter] spalte)	Durchschnittswert
COUNT({*|[Filter] ausdruck})	Anzahl Zeilen
MAX([Filter] ausdruck)	Höchster Wert
MIN([Filter] ausdruck)	Kleinster Wert
SUM([Filter] spalte)	Summe
Ausdruck für Filter	
ALL	Berücksichtigung aller Werte (Standard)
DISTINCT	Keine Berücksichtigung doppelter Werte

Berechnung numerischer Werte

Gruppenfunktionen mit Rückgabe numerischer Werte

- Bildung von Werten jeweils aus einer Spalte

```
SELECT AVG(gehalt), MAX(gehalt), MIN(gehalt), SUM(gehalt)
FROM mitarbeiter
WHERE anrede = 'Frau';
```

AVG(GEHALT)	MAX(GEHALT)	MIN(GEHALT)	SUM(GEHALT)
3912,310810810810810810810	35000	500	289511

```
SELECT AVG(net_preis), MAX(net_preis), MIN(net_preis)
FROM artikel;
```

AVG(NET_PREIS)	MAX(NET_PREIS)	MIN(NET_PREIS)
27273,2	190000	73

Die Funktionen AVG und SUM sind nur bei numerischen Werten zulässig.

Gruppenfunktionen und Textwerte

Gruppenfunktionen auf Zeichenketten und Datumswerten

- MIN, MAX mit allen Datentypen möglich
- Textwerte nach alphabetischer Reihenfolge

```
SELECT MIN(name), MAX(name)
FROM mitarbeiter;
```

MIN(NAME)	MAX(NAME)
Abei	von Bingen

```
SELECT MAX(geburtstag) "Geburtstag jüngster Ma.", MIN(geburtstag) "Geburtstag ältester Ma."
FROM mitarbeiter;
```

Geburtstag jüngster Ma.	Geburtstag ältester Ma.
02.10.2001	04.07.1955

Zählfunktion

- COUNT() – listet Anzahl der Datensätze je Gruppe auf
- COUNT(*) – Anzahl aller Zeilen in einer Tabelle
- COUNT([Filter] ausdruck) – Anzahl der mit "ausdruck" übereinstimmenden Werte
  [Filter]: DISTINCT – Keine Berücksichtigung doppelter und NULL-Werte

```
SELECT COUNT(*)
FROM mitarbeiter
WHERE anrede = 'Frau';
```

COUNT(*)
74

```
SELECT COUNT(*) FROM bestellung
WHERE SUBSTR(bestelldatum,7,4) =
'2017';
```

COUNT(*)
29

COUNT-Abfragen einer nur NULL-Werte enthaltende Spalte erzeugen bei Filterung mittels DISTINCT den Wert "0".

Verwendung von DISTINCT in der Zählfunktion

Syntax und Wirkung – COUNT(DISTINCT ausdruck)

- Rückgabe der Zahl der eindeutigen, nicht leeren Werte für "ausdruck"
- Ausdruck – [numerischer Wert, Datumswert, Zeichen(kette)]
- Frage: Wie viele Abteilungen gibt es in der Firma?

```
SELECT COUNT(DISTINCT abt_nr) Abteilungen FROM mitarbeiter;
```

ABTEILUNGEN
35

- Frage: In wieviele Länder liefert das Unternehmen (außer Deutschland)?

```
SELECT COUNT(DISTINCT land) Auslandsmärkte FROM kunde
WHERE land <> 'Deutschland';
```

AUSLANDSMÄRKTE
14

Quelle: Greenberg 2002, S. 241

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

Gruppenfunktion GROUP BY

- Zulässige Datentypen – CHAR, VARCHAR2, NUMBER, DATE
- GROUP BY – Zusammenfassung der Werte innerhalb einer Spalte
- ORDER BY – Sortierung in der Spalte

```
SELECT [spalte,] [DISTINCT|ALL] gruppenfunktion(spalte), ...
FROM tabelle
  [WHERE bedingung]
  [GROUP BY spalte]
```

GROUP BY fasst Datensätze zu Gruppen zusammen.

Quelle: Greenberg 2002, S. 237

Bildung von Zeilengruppen – Dezidierte Selektion

Gruppierung der Ergebnisse mittels GROUP BY

- Selektion der Ergebnisse einer Gruppenfunktion nach verschiedenen Attributen
- Frage: Wie hoch sind die Personalausgaben in den einzelnen Abteilungen?

```
SELECT abt_nr, SUM(gehalt) FROM mitarbeiter
GROUP BY abt_nr;
```

ABT_NR	SUM (GEHALT)
-	235180
260M	11940
310V	16470
210V	20024
210E	17660
…	…

Bei der Gruppierung mittels GROUP BY bilden die NULL-Werte standardmäßig eine eigene Gruppe.

Quelle: Greenberg 2002, S. 245

Gruppierung nach mehreren Spalten

Suche in der Tabelle "MITARBEITER" das Durchschnittsgehalt der einzelnen Berufe innerhalb jeder Abteilung

```
SELECT position, abt_nr, AVG(gehalt) "Mittl. Gehalt" FROM mitarbeiter
GROUP BY abt_nr, position;
```

POSITION	ABT_NR	GEHALT
Vertriebsgruppenleiterin	310V	4490
Vertriebsbeauftragter	110V	1310
Vertriebsbeauftragter	210V	1380
Vertriebsgruppenleiterin	210V	4515
Vertriebsbeauftragter	110V	1310
Geschäftsleiter	10VL	7330
...	...	...
Qualitätsmanager	106Q	4080
...	...	...
Sachbearbeiterin	510L	2700
Sachbearbeiter	510L	2400
Sachbearbeiterin	520G	2500
Sachbearbeiterin	520G	2630
Sachbearbeiter	530A	2400
...	...	...

***Beispiel**:*
Abteilung 520G
2 Sachbearbeiterinnen
Durchschnittsgehalt = 2565 Euro

POSITION	ABT_NR	Mittl. Gehalt
...	...	...
Leiter interne Revision	107R	5180
Sekretärin	330E	2500
Sekretärin	340R	2500
Entwicklungsingenieu	340R	4580
Elektriker	630E	3100
Vertriebsbeauftragter	110V	1352
...	...	...
Sachbearbeiterin	520G	2565
...	...	...

Quelle: Greenberg 2002, S. 245

Kombination von GROUP BY und ORDER BY- Klausel

Gruppierung über mehrere Spalten

- Abfrage des Durchschnitts aller Gehälter innerhalb jeder einzelnen Abteilung bezogen auf die Berufe

```
SELECT abt_nr, position, AVG(gehalt) Durchschnittsgehalt
FROM mitarbeiter
GROUP BY abt_nr, position
ORDER BY abt_nr;
```

ABT_NR	POSITION	DURCHSCHNITTSGEHALT
100V	Assistent der Konzernleitung	3780
100V	Assistentin der Konzernleitung	3400
100V	Investor Relations	3630
100V	Leiterin Vorstandsstab	6800
100V	Public Relations	3610
105C	Abteilungsleiter	8900
105C	Controller	3400
105C	Sekretärin	2450
...	...	...

Verschachteln von Gruppenfunktionen

```
SELECT MAX(AVG(gehalt))"Maximales Durchschnittsgehalt"
FROM mitarbeiter
GROUP BY abt_nr;
```

Maximales Durchschnittsgehalt
19598,33333…..

```
SELECT MAX(AVG(gehalt)) "Maximales Durchschnittsgehalt"
FROM mitarbeiter
WHERE abt_nr IS NOT NULL
GROUP BY abt_nr;
```

Maximales Durchschnittsgehalt
4900

Eine Auswertung verschachtelter Funktionen erfolgt immer von innen nach außen.

Quelle: Greenberg 2002, S. 257

Aggregierung von Daten

Filterung von Gruppenergebnissen

Unterabfragen (Subqueries)

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: Elmasri 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.

Aggregierung von Daten

Filterung von Gruppenergebnissen

Unterabfragen (Subqueries)

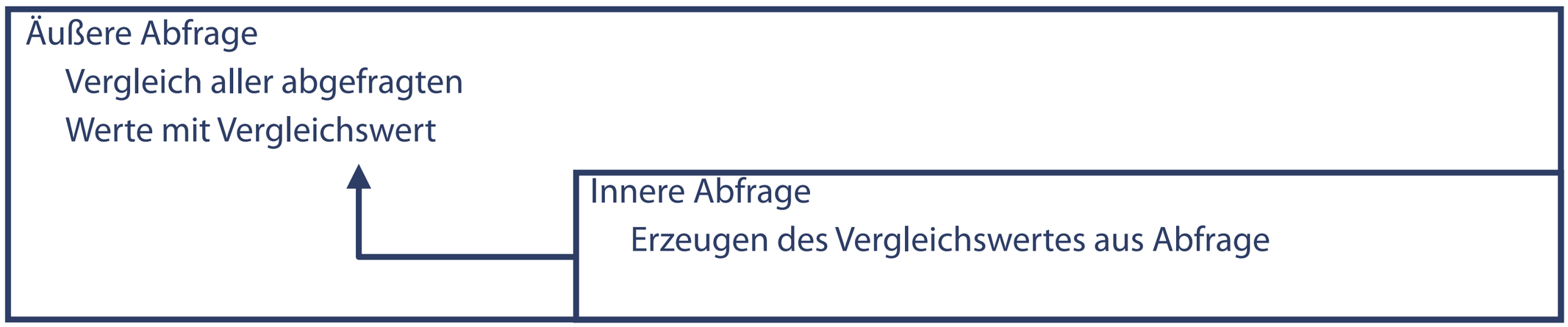
Unterabfragen und Lösungsansätze

Problemstellung

- Vergleiche zwischen zwei Werten einer Spalte

Vorgehensweise

1. Erste (innere) Abfrage – Aufruf (Abfrage) des Vergleichswertes
2. Zweite (äußere) Abfrage – Vergleich der Abfragewerte mit dem aus der inneren Abfrage ermittelten Wert (Vergleichswert)
3. Verbindung der beiden Schritte
   —> Erste Abfrage in die zweite eingebettet



Syntax von Unterabfragen

Problemstellung – Datenauswahl von unbekannten Werten

- Ausführung der Unterabfrage (innere Abfrage), um Wert oder Werteliste zu generieren
- In Hauptabfrage (äußere Abfrage) Erzeugung der eigentlichen Ausgabeliste

```
SELECT spalte
FROM tabelle
WHERE ausdruck vergleichsoperator
                                  (SELECT select_list
                                   FROM tabelle);
```

Quelle: Greenberg 2002, S. 266

Single-Row Unterabfragen

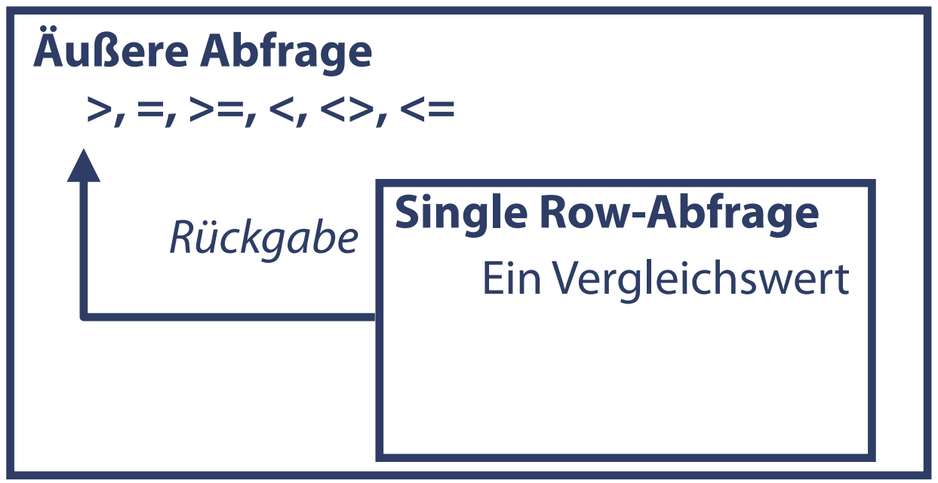
Beispiel – Rückgabe eines Einzelwertes

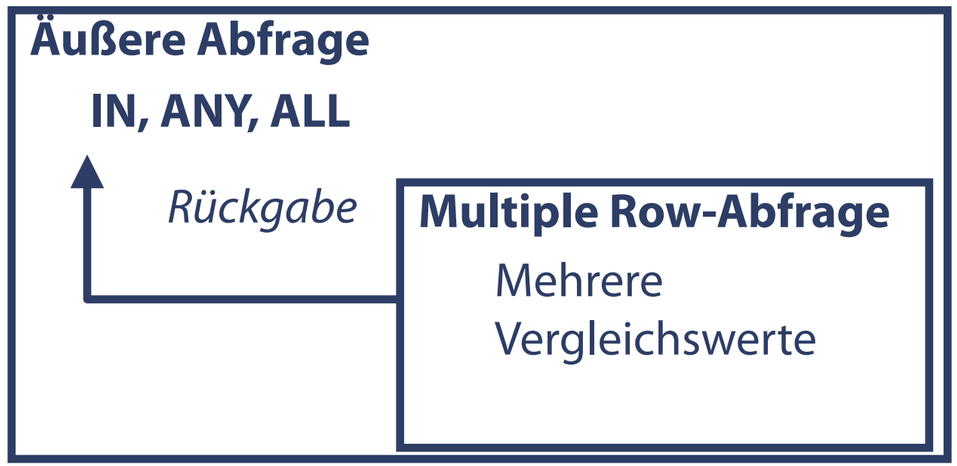
- Wer arbeitet in der Abteilung, zu der auch Karl Plenk gehört?

```
SELECT name, vorname, position
FROM mitarbeiter
WHERE abt_nr =
      (SELECT abt_nr
      FROM mitarbeiter
      WHERE name = 'Plenk'
      AND vorname = 'Karl');
```

NAME	VORNAME	POSITION
Kettler	Gunter	Abteilungsleiter
Klein	Stefan	Konstrukteur
Plenk	Karl	Konstrukteur
Berg	Christin	Sekretärin

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

Single-Row Unterabfragen – Verfeinerung

Beispiel – modifizierte Filterung

- Wer arbeitet in der Abteilung, zu der auch der Mitarbeiter Plenk gehört?
  Die Ergebnisausgabe soll ohne Herrn Plenk erfolgen.

```
SELECT name, vorname, position
FROM mitarbeiter
WHERE abt_nr =          (SELECT abt_nr
                         FROM mitarbeiter
                         WHERE name = 'Plenk'
                         AND vorname = 'Karl')
AND name <> 'Plenk';
```

NAME	VORNAME	POSITION
Kettler	Gunter	Abteilungsleiter
Klein	Stefan	Konstrukteur
Berg	Christin	Sekretärin

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
…	…	…

Leere Werte aus Unterabfragen

Problemstellung – Rückgabe von NULL-Werten

- Leere Ergebnisse aus Unterabfragen ergeben leere Werte der äußeren Abfrage

```
SELECT name, position
FROM mitarbeiter
WHERE position =
    (SELECT position
    FROM mitarbeiter
    WHERE name = 'Penk') ;
```

```
No data found
```

Quelle: Greenberg 2002, S. 279

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.

Operator ALL in Unterabfragen

Innere Abfrage

- Ausgabe 19 Zeilen (Gehälter der 19 Sekretärinnen)

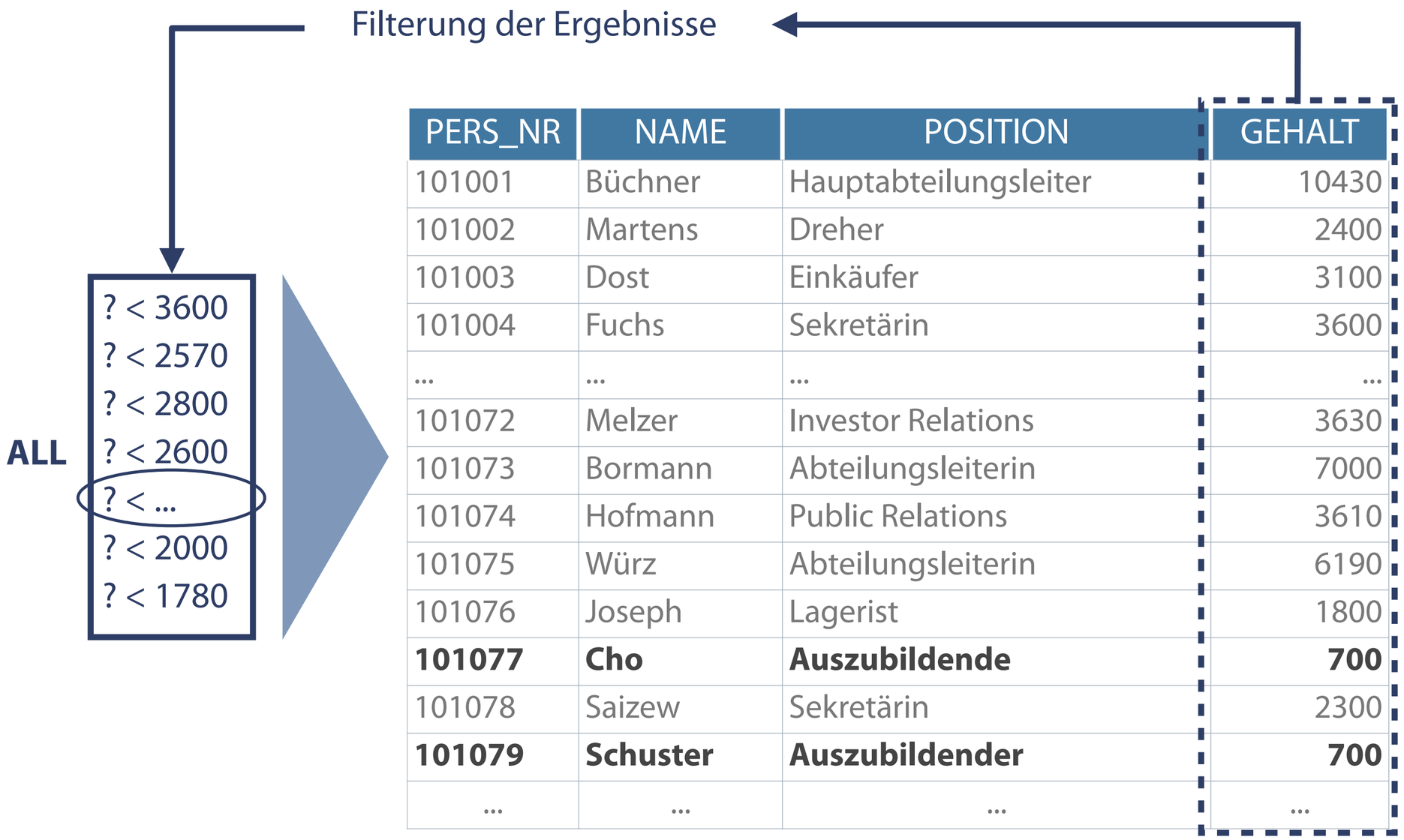
Äußere Abfrage – Übernahme aus Vergleich in der WHERE-Klausel

- <ALL – Kleiner als alle Werte (weniger als minimaler Wert – 1780)
- >ALL – Größer als alle Werte (mehr als maximaler Wert – 3600)
- =ALL – Übereinstimmung zwischen Vergleichswert mit allen Rückgabewerten – praktisch nicht möglich

```
SELECT pers_nr, name, position, gehalt
FROM mitarbeiter
WHERE gehalt < ALL
                 (3600, 2570, 2800, 2600, ..., 1780)
                 (SELECT gehalt FROM mitarbeiter
                 WHERE position = 'Sekretärin')
ORDER BY pers_nr;
```

Quelle: Greenberg 2002, S. 278

Operator ALL in Unterabfragen – Ergebnistabelle



PERS_NR	NAME	POSITION	GEHALT
101001	Büchner	Hauptabteilungsleiter	10430
101002	Martens	Dreher	2400
101003	Dost	Einkäufer	3100
101004	Fuchs	Sekretärin	3600
...	...	...	...
101072	Melzer	Investor Relations	3630
101073	Bormann	Abteilungsleiterin	7000
101074	Hofmann	Public Relations	3610
101075	Würz	Abteilungsleiterin	6190
101076	Joseph	Lagerist	1800
101077	**Cho**	**Auszubildende**	**700**
101078	Saizew	Sekretärin	2300
101079	**Schuster**	**Auszubildender**	**700**
...	...	...	...

Operator ANY in Unterabfragen – Beispiel

Innere Abfrage

- Ausgabe 19 Zeilen (Gehälter der 19 Sekretärinnen)

Äußere Abfrage – Übernahme aus Vergleich in WHERE-Klausel

- < ANY – Kleiner als irgendein Wert (weniger als maximaler Wert = 3600)
- > ANY – Größer als irgendein Wert (mehr als minimaler Wert = 1780)
- = ANY – Entspricht einem der 14 Werte (3600, 2570, 2800, 2600, ..., 1780) – analog der Funktion IN

```
SELECT pers_nr, name, position, gehalt
FROM mitarbeiter
WHERE gehalt < ANY
ORDER BY pers_nr;
```

(3600, 2570, 2800, 2600, ..., 1780)

```
(SELECT gehalt
FROM mitarbeiter
WHERE position = 'Sekretärin')
```

Quelle: Greenberg 2002, S. 277

Operator ANY in Unterabfragen – Ausgangstabellen

Innere Abfrage: Suche nach dem Wert ‘Sekretärin‘

Vergleich der Werte über ANY

Äußere Abfrage: Rückgabewerte aus der inneren Abfrage, um Vergleich zu formulieren

POSITION	GEHALT
Sekretärin	1780
Sekretärin	1950
Sekretärin	2405
Sekretärin	1780
…	…

ANY – nimm irgendeinen Wert aus rechter Seite

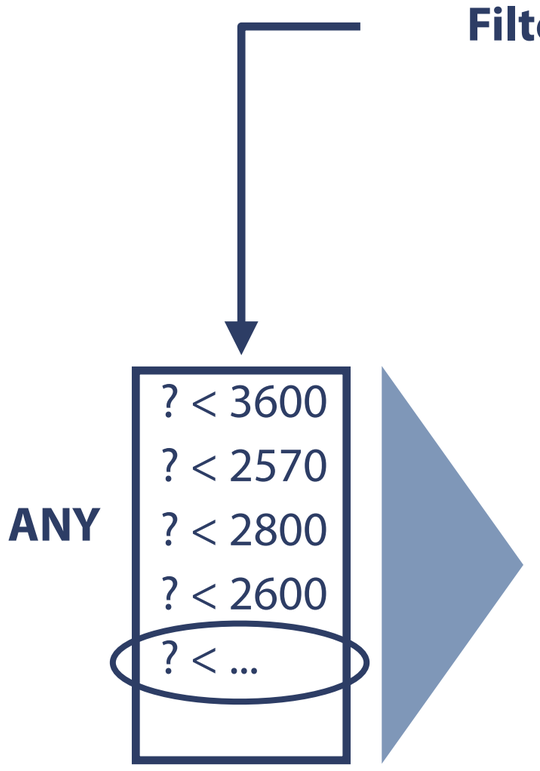
Vergleich Gehalt aus äußerer Abfrage mit jedem Ergebnis-wert aus innerer Abfrage?

$gehalt_{außen} \leqq ANY\ gehalt_{innen}$

PERS_NR	NAME	POSITION	GEHALT	ABT_NR
…	…	…	…	…
101020	Decker	Programmierer	4100	620W
101021	Wagenheinrich	Abteilungsleiter	6500	420F
101022	Brauer	Sekretärin	2570	210E
101023	Krause	Meister	3680	260F
101024	Meyer	Abteilungsleiterin	6510	540P
101025	Petersen	Einkäufer	2890	210E
101026	Schulz-Niemeyer	Abteilungsleiter	8900	105C
101027	Plenk	Konstrukteur	4000	310T
101028	Engel	Buchhalter	4100	420F
101029	Roth	Controller	3400	105C
101030	Riekhoff	Chefsekretärin	3480	
101031	Groß	Sekretärin	2800	230P
101032	Eckert	Wirtschaftsingenieur	4270	230P
…	…	…	…	

WHERE-Klausel – Abfrage Werte Gehalt

Operator ANY in Unterabfragen – Ergebnistabelle



Filterung der Ergebnisse

PERS_NR	NAME	POSITION	GEHALT
…	…	…	…
101020	Decker	Programmierer	4100
101021	Wagenheinrich	Abteilungsleiter	6500
101022	**Brauer**	**Sekretärin**	**2570**
101023	Krause	Meister	3680
101024	Meyer	Abteilungsleiterin	6510
101025	**Petersen**	**Einkäufer**	**2890**
101026	Schulz-Niemeyer	Abteilungsleiter	8900
101027	Plenk	Konstrukteur	4000
101028	Engel	Buchhalter	4100
101029	**Roth**	**Controller**	**3400**
101030	**Riekhoff**	**Chefsekretärin**	**3480**
101031	**Groß**	**Sekretärin**	**2800**
101032	Eckert	Wirtschaftsingenieur	4270
...	...	...	...

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 in einer Unterabfrage – Alternative

Bedingung – Wirkung verschiedener Operatoren

- Bei Erwartung der Rückgabe von NULL-Werten keine Verwendung von NOT IN
- Alternatives Vorgehen: Einsatz des Operators IN
- Wirkweise von IN entspricht "=ANY"

```
SELECT name
FROM mitarbeiter
WHERE pers_nr IN
    (SELECT leiter
    FROM mitarbeiter);
```

NAME
Köhler
Ernst
Klemm
Krajcsir
Michalke
…

NULL-Werte in einer Unterabfrage – Lösung

Bedingung

- Verhinderung der Rückgabe von NULL-Werten für Einsatz des Operators NOT IN

Lösung

- WHERE-Klausel in Unterabfrage
- Wirkweise von IS NOT NULL – Filtern aller nicht leeren Ergebnisse

Beispiel Aufgabenstellung

- Ausgabe aller Positionen (Berufe), die keine Leitungsfunktion besitzen

```
SELECT name, position
FROM mitarbeiter
WHERE pers_nr NOT IN
    (SELECT leiter FROM mitarbeiter
     WHERE leiter IS NOT NULL);
```

Kontrollfragen

- Was sind Aggregatfunktionen?
- Können Gruppenfunktionen auf beliebige Datentypen angewandt werden?
- Wie lässt sich eine Gruppierung nach mehreren Spalten realisieren?
- Welche Aufgabe hat eine Unterabfrage?
- Worin besteht der Unterschied zwischen Single Row- und Multiple Row-Abfragen?

Literatur

Kemper, A./Eickler, A.: Datenbanksysteme; 6. Auflage, 2006, Oldenbourg Verlag

Heuer, A./Saake, G.: Datenbanken, Konzepte und Sprachen; 2. Auflage, 1995, Thomson

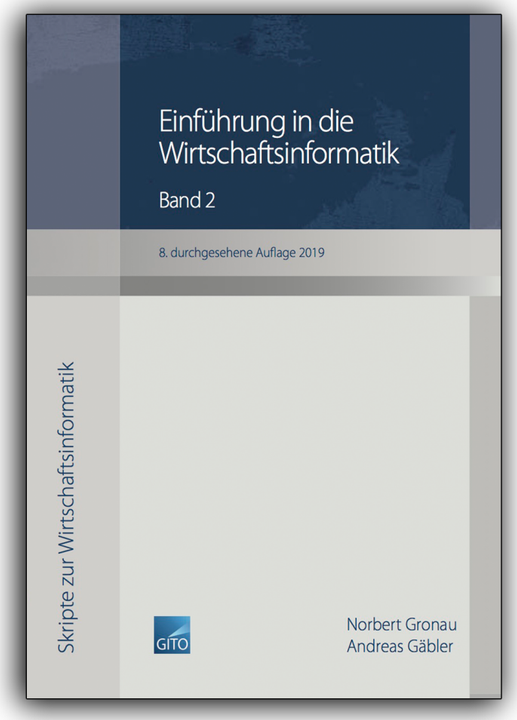
Vossen, G.: Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme; 4. Aufl. - Oldenbourg Verlag München 2000

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

Mertens P. et. al: Grundzüge der Wirtschaftsinformatik; 9. Auflage; 2005, Springer Verlag

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

Zum Nachlesen



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

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

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