IB-DB 4 - 21.3.2015

Dipl.-Ing. Reinhard Schlager

its
FH Salzburg

2015/ IB-Datenbanksysteme

Gliederung

Gliederung

HAVING

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary)> 10000
```

HAVING

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary)> 10000
```

HAVING

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary)> 10000
```

HAVING

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary)> 10000
```

Gliederung

HAVING mit WHERE

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist, ohne Abteilung 100 zu berücksichtigen

```
SELECT department_id, MIN(salary)
FROM employees
WHERE department_id <> 100
GROUP BY department_id
HAVING MIN(salary)> 10000
```

HAVING mit WHERE

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist, ohne Abteilung 100 zu berücksichtigen

```
SELECT department_id, MIN(salary)
FROM employees
WHERE department_id <> 100
GROUP BY department_id
HAVING MIN(salary)> 10000
```

HAVING mit WHERE

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist, ohne Abteilung 100 zu berücksichtigen

```
SELECT department_id, MIN(salary)
FROM employees
WHERE department_id <> 100
GROUP BY department_id
HAVING MIN(salary)> 10000
```

HAVING mit WHERE

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist, ohne Abteilung 100 zu berücksichtigen

```
SELECT department_id, MIN(salary)
FROM employees
WHERE department_id <> 100
GROUP BY department_id
HAVING MIN(salary)> 10000
```

HAVING mit WHERE

Die Abteilung, in denen das kleinste Gehalt größer 10000 ist, ohne Abteilung 100 zu berücksichtigen

```
SELECT department_id, MIN(salary)
FROM employees
WHERE department_id <> 100
GROUP BY department_id
HAVING MIN(salary)> 10000
```

Gliederung

SELF JOIN

Beispiel:Vorgesetzer von

Gesucht ist eine Liste Mitarbeiter,Vorgesetzter ...

```
SELECT e.last_name AS Mitarbeiter,
v.last_name AS Vorgesetzter
FROM
employees e,
employees v
WHERE e.manager_id = v.employee_id
```

SELF JOIN

Beispiel:Vorgesetzer von

Gesucht ist eine Liste Mitarbeiter,Vorgesetzter ...

```
SELECT e.last_name AS Mitarbeiter,
v.last_name AS Vorgesetzter
FROM
employees e,
employees v
WHERE e.manager_id = v.employee_id
```

SELF JOIN

Beispiel:Vorgesetzer von

Gesucht ist eine Liste Mitarbeiter,Vorgesetzter ...

```
SELECT e.last_name AS Mitarbeiter,
v.last_name AS Vorgesetzter
FROM
employees e,
employees v
WHERE e.manager_id = v.employee_id
```

SELF JOIN

Beispiel:Vorgesetzer von

Gesucht ist eine Liste Mitarbeiter,Vorgesetzter ...

```
SELECT e.last_name AS Mitarbeiter,
v.last_name AS Vorgesetzter
FROM
employees e,
employees v
WHERE e.manager_id = v.employee_id
```

SELF JOIN

Beispiel:Vorgesetzer von

Gesucht ist eine Liste Mitarbeiter,Vorgesetzter ...

```
SELECT e.last_name AS Mitarbeiter,
v.last_name AS Vorgesetzter
FROM
employees e,
employees v
WHERE e.manager_id = v.employee_id
```

SELF JOIN

Beispiel:Vorgesetzer von

Gesucht ist eine Liste Mitarbeiter,Vorgesetzter ...

```
SELECT e.last_name AS Mitarbeiter,
v.last_name AS Vorgesetzter
FROM
employees e,
employees v
WHERE e.manager_id = v.employee_id
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees

SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees

SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees
```

```
SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees
```

```
SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees
```

```
SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees

SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees

SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

MINUS

Von der ersten Menge wird die zweite Menge abgezogen.

```
SELECT department_id FROM departments
MINUS
SELECT department_id FROM employees

SELECT d.department_id FROM departments d
WHERE d.department_id  NOT IN(
   SELECT  department_id  FROM employees
   WHERE  department_id IS NOT NULL )
```

INTERSECT

INTERSECT ergibt die Überschneidungsmenge von Elementen, die zu beiden Mengen gehören.

```
SELECT field1, field2, ... field_n FROM table
INTERSECT
SELECT field1, field2, ... field_n FROM table
```

INTERSECT

INTERSECT ergibt die Überschneidungsmenge von Elementen, die zu beiden Mengen gehören.

```
SELECT field1, field2, ... field_n FROM table
INTERSECT
SELECT field1, field2, ... field_n FROM table
```

INTERSECT

INTERSECT ergibt die Überschneidungsmenge von Elementen, die zu beiden Mengen gehören.

```
SELECT field1, field2, ... field_n FROM table
INTERSECT
SELECT field1, field2, ... field_n FROM table
```

Gliederung

NULL

Besonderheiten bei NULL und AVG

```
SELECT AVG(salary) FROM employees;
6434,92

SELECT AVG(NVL(salary,0)) FROM employees;
6376,42
```

NULL

Besonderheiten bei NULL und AVG

```
SELECT AVG(salary) FROM employees;
6434,92

SELECT AVG(NVL(salary,0)) FROM employees;
6376,42
```

NULL

Besonderheiten bei NULL und AVG

```
SELECT AVG(salary) FROM employees;
6434,92

SELECT AVG(NVL(salary,0)) FROM employees;
6376,42
```

NULL

Besonderheiten bei NULL und AVG

```
SELECT AVG(salary) FROM employees;
6434,92

SELECT AVG(NVL(salary,0)) FROM employees;
6376,42
```

Gliederung

Inline View

Inline Views sind Unterabfragen nach FROM.

```
SELECT a.last_name, a.salary, a.department_id,
FROM employees a,
(SELECT e.department_id,max(e.salary) AS maxsal
 FROM   employees e
 GROUP  BY e.department_id) b
WHERE a.department_id = b.department_id
AND a.salary < b.maxsal
```

Inline View

Inline Views sind Unterabfragen nach FROM.

```
SELECT a.last_name, a.salary, a.department_id,
FROM employees a,
(SELECT e.department_id,max(e.salary) AS maxsal
 FROM   employees e
 GROUP  BY e.department_id) b
WHERE a.department_id = b.department_id
AND a.salary < b.maxsal
```

Inline View

Inline Views sind Unterabfragen nach FROM.

```
SELECT a.last_name, a.salary, a.department_id,
FROM employees a,
(SELECT e.department_id,max(e.salary) AS maxsal
 FROM   employees e
 GROUP  BY e.department_id) b
WHERE a.department_id = b.department_id
AND a.salary < b.maxsal
```

Inline View

Inline Views sind Unterabfragen nach FROM.

```
SELECT a.last_name, a.salary, a.department_id,
FROM employees a,
(SELECT e.department_id,max(e.salary) AS maxsal
 FROM   employees e
 GROUP  BY e.department_id) b
WHERE a.department_id = b.department_id
AND a.salary < b.maxsal
```

Inline View

Inline Views sind Unterabfragen nach FROM.

```
SELECT a.last_name, a.salary, a.department_id,
FROM employees a,
(SELECT e.department_id,max(e.salary) AS maxsal
 FROM   employees e
 GROUP  BY e.department_id) b
WHERE a.department_id = b.department_id
AND a.salary < b.maxsal
```

Gliederung

Top n Rows

Welche drei Mitarbeiter verdienen am besten?

```
SELECT ROWNUM AS rank, last_name, salary
FROM (
SELECT last_name,salary
FROM employees
WHERE salary IS NOT NULL
ORDER BY salary DESC)
WHERE ROWNUM <= 3
```

Top n Rows

Welche drei Mitarbeiter verdienen am besten?

```
SELECT ROWNUM AS rank, last_name, salary
FROM (
SELECT last_name,salary
FROM employees
WHERE salary IS NOT NULL
ORDER BY salary DESC)
WHERE ROWNUM <= 3
```

Top n Rows

Welche drei Mitarbeiter verdienen am besten?

```
SELECT ROWNUM AS rank, last_name, salary
FROM (
SELECT last_name,salary
FROM employees
WHERE salary IS NOT NULL
ORDER BY salary DESC)
WHERE ROWNUM <= 3
```

Top n Rows

Welche drei Mitarbeiter verdienen am besten?

```
SELECT ROWNUM AS rank, last_name, salary
FROM (
SELECT last_name,salary
FROM employees
WHERE salary IS NOT NULL
ORDER BY salary DESC)
WHERE ROWNUM <= 3
```

Top n Rows

Welche drei Mitarbeiter verdienen am besten?

```
SELECT ROWNUM AS rank, last_name, salary
FROM (
SELECT last_name,salary
FROM employees
WHERE salary IS NOT NULL
ORDER BY salary DESC)
WHERE ROWNUM <= 3
```

Top n Rows

Welche drei Mitarbeiter verdienen am besten?

```
SELECT ROWNUM AS rank, last_name, salary
FROM (
SELECT last_name,salary
FROM employees
WHERE salary IS NOT NULL
ORDER BY salary DESC)
WHERE ROWNUM <= 3
```

Top n Rows

Welche drei Mitarbeiter verdienen am besten?

```
SELECT ROWNUM AS rank, last_name, salary
FROM (
SELECT last_name,salary
FROM employees
WHERE salary IS NOT NULL
ORDER BY salary DESC)
WHERE ROWNUM <= 3
```

Gliederung

LIKE

LIKE - Vergleich mit Platzhaltern

```
SELECT * FROM employees

... WHERE last_name LIKE 'F%'
{Fager, Feger, Federer, Fissl, Furtal}
... WHERE last_name LIKE 'F_ger'
{Fager, Feger}
... WHERE last_name LIKE '%l'
{Fissl, Furtal}
```

LIKE
LIKE - Vergleich mit Platzhaltern

```
SELECT * FROM employees

... WHERE last_name LIKE 'F%'
{Fager, Feger, Federer, Fissl, Furtal}
... WHERE last_name LIKE 'F_ger'
{Fager, Feger}
... WHERE last_name LIKE '%l'
{Fissl, Furtal}
```

LIKE

LIKE - Vergleich mit Platzhaltern

```
SELECT * FROM employees

... WHERE last_name LIKE 'F%'
{Fager, Feger, Federer, Fissl, Furtal}
... WHERE last_name LIKE 'F_ger'
{Fager, Feger}
... WHERE last_name LIKE '%l'
{Fissl, Furtal}
```

LIKE

LIKE - Vergleich mit Platzhaltern

```
SELECT * FROM employees

... WHERE last_name LIKE 'F%'
{Fager, Feger, Federer, Fissl, Furtal}
... WHERE last_name LIKE 'F_ger'
{Fager, Feger}
... WHERE last_name LIKE '%l'
{Fissl, Furtal}
```

LIKE

LIKE - Vergleich mit Platzhaltern

```
SELECT * FROM employees

... WHERE last_name LIKE 'F%'
{Fager, Feger, Federer, Fissl, Furtal}
... WHERE last_name LIKE 'F_ger'
{Fager, Feger}
... WHERE last_name LIKE '%l'
{Fissl, Furtal}
```

LIKE

LIKE - Vergleich mit Platzhaltern

```
SELECT * FROM employees

... WHERE last_name LIKE 'F%'
{Fager, Feger, Federer, Fissl, Furtal}
... WHERE last_name LIKE 'F_ger'
{Fager, Feger}
... WHERE last_name LIKE '%l'
{Fissl, Furtal}
```

LIKE

LIKE - Vergleich mit Platzhaltern

```
SELECT * FROM employees

... WHERE last_name LIKE 'F%'
{Fager, Feger, Federer, Fissl, Furtal}
... WHERE last_name LIKE 'F_ger'
{Fager, Feger}
... WHERE last_name LIKE '%l'
{Fissl, Furtal}
```

LIKE
ESCAPE

Durch ESCAPE kann man nach _ und % suchen und vergleichen.

```
SELECT * FROM employees
WHERE last_name LIKE '%/_%' ESCAPE '/'
```

. . . sucht nach Nachnamen, die ein _ enthalten.

LIKE
ESCAPE

Durch ESCAPE kann man nach _ und % suchen und vergleichen.

```
SELECT * FROM employees
WHERE last_name LIKE '%/_%' ESCAPE '/'
```

. . . sucht nach Nachnamen, die ein _ enthalten.

Übung 4 IB-DB 21.3.2015
HR schema

- Erzeugen Sie eine Liste aller Mitarbeiter (Vorname,Nachname), die in einer Abteilung arbeiten, in der alle Mitarbeiter dieser Abteilung zusammen über 150000 verdienen
- Ergänzen sie diese Liste um den Namen der Abteilung (Vorname,Nachname, Abteilungsname)
- Finden Sie die drei Mitarbeiter, die am schlechtesten verdienen (Vorname,Nachname,salary)
- Schreiben Sie *EIN* SQL Statement, dass diese drei Gehälter um 1000 erhöht

Übung 4 IB-DB 21.3.2015
HR schema

- Erzeugen Sie eine Liste aller Mitarbeiter (Vorname,Nachname), die in einer Abteilung arbeiten, in der alle Mitarbeiter dieser Abteilung zusammen über 150000 verdienen
- Ergänzen sie diese Liste um den Namen der Abteilung (Vorname,Nachname, Abteilungsname)
- Finden Sie die drei Mitarbeiter, die am schlechtesten verdienen (Vorname,Nachname,salary)
- Schreiben Sie *EIN* SQL Statement, dass diese drei Gehälter um 1000 erhöht

Übung 4 IB-DB 21.3.2015

HR schema

- Erzeugen Sie eine Liste aller Mitarbeiter (Vorname,Nachname), die in einer Abteilung arbeiten, in der alle Mitarbeiter dieser Abteilung zusammen über 150000 verdienen
- Ergänzen sie diese Liste um den Namen der Abteilung (Vorname,Nachname, Abteilungsname)
- Finden Sie die drei Mitarbeiter, die am schlechtesten verdienen (Vorname,Nachname,salary)
- Schreiben Sie *EIN* SQL Statement, dass diese drei Gehälter um 1000 erhöht

Übung 4 IB-DB 21.3.2015

HR schema

- Erzeugen Sie eine Liste aller Mitarbeiter (Vorname,Nachname), die in einer Abteilung arbeiten, in der alle Mitarbeiter dieser Abteilung zusammen über 150000 verdienen
- Ergänzen sie diese Liste um den Namen der Abteilung (Vorname,Nachname, Abteilungsname)
- Finden Sie die drei Mitarbeiter, die am schlechtesten verdienen (Vorname,Nachname,salary)
- Schreiben Sie *EIN* SQL Statement, dass diese drei Gehälter um 1000 erhöht

Übung 4 IB-DB 21.3.2015(2)
HR schema

- Listen Sie die Angestelltennummern und Job-Kennungen der Angestellten auf, deren Berufsbezeichnung momentan gleich lautet wie zu Beginn ihrer Tätigkeit im Unternehmen, d.h. der Job hat sich nicht verändert (INTERSECT, Lösung 2 Zeilen).
- Erzeugen Sie eine Liste Vorname, NachName, salary, Gehaltsdifferenz zum eigenen Manager

Übung 4 IB-DB 21.3.2015(2)

HR schema

- Listen Sie die Angestelltennummern und Job-Kennungen der Angestellten auf, deren Berufsbezeichnung momentan gleich lautet wie zu Beginn ihrer Tätigkeit im Unternehmen, d.h. der Job hat sich nicht verändert (INTERSECT, Lösung 2 Zeilen).
- Erzeugen Sie eine Liste Vorname, NachName, salary, Gehaltsdifferenz zum eigenen Manager