DOAG 2014

Analytische Funktionen in SQL für Einsteiger

Ewald GmbH
Jürgen Habdank

juergen.habdank@ewald.de

Ewald GmbH

Über 20 Jahre IT-Dienstleister

Von der Beratung bis zur individuelle Systemlösung

Von der Analyse bis zu Schulung und Support

Schwerpunktmäßig im Finanzbereich

Datenbank: Oracle, Gupta/Unify

Datenmodell: PowerDesigner

Entwicklung: Oracle, Microsoft, Gupta/Unify, SAP

Referent

Über 30 Jahre IT-Erfahrung

Seit etwa 20 Jahren in verschiedenen IT-Projekten

Analyse, Konzept, Realisierung, Test, Dokumentation

Verschiedenste Sprachen und Entwicklungswerkzeuge

Viele Anforderungen auch in SQL bzw. PL/SQL

Datenbank schwerpunktmäßig mit Oracle

Motivation

Überblick über das Thema „Analytische Funktionen“

Einfache Syntax

Ansonsten prozeduraler/aufwändiger SQL-Code

Sehr performant

Konkrete Beispiele

Erster Einblick in die Möglichkeiten

Selbst ausprobieren

Syntax und Semantik

analytic_function::=

- Berechnung für jede Zeile der Eingabemenge
- Verwendung im „Select“ (+ Sub-Query) und „Order By“
- Keine Verwendung im „Where“
- Keine Schachtelung
- Auch Verwendung von Benutzerfunktionen

Syntax und Semantik

analytic_clause::=

- Aggregats-Funktionen: Gruppierung mit „Group By“
- Hier „query_partition_clause“ und „windowing_clause“
- Sortierung der Partitionen durch „order_by_clause“
- Sortierung nur auf die Partitionen der Eingabemenge
- Keine Sortierung des gesamten Result-Sets.

Syntax und Semantik

analytic_clause::=

- Komponenten definieren Fenster („window“), auf dem die Funktion operiert
- Fenster bewegt sich über die Eingabemenge
- Berechnung des Fensters abhängig von aktueller Zeile
- Logische Intervalle oder physikalische Rows

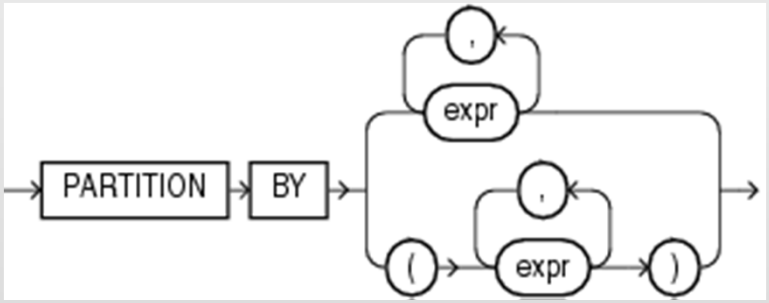
Syntax und Semantik

analytic_clause::=

- „windowing-clause“ nur mit „order_by_clause“
- Ausführung nach „From“, „Where“, „Group By“ und „Having“
- Ausführung vor „Order By“

Syntax und Semantik

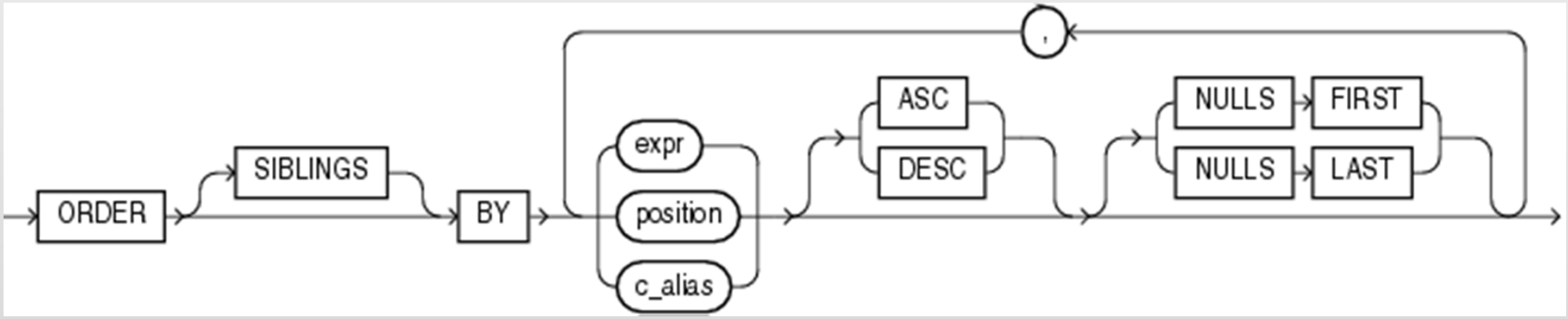
query_partition_clause::=



- Hier nur oberer Zweig des Syntaxdiagramms
- „query_partition_clause“ unterteilt die Eingabemenge
- Ohne „query_partition_clause“ eine Partition
- Ergebnis der analytischen Funktion je Partition
- Für jede neue Partition wird Berechnung zurückgesetzt

Syntax und Semantik

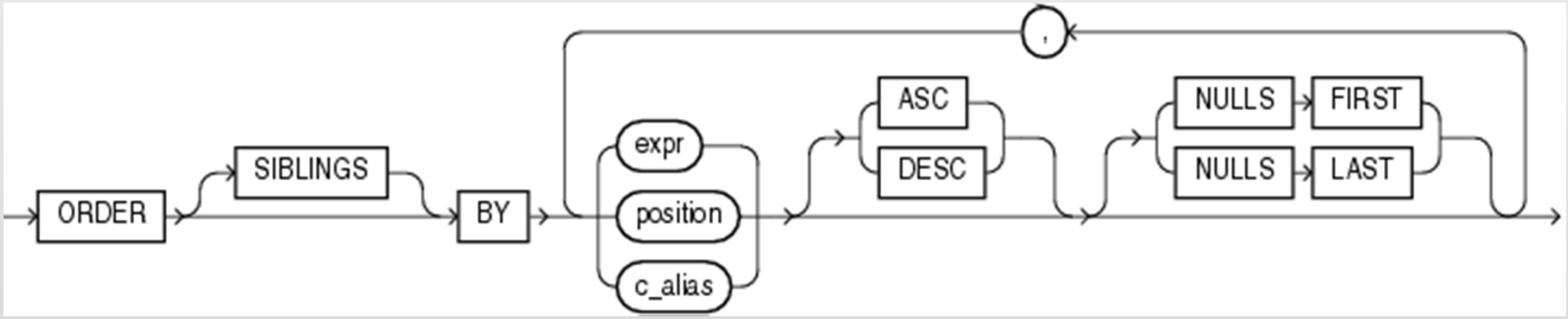
order_by_clause::=



- „SIBLINGS“ nicht für analytische Funktionen
- Für analytische Funktionen nur Ausdrücke
- „order_by_clause“ sortiert die Menge, die durch „query_partition_clause“ festgelegt wird
- Liefert sortierte Menge für die analytische Funktion
- Keine Sortierung für das gesamte Result-Set

Syntax und Semantik

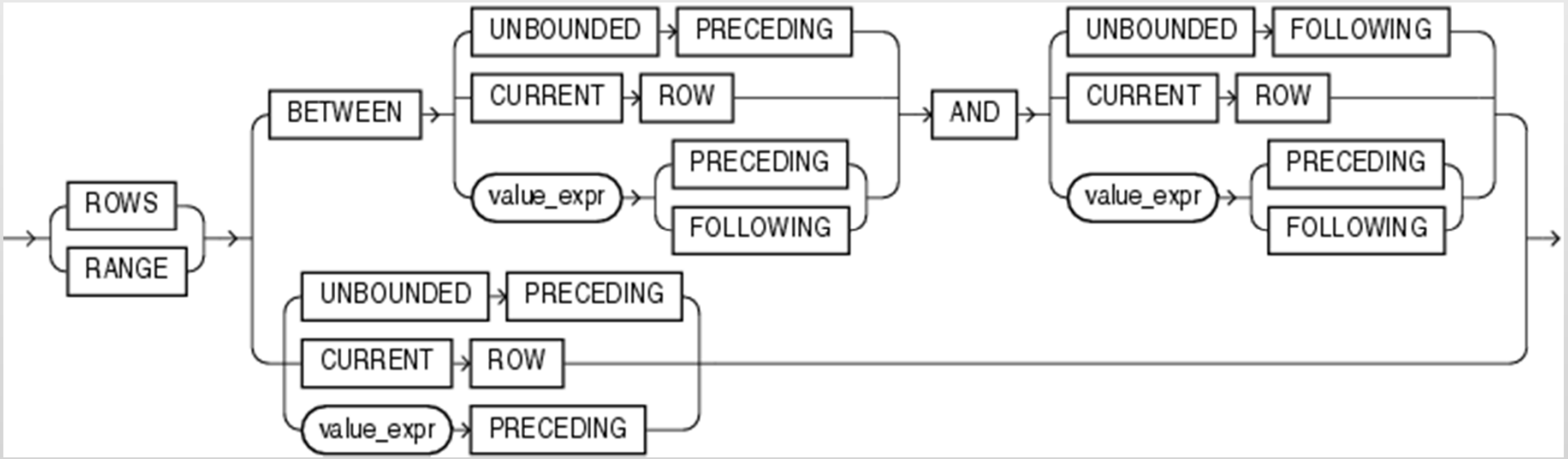
order_by_clause::=



- Für logischen Bereich mehrere Sortierkriterien nur für „RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW“ und „RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING“ erlaubt
- Für physikalischen Bereich mehrere Sortierkriterien
- „Null“-Werte haben den höchsten Wert
- „Null“-Werte am besten ausschließen

Syntax und Semantik

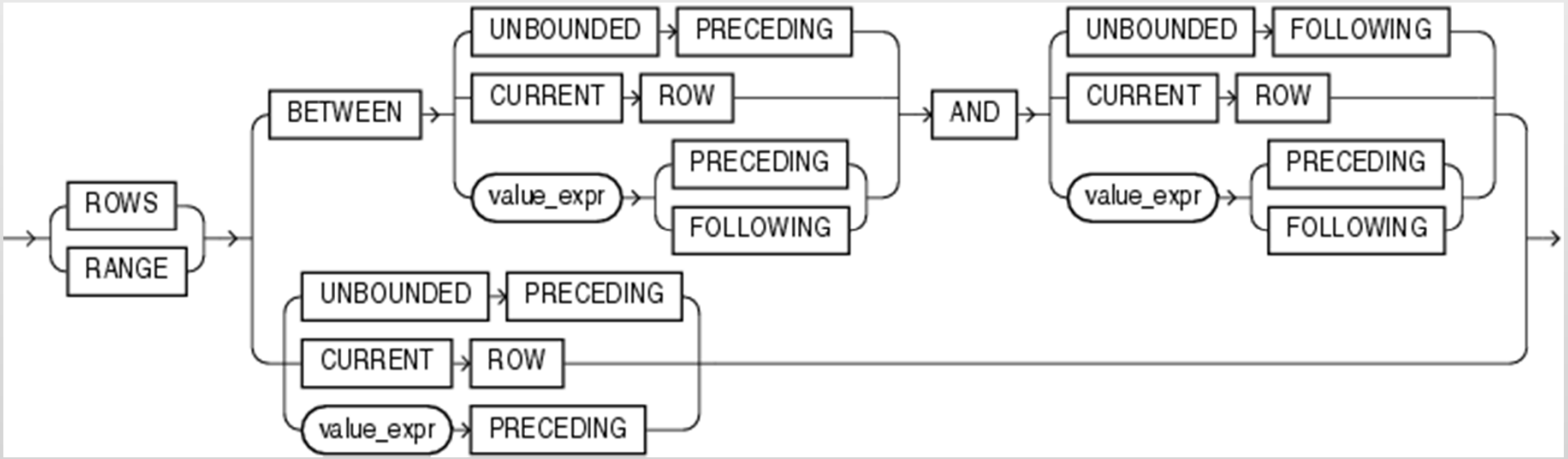
windowing_clause::=



- „windowing_clause“: Anfang und Ende des Fensters
- Bezogen auf aktuelle Row des gesamten Result-Sets
- Berechnung der analytische Funktion basiert auf Fenster
- „CURRENT ROW“: Logischer Wert bzw. physikalische Row
- „CURRENT ROW“: Row für Berechnung des Ergebnisses

Syntax und Semantik

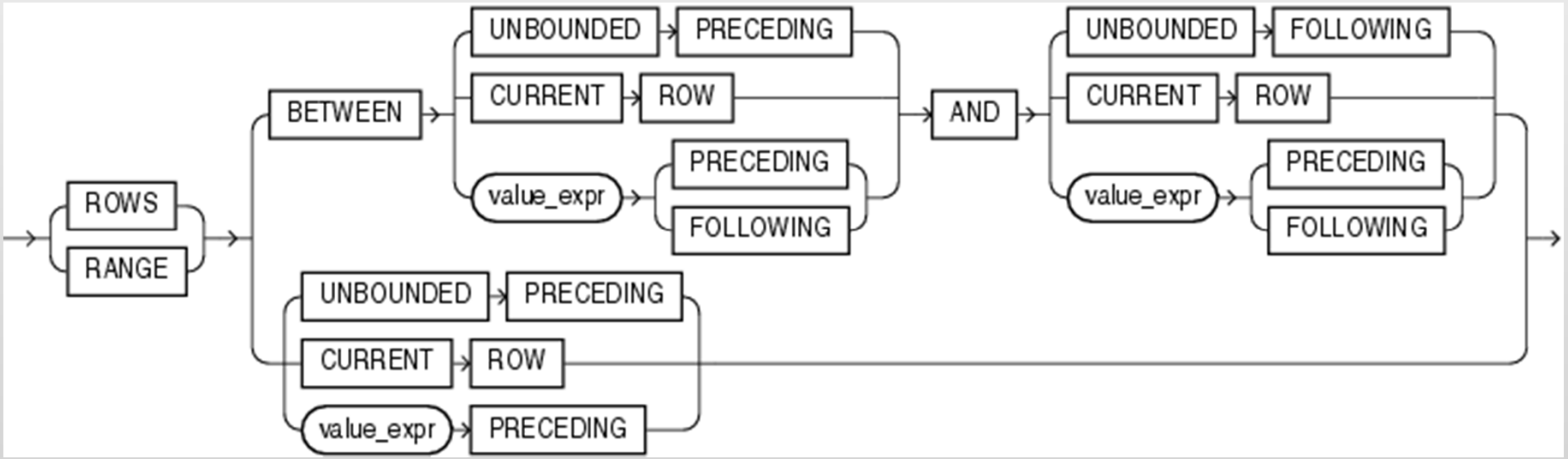
windowing_clause::=



- „UNBOUNDED PRECEDING“: Erste Zeile der Partition
- „UNBOUNDED FOLLOWING“: Letzte Zeile der Partition
- Ohne Start-/Endpunkt (unterer Zweig): „CURRENT ROW“
- Standard ist „RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW“ und kann entfallen

Syntax und Semantik

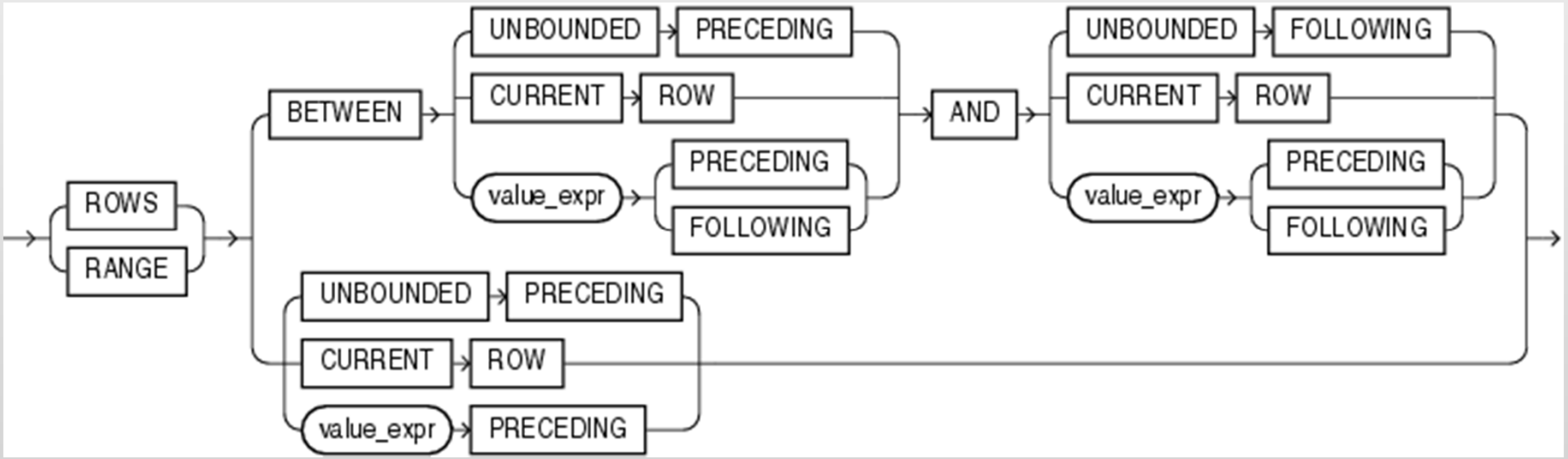
windowing_clause::=



- „windowing-clause“ nur wenn „order_by_clause“
- Ohne „order_by_clause“ kein Standard für „windowing_clause“
- „ROWS“: Zeilen auf Basis der physikalischen Rows
- „ROWS“: Bezug ist Position der Zeile

Syntax und Semantik

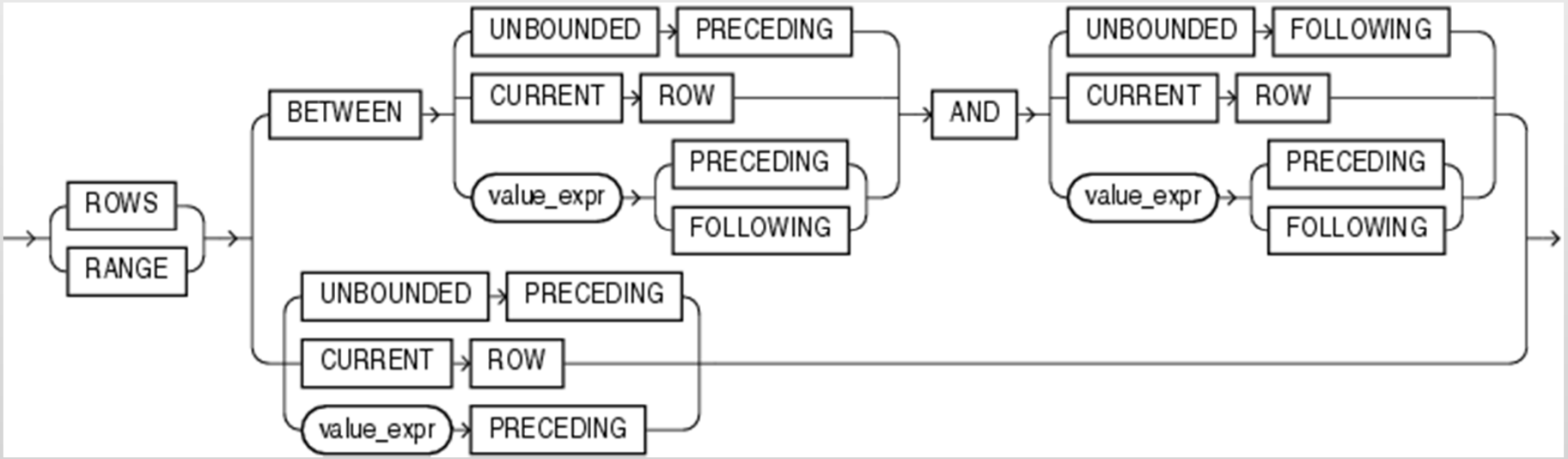
windowing_clause::=



- „RANGE“: Zeilen auf Basis einer “where_clause”
- „RANGE“: Bezug ist Wert der Zeile
- „RANGE“: Intervalle sind numerisch oder „Date“
- „RANGE“: „CURRENT ROW“ ist der aktuelle Wert der Zeile nicht die physikalische Row

Syntax und Semantik

windowing_clause::=



- „<value_expr> PRECEDING“ bzw. „<value_expr> FOLLOWING“ definiert logisches/physikalisches Intervall
- „windowing_clause“ nicht bei allen analytischen Funktionen erlaubt

Überblick Funktionen

AVG	
CORR	
COUNT	
COVAR_POP	
COVAR_SAMP	
CUME_DIST	(keine „windowing_clause“)
DENSE_RANK	(keine „windowing_clause“)
FIRST_VALUE	
LAG	(keine „windowing_clause“)
LAST_VALUE	
LEAD	(keine „windowing_clause“)
MAX	
MIN	

Überblick Funktionen

NTILE	(keine „windowing_clause“)
PERCENT_RANK	(keine „windowing_clause“)
RANK	(keine „windowing_clause“)
RATIO_TO_REPORT	(keine „windowing_clause“)
REGR_ …Functions	
ROW_NUMBER	(keine „windowing_clause“)
STDDEV	
STDDEV_POP	
STDDEV_SAMPS	
SUM	
VAR_POP	
VAR_SAMP	
VARIANCE	

Beispiele

ORACLE 11 XE

SQL Developer 4.0.2

HR-Schema

Evtl. Schema „HR“ aus anderem Schema entsperren:

```
Alter User HR Account Unlock;
```

01 Laufende Summe/Angestellte

```
Select
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>
From
  EMPLOYEES
Order By
  EMPLOYEES.SALARY,
  EMPLOYEES.LAST_NAME;
```

01 Laufende Summe/Angestellte

<Analytische Funktion 1>::=

```
Sum(EMPLOYEES.SALARY)
  Over(Order By
          EMPLOYEES.SALARY,
          EMPLOYEES.LAST_NAME
        Range Between Unbounded Preceding And
        Current Row
       ) CurrSumEmp
```

01 Laufende Summe/Angestellte

Olson	2100	2100
Markle	2200	4300
Philtanker	2200	6500
Gee	2400	8900
Landry	2400	11300
Colmenares	2500	13800
Marlow	2500	16300
Patel	2500	18800
Perkins	2500	21300
Sullivan	2500	23800
Vargas	2500	26300
Grant	2600	28900
Himuro	2600	31500

02 Lauf. Summe/Angestellte/Abt.

```
Select
  DEPARTMENTS.DEPARTMENT_NAME,
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>
From
  EMPLOYEES
  Inner Join DEPARTMENTS On
    EMPLOYEES.DEPARTMENT_ID =
    DEPARTMENTS.DEPARTMENT_ID
Order By
  DEPARTMENTS.DEPARTMENT_NAME,
  EMPLOYEES.LAST_NAME;
```

02 Lauf. Summe/Angestellte/Abt.

<Analytische Funktion 1>::=

```
Sum(EMPLOYEES.SALARY)
  Over(Partition By
         DEPARTMENTS.DEPARTMENT_NAME
       Order By
         EMPLOYEES.LAST_NAME
       Range Between Unbounded Preceding And
       Current Row
      ) CurrSumEmpPerDep
```

02 Lauf. Summe/Angestellte/Abt.

Accounting	Gietz	8300	8300
Accounting	Higgins	12008	20308
Administration	Whalen	4400	4400
Executive	De Haan	17000	17000
Executive	King	24000	41000
Executive	Kochhar	17000	58000
Finance	Chen	8200	8200
Finance	Faviet	9000	17200
Finance	Greenberg	12008	29208
Finance	Popp	6900	36108
Finance	Sciarra	7700	43808
Finance	Urman	7800	51608
Human Resources	Mavris	6500	6500

03 Mehrere Summen

```
Select
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>,
  <Analytische Funktion 3>,
  <Analytische Funktion 4>
From
  EMPLOYEES
Order By
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.SALARY;
```

03 Mehrere Summen

<Analytische Funktion 1>::=

```
Sum(EMPLOYEES.SALARY)
  Over(Partition By
         EMPLOYEES.DEPARTMENT_ID
       Order By
         EMPLOYEES.SALARY
      ) CurrSumEmpPerDep
```

03 Mehrere Summen

<Analytische Funktion 2>::=

```
Sum(EMPLOYEES.SALARY)
  Over(Partition By
          EMPLOYEES.DEPARTMENT_ID
       ) SumEmpPerDep
-- Kein Standard für -'windowing-clause', da keine
   'order_by_clause'
```

03 Mehrere Summen

<Analytische Funktion 3>::=

```
Sum(EMPLOYEES.SALARY)
  Over(Order By
          EMPLOYEES.DEPARTMENT_ID,
          EMPLOYEES.SALARY
        ) CurrSumEmpAllDep
```

03 Mehrere Summen

<Analytische Funktion 4>::=

```
Sum(EMPLOYEES.SALARY)
  Over(
        ) SumEmpAllDep
-- Kein Standard für -'windowing-clause', da keine
   'order_by_clause'
```

03 Mehrere Summen

10	Whalen	4400	4400	4400	4400	691416
20	Fay	6000	6000	19000	10400	691416
20	Hartstein	13000	19000	19000	23400	691416
30	Colmenares	2500	2500	24900	25900	691416
30	Himuro	2600	5100	24900	28500	691416
30	Tobias	2800	7900	24900	31300	691416
30	Baida	2900	10800	24900	34200	691416
30	Khoo	3100	13900	24900	37300	691416
30	Raphaely	11000	24900	24900	48300	691416
40	Mavris	6500	6500	6500	54800	691416
50	Olson	2100	2100	156400	56900	691416
50	Philtanker	2200	6500	156400	61300	691416
50	Markle	2200	6500	156400	61300	691416

04 Nachfolger/Vorgänger

```
Select
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>,
  <Analytische Funktion 3>
From
  EMPLOYEES
Order By
  EMPLOYEES.SALARY;
```

04 Nachfolger/Vorgänger

<Analytische Funktion 1>::=

```
 Lead(EMPLOYEES.LAST_NAME,
          1,
          'nicht vorhanden'
         )
   Over(Order By
              EMPLOYEES.SALARY
          ) NextEmp,
 -- Keine 'windowing_clause' für 'Lead' erlaubt =>
    Komplette Menge
```

04 Nachfolger/Vorgänger

```
<Analytische Funktion 2>::=
  Lead(EMPLOYEES.LAST_NAME,
           2,
           Null
          )
    Over(Order By
              EMPLOYEES.SALARY
           ) AfterNextSal,
  -- Keine 'windowing_clause' für 'Lead' erlaubt =>
     Komplette Menge
```

04 Nachfolger/Vorgänger

```
<Analytische Funktion 3>::=
  Lag(EMPLOYEES.LAST_NAME,
         1,
         'nicht vorhanden'
       )
    Over(Order By
              EMPLOYEES.SALARY
           ) PrevEmp,
  -- Keine 'windowing_clause' für 'Lag' erlaubt =>
     Komplette Menge
```

04 Nachfolger/Vorgänger

Olson	2100	Markle	2200	nicht vorhanden
Markle	2200	Philtanker	2400	Olson
Philtanker	2200	Landry	2400	Markle
Landry	2400	Gee	2500	Philtanker
Gee	2400	Colmenares	2500	Landry
Colmenares	2500	Marlow	2500	Gee
Marlow	2500	Patel	2500	Colmenares
Patel	2500	Vargas	2500	Marlow
Vargas	2500	Sullivan	2500	Patel
Sullivan	2500	Perkins	2600	Vargas
Perkins	2500	Himuro	2600	Sullivan
Himuro	2600	Matos	2600	Perkins
Matos	2600	OConnell	2600	Himuro

05 Nachfolger/Vorgänger

```
Select
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>
From
  EMPLOYEES
Order By
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.SALARY;
```

05 Nachfolger/Vorgänger

```
<Analytische Funktion 1>::=
  Lead(EMPLOYEES.LAST_NAME,
           1,
           'nicht vorhanden'
          )
    Over(Partition By
              EMPLOYEES.DEPARTMENT_ID
            Order By
              EMPLOYEES.SALARY
           ) NextEmp,
  -- Keine 'windowing_clause' für 'Lead' erlaubt =>
     Komplette Menge
```

05 Nachfolger/Vorgänger

```
<Analytische Funktion 2>::=
  Lag(EMPLOYEES.LAST_NAME,
        1,
        'nicht vorhanden'
      )
    Over(Partition By
           EMPLOYEES.DEPARTMENT_ID
         Order By
           EMPLOYEES.SALARY
        ) PrevEmp,
  -- Keine 'windowing_clause' für 'Lag' erlaubt =>
     Komplette Menge
```

05 Nachfolger/Vorgänger

10	Whalen	4400	nicht vorh.	nicht vorh.
20	Fay	6000	Hartstein	nicht vorh.
20	Hartstein	13000	nicht vorh.	Fay
30	Colmenares	2500	Himuro	nicht vorh.
30	Himuro	2600	Tobias	Colmenares
30	Tobias	2800	Baida	Himuro
30	Baida	2900	Khoo	Tobias
30	Khoo	3100	Raphaely	Baida
30	Raphaely	11000	nicht vorh.	Khoo
40	Mavris	6500	nicht vorh.	nicht vorh.
50	Olson	2100	Philtanker	nicht vorh.
50	Philtanker	2200	Markle	Olson
50	Markle	2200	Gee	Philtanker

06 Rangfolge

```
Select
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.HIRE_DATE,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>
From
  EMPLOYEES
Order By
  EMPLOYEES.HIRE_DATE;
```

06 Rangfolge

<Analytische Funktion 1>::=

```
Rank()
  Over(Order By
          EMPLOYEES.HIRE_DATE
        ) RankWithGaps
```

06 Rangfolge

<Analytische Funktion 2>::=

```
Dense_Rank()
  Over(Order By
          EMPLOYEES.HIRE_DATE
       ) RankWithoutGaps
```

06 Rangfolge

De Haan	13.01.01	1	1
Gietz	07.06.02	2	2
Baer	07.06.02	2	2
Mavris	07.06.02	2	2
Higgins	07.06.02	2	2
Faviet	16.08.02	6	3
Greenberg	17.08.02	7	4
Raphaely	07.12.02	8	5
Kaufling	01.05.03	9	6
Khoo	18.05.03	10	7
King	17.06.03	11	8
Ladwig	14.07.03	12	9
Whalen	17.09.03	13	10

07 Rangfolge

```
Select
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.HIRE_DATE,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>
From
  EMPLOYEES
Order By
  EMPLOYEES.HIRE_DATE;
```

07 Rangfolge

<Analytische Funktion 1>::=

```
Rank()
  Over(Partition By
         To_Char(EMPLOYEES.HIRE_DATE,'YYYY')
       Order By
         EMPLOYEES.HIRE_DATE
       ) RankYearWithGaps
```

07 Rangfolge

<Analytische Funktion 2>::=

```
Row_Number()
  Over(Order By
          EMPLOYEES.HIRE_DATE
       ) ConsNumbering
```

07 Rangfolge

De Haan	13.01.01	1	1
Gietz	07.06.02	1	2
Baer	07.06.02	1	3
Mavris	07.06.02	1	4
Higgins	07.06.02	1	5
Faviet	16.08.02	5	6
Greenberg	17.08.02	6	7
Raphaely	07.12.02	7	8
Kaufling	01.05.03	1	9
Khoo	18.05.03	2	10
King	17.06.03	3	11
Ladwig	14.07.03	4	12
Whalen	17.09.03	5	13

08 Erster/letzter Wert in Gruppe

```
Select
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>,
  <Analytische Funktion 3>
From
  EMPLOYEES
Order By
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.SALARY;
```

08 Erster/letzter Wert in Gruppe

<Analytische Funktion 1>::=

```
EMPLOYEES.SALARY -
First_Value(EMPLOYEES.SALARY)
  Over(Partition By
         EMPLOYEES.DEPARTMENT_ID
       Order By
         EMPLOYEES.SALARY
       ) DiffLowSalDep
```

08 Erster/letzter Wert in Gruppe

```
<Analytische Funktion 2>::=
  Last_Value(EMPLOYEES.SALARY)
    Over(Partition By
           EMPLOYEES.DEPARTMENT_ID
         Order By
           EMPLOYEES.SALARY
        ) HighSalDepWrong,
  -- Werte nicht wie erwartet, da Standard für
     'windowing_clause' gleich 'Range Between Unbounded
     Preceding And Current Row' !!!
```

08 Erster/letzter Wert in Gruppe

<Analytische Funktion 3>::=

```
Last_Value(EMPLOYEES.SALARY)
  Over(Partition By
         EMPLOYEES.DEPARTMENT_ID
       Order By
         EMPLOYEES.SALARY
       Range Between Unbounded Preceding And
       Unbounded Following
      ) HighSalDepRight
```

08 Erster/letzter Wert in Gruppe

Whalen	10	4400	0	4400	4400
Fay	20	6000	0	6000	13000
Hartstein	20	13000	7000	13000	13000
Colmenares	30	2500	0	2500	11000
Himuro	30	2600	100	2600	11000
Tobias	30	2800	300	2800	11000
Baida	30	2900	400	2900	11000
Khoo	30	3100	600	3100	11000
Raphaely	30	11000	8500	11000	11000
Mavris	40	6500	0	6500	6500
Olson	50	2100	0	2100	8200
Philtanker	50	2200	100	2200	8200
Markle	50	2200	100	2200	8200

09 Physikalische Rows

```
Select
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>
From
  EMPLOYEES
Order By
  EMPLOYEES.DEPARTMENT_ID,
  EMPLOYEES.LAST_NAME;
```

09 Physikalische Rows

```
<Analytische Funktion 1>::=
  Sum(EMPLOYEES.SALARY)
    Over(Partition By
           EMPLOYEES.DEPARTMENT_ID
         Order By
           EMPLOYEES.LAST_NAME
         Rows 2 Preceding
        ) SumSalPerDep
  -- Die zwei vorhergehende Rows und die aktuelle Row
  -- Kurz für 'Rows Between 2 Preceding And Current Row'
```

09 Physikalische Rows

10	Whalen	4400	4400
20	Fay	6000	6000
20	Hartstein	13000	19000
30	Baida	2900	2900
30	Colmenares	2500	5400
30	Himuro	2600	8000
30	Khoo	3100	8200
30	Raphaely	11000	16700
30	Tobias	2800	16900
40	Mavris	6500	6500
50	Atkinson	2800	2800
50	Bell	4000	6800
50	Bissot	3300	10100

10 Logischer Range

```
Select
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.HIRE_DATE,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>
From
  EMPLOYEES
Order By
  EMPLOYEES.HIRE_DATE,
  EMPLOYEES.LAST_NAME;
```

10 Logischer Range

<Analytische Funktion 1>::=

```
First_Value(EMPLOYEES.LAST_NAME)
  Over(Order By
          EMPLOYEES.HIRE_DATE
        Range Between 100 Preceding And 100 Following
       ) EmpMax100DaysBefore,
-- Angestellter (kleinster Datums-Wert), der höchstens
   hundert Tage vorher eingestellt wurde
```

10 Logischer Range

<Analytische Funktion 2>::=

```
Last_Value(EMPLOYEES.LAST_NAME)
  Over(Order By
          EMPLOYEES.HIRE_DATE
        Range Between 100 Preceding And 100 Following
       ) EmpMax100DaysAfter,
-- Angestellter (kleinster Datums-Wert), der höchstens
   hundert Tage nachher eingestellt wurde
```

10 Logischer Range

De Haan	13.01.01	De Haan	De Haan
Baer	07.06.02	Baer	Greenberg
Gietz	07.06.02	Baer	Greenberg
Higgins	07.06.02	Baer	Greenberg
Mavris	07.06.02	Baer	Greenberg
Faviet	16.08.02	Baer	Greenberg
Greenberg	17.08.02	Baer	Greenberg
Raphaely	07.12.02	Raphaely	Raphaely
Kaufling	01.05.03	Kaufling	Ladwig
Khoo	18.05.03	Kaufling	Ladwig
King	17.06.03	Kaufling	Whalen
Ladwig	14.07.03	Kaufling	Rajs
Whalen	17.09.03	King	Rajs

11 Logischer Range

```
Select
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.HIRE_DATE,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>
From
  EMPLOYEES
Order By
  EMPLOYEES.HIRE_DATE,
  EMPLOYEES.LAST_NAME;
```

11 Logischer Range

<Analytische Funktion 1>::=

```
Round(Avg(EMPLOYEES.SALARY)
        Over(Order By
                EMPLOYEES.HIRE_DATE
              Range Between Numtoyminterval(2,'Month')
              Preceding And Numtoyminterval(2,'Month')
              Following
             ),2
      ) AvgSal2MonthBefore2MonthAfter
```

11 Logischer Range

Haan	13.01.01	17000	17000
Baer	07.06.02	10000	9202
Gietz	07.06.02	8300	9202
Higgins	07.06.02	12008	9202
Mavris	07.06.02	6500	9202
Faviet	16.08.02	9000	10504
Greenberg	17.08.02	12008	10504
Raphaely	07.12.02	11000	11000
Kaufling	01.05.03	7900	11666,67
Khoo	18.05.03	3100	9650
King	17.06.03	24000	9650
Ladwig	14.07.03	3600	10233,33
Whalen	17.09.03	4400	3950

12 Doppelte Werte

```
Select
  EMPLOYEES_LIST.LAST_NAME,
  EMPLOYEES_LIST.FIRST_NAME,
  EMPLOYEES_LIST.LAST_NAME_NEXT,
  EMPLOYEES_LIST.FIRST_NAME_NEXT
From
  <Sub-Query 1>
Where
  EMPLOYEES_LIST.LAST_NAME =
  EMPLOYEES_LIST.LAST_NAME_NEXT
Order By
  EMPLOYEES_LIST.LAST_NAME,
  EMPLOYEES_LIST.FIRST_NAME;
```

12 Doppelte Werte

```
<Sub-Query 1>::=
 (
  Select
    EMPLOYEES.FIRST_NAME,
    EMPLOYEES.LAST_NAME,
    <Analytische Funktion 1>,
    <Analytische Funktion 2>
  From
    EMPLOYEES
 ) EMPLOYEES_LIST
```

12 Doppelte Werte

<Analytische Funktion 1>::=

```
    Lead(EMPLOYEES.FIRST_NAME,
            1,
            'nicht vorhanden‘
           )
      Over(Order By
              EMPLOYEES.LAST_NAME
            ) FIRST_NAME_NEXT
```

12 Doppelte Werte

<Analytische Funktion 2>::=

```
    Lead(EMPLOYEES.LAST_NAME,
            1,
            'nicht vorhanden‘
           )
      Over(Order By
              EMPLOYEES.LAST_NAME
            ) LAST_NAME_NEXT
```

12 Doppelte Werte

Cambrault	Gerald	Cambrault	Nanette
Grant	Douglas	Grant	Kimberely
King	Janette	King	Steven
Smith	Lindsey	Smith	William
Taylor	Jonathon	Taylor	Winston

13 Kleinster Wert

```
Select
  EMPLOYEES.MANAGER_ID,
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.HIRE_DATE,
  EMPLOYEES.SALARY,
  <Analytische Funktion 1>
From
  EMPLOYEES
Order By
  EMPLOYEES.MANAGER_ID,
  EMPLOYEES.LAST_NAME,
  EMPLOYEES.HIRE_DATE,
  EMPLOYEES.SALARY;
```

13 Kleinster Wert

<Analytische Funktion 1>::=

```
Min(EMPLOYEES.SALARY)
  Over(Partition By
         EMPLOYEES.MANAGER_ID
       Order By
         EMPLOYEES.HIRE_DATE
       Range Between Unbounded Preceding And
       Current Row
      ) MinSalEmp
```

13 Kleinster Wert

101	Mavris	07.06.02	6500	6500
101	Baer	07.06.02	10000	6500
101	Higgins	07.06.02	12008	6500
101	Greenberg	17.08.02	12008	6500
101	Whalen	17.09.03	4400	4400
102	Hunold	03.01.06	9000	9000
103	Austin	25.06.05	4800	4800
103	Pataballa	05.02.06	4800	4800
103	Lorentz	07.02.07	4200	4200
103	Ernst	21.05.07	6000	4200
108	Faviet	16.08.02	9000	9000
108	Chen	28.09.05	8200	8200
108	Sciarra	30.09.05	7700	7700

Performance (Vorbereitung)

```
Create Table
  BIG_EMPLOYEES
As
Select
  ALL_OBJECTS.OBJECT_NAME LAST_NAME,
  Mod(ALL_OBJECTS.OBJECT_ID,50) DEPARTMENT_ID,
  ALL_OBJECTS.OBJECT_ID SALARY
From
  ALL_OBJECTS
Where
  Rownum <= 5000;
  -- Testtabelle mit 5000 Einträgen
```

Performance (mit analytic)

```
Select
  BIG_EMPLOYEES.LAST_NAME,
  BIG_EMPLOYEES.DEPARTMENT_ID,
  BIG_EMPLOYEES.SALARY,
  <Analytische Funktion 1>,
  <Analytische Funktion 2>,
  <Analytische Funktion 3>
From
  BIG_EMPLOYEES
Order By
  BIG_EMPLOYEES.DEPARTMENT_ID,
  BIG_EMPLOYEES.LAST_NAME;
```

Performance (mit analytic)

<Analytische Funktion 1>::=

```
Sum(BIG_EMPLOYEES.SALARY)
  Over(Order By
          BIG_EMPLOYEES.DEPARTMENT_ID,
          BIG_EMPLOYEES.LAST_NAME
        ) CurrSumEmpSal
```

Performance (mit analytic)

<Analytische Funktion 2>::=

```
Sum(BIG_EMPLOYEES.SALARY)
  Over(Partition By
         BIG_EMPLOYEES.DEPARTMENT_ID
       Order By
         BIG_EMPLOYEES.LAST_NAME
      ) CurrSumDepSal
```

Performance (mit analytic)

<Analytische Funktion 3>::=

```
Row_Number()
  Over(Partition By
         BIG_EMPLOYEES.DEPARTMENT_ID
       Order By
         BIG_EMPLOYEES.LAST_NAME
      ) CurrDepCount
```

Performance (ohne analytic)

```
Select
  BIG_EMPLOYEES.LAST_NAME,
  BIG_EMPLOYEES.DEPARTMENT_ID,
  BIG_EMPLOYEES.SALARY,
  <Sub-Query 1>,
  <Sub-Query 2>,
  <Sub-Query 3>
From
  BIG_EMPLOYEES
Order By
  BIG_EMPLOYEES.DEPARTMENT_ID,
  BIG_EMPLOYEES.LAST_NAME;
```

Performance (ohne analytic)

```
<Sub-Query 1>::=
 (
  Select
   Sum(BIG_EMPLOYEES_2.SALARY)
  From
   BIG_EMPLOYEES BIG_EMPLOYEES_2
  Where
   BIG_EMPLOYEES_2.DEPARTMENT_ID < … Or
   (
    BIG_EMPLOYEES_2.DEPARTMENT_ID = … And
    BIG_EMPLOYEES_2.LAST_NAME <= …
   )
 ) CurrSumEmpSal
```

Performance (ohne analytic)

```
<Sub-Query 2>::=
 (
  Select
   Sum(BIG_EMPLOYEES_3.SALARY)
  From
   BIG_EMPLOYEES BIG_EMPLOYEES_3
  Where
   BIG_EMPLOYEES_3.DEPARTMENT_ID =
   BIG_EMPLOYEES.DEPARTMENT_ID And
   BIG_EMPLOYEES_3.LAST_NAME <=
   BIG_EMPLOYEES.LAST_NAME
 ) CurrSumDepSal
```

Performance (ohne analytic)

<Sub-Query 3>::=

```
 (
   Select
     Count(BIG_EMPLOYEES_4.LAST_NAME)
   From
     BIG_EMPLOYEES BIG_EMPLOYEES_4
   Where
     BIG_EMPLOYEES_4.DEPARTMENT_ID =
     BIG_EMPLOYEES.DEPARTMENT_ID And
     BIG_EMPLOYEES_4.LAST_NAME <=
     BIG_EMPLOYEES.LAST_NAME
 ) CurrDepCount
```

Performance (Vergleich)

Mit analytischen Funktionen:

- Abfragezeit ca. 0,14s
- 25 Zugriffe

Ohne analytische Funktionen:

- Abfragezeit ca. 6,2s
- 375.025 Zugriffe

etwa Faktor 30 - 40

Fazit

Analytische Funktionen sind

einfach zu programmieren

sehr mächtig

performant

Kontakt

Ewald GmbH
Miesbacher Str. 38a
D-83620 Feldkirchen-Westerham

Telefon: +49 (0) 8063-256 8063
Fax: +49 (0) 8063-256 8064
E-Mail juergen.habdank@ewald.de
Internet: www.ewald.de