Mittwoch, 13.02.2008, 17:00 Uhr

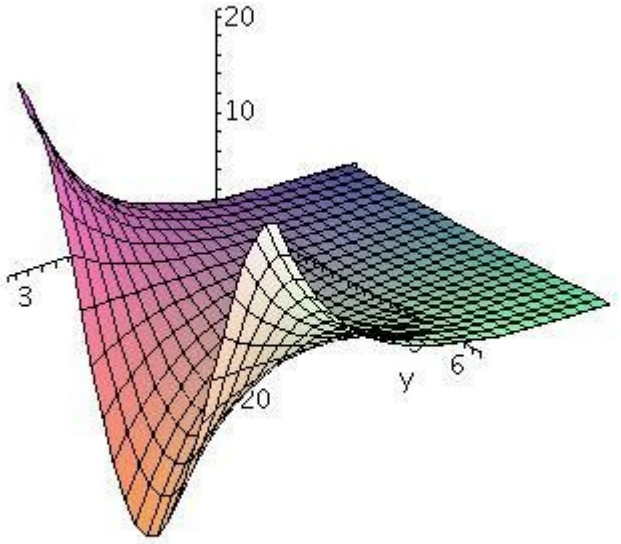
DOAG
Deutsche ORACLE-Anwendergruppe e.V.
Regionaltreffen
München/Südbayern

# Oracle Analytic Functions

**Seit Jahren auf dem Markt (8.1.6),**
**jedoch unbekannt und selten im Einsatz**

# Agenda

- Einsatzmöglichkeiten
- Syntax und Grundlagen
- Wie und wo kann es eingesetzt werden
- Braucht ein DBA das überhaupt?
- Performance-Überlegungen



# Analytic Functions (1)

- Vereinfacht die zum Teil sehr komplexe Fragestellung bei der Datenauswertung
- Verfügbar ab Oracle 8i (SQL)
- In PL/SQL erst ab 9i verfügbar
- Bei Problemen in 9i siehe Note:375710.1

# Analytic Functions (1)

- Fragestellungen der Anwender:
  - Was sind die Top-3-Produkte (nach Umsatz) je Region?
  - Welches ist die Region mit dem größten Umsatz jedes  Produkts?
  - Welchen Anteil am Gesamtumsatz der Region haben die Top-3-Produkte der Region ?
  - Wie hoch war der prozentuale Anteil des Umsatzes jedes Monats am Gesamtumsatz des Jahres ?
  - Wie groß ist die Änderung des prozentualen Anteils eines Produkts am Umsatz gegenüber dem Vorjahr ?

# Analytic Functions (2)

- Fragestellungen für den DBA:
  - Fehlt ein Wert in einer Folge von Daten?
    - Gibt es Lücken in meinen Audit Logs der DB?
  - Können doppelte Daten aus einer Tabelle gefiltert werden?
  - Wie groß sind meine Archivelogs usw. über den Tag/Monat verteilt in Prozent, und Abweichung dazu usw.
    - Wie viel Prozent werden in welchen Stunden geschrieben?

# Funktion auf der Ergebnismenge

- Definition: „Analytische Funktion“
  - Als analytisch bezeichnet man in der Mathematik eine Funktion, die lokal durch eine konvergente Potenzreihe gegeben ist

---

Es sei $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$. Es sei $D \subseteq \mathbb{K}$ eine offene Teilmenge. Eine Funktion $f : D \to \mathbb{K}$ heißt **analytisch** im Punkte $x_0 \in D$, wenn es eine Potenzreihe

$$\sum_{n=0}^{\infty} a_n (x - x_0)^n$$

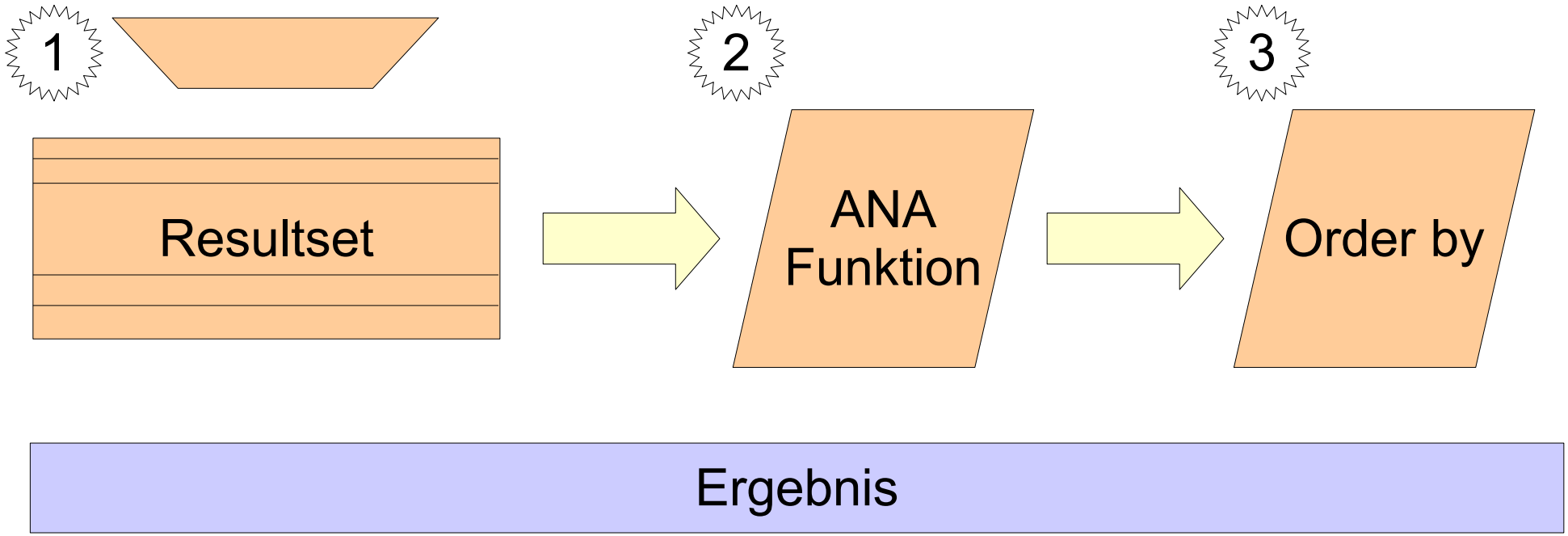
gibt, die auf einer Umgebung von $x_0$ gegen $f(x)$ konvergiert. Ist $f$ in jedem Punkte von $D$ analytisch, so heißt f **analytisch.**

- Oracle ANA => Funktionen auf einer Menge der Ergebnisdaten durchführen

# Ergebnismenge verarbeiten

- Funktionen auf der Ergebnismenge einer Abfrage durchführen

```
Select ..... ana_func OVER ()  from tab a group by wert order by 1
```



# Die Syntax verstehen (1)

- Partition by
  - Ergebnismenge partitionieren

```
Select spalte1,spalte2
```

```
function OVER ( MENGE SORTIERUNG FENSTER )
```

```
from tab
group by spalte1,spalte2
```

(

)

MENGE


```
Partition by
```

# Arbeiten auf einer Menge - Partition

- Fragestellung:
  - Wie hoch ist das Gehalt eines Mitarbeiters im Vergleich zum Durchschnittsgehalt in seiner Abteilung?

```
select ename, deptno, sal,
         avg(sal) over (partition by deptno) as durchschnitt
  from emp
```


| ENAME | DEPTNO | SAL | DURCHSCHNITT |
|---|---|---|---|
| CLARK | 10 | 2450 | 2083,33333 |
| MILLER | 10 | 1300 | 2083,33333 |
| ... | | | |
| SMITH | 20 | 800 | 2175 |
| .... | | | |
| ALLEN | 30 | 1601 | 1567,66667 |
| BLAKE | 30 | 2851 | 1567,66667 |

# Vergleich

- ANA zur Standard-Lösung

```
select ename, deptno, sal,
         avg(sal) over (partition by deptno) as durchschnitt
  from emp
```

```
select e.ename, e.deptno, e.sal, s.durchschnitt
  from emp e, (select avg(sal) as durchschnitt,deptno
                 from emp
                group by deptno) s
 where s.deptno=e.deptno
 order by e.deptno
```

# Die Syntax verstehen (2)

- Order by
  - „Vergleichsmenge“ sortiert betrachten

```
Select spalte1,spalte2
```

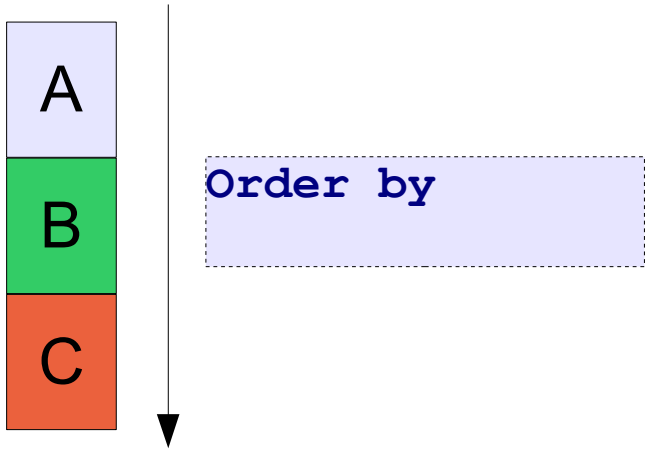
```
function OVER ( MENGE SORTIERUNG FENSTER )
```

```
from tab
group by spalte1,spalte2
```

(

SORTIERUNG

)



# Reihenfolge innerhalb der Ergebnismenge

- Fragestellung:
  - Wie ist die Reihenfolge der Gehälter der Mitarbeiter?

```
select empno, ename, deptno,  sal ,
           rank () over (order by sal) as rang
  from emp
 order by deptno
```

```
  EMPNO ENAME          DEPTNO       SAL      RANG
------- ---------- -------- -------- --------
   7839 KING               10     5000         1
   7902 FORD               20     3000         2
   ......
   7876 ADAMS              20     1100        12
   7788 SCOTT              20     3000         2
   7900 JAMES              30      950        14
   7654 MARTIN             30     1250        10
    ...
   7844 TURNER             30     1500         8
```

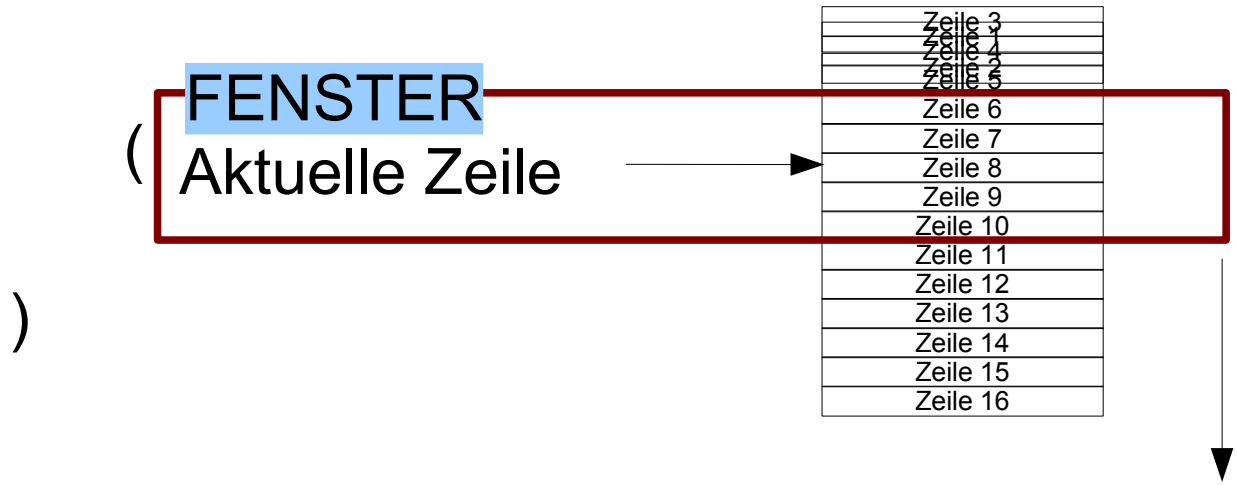
# Die Syntax verstehen (3)

- Window
  - Auf Ergebnismenge zeilenweise arbeiten

```
Select spalte1,spalte2
```

```
function OVER ( MENGE SORTIERUNG FENSTER )
```

```
from tab
group by spalte1,spalte2
```




```
rows
between .. preceding
and
.. following
```

# Moving-Window-Klausel

- Physical Units
  - rows between 1 preceding and n following
- Logical Units
  - Numbers:
    - range between  1000 preceding and n following
  - Dates:
    - range between 30 preceding and 10 following
    - range between interval ‘1’ MONTH preceding and interval ‘2’ YEAR  following
  - Custom
    - range between myFunc(x) preceding and  myFunc2(x)  following

# Ein Fenster über die Ergebnismenge schieben

- Window Function

```
...
avg(sal)
over (order by hiredate
        rows between 1 preceding
                 and 1 following)
....
```

```
select ename, hiredate,sal
  from emp order by hiredate
```

| ENAME | HIREDATE | SAL |
|---|---|---|
| SMITH | 17.12.80 | 800 |
| ALLEN | 20.02.81 | 1601 |
| WARD | 22.02.81 | 1251 |
| JONES | 02.04.81 | 2975 |
| BLAKE | 01.05.81 | 2851 |
| CLARK | 09.06.81 | 2450 |
| TURNER | 08.09.81 | 1501 |
| MARTIN | 28.09.81 | 1251 |
| KING | 17.11.81 | 2500 |
| JAMES | 03.12.81 | 951 |
| FORD | 03.12.81 | 3000 |
| MILLER | 23.01.82 | 1300 |
| SCOTT | 19.04.87 | 3000 |
| ADAMS | 23.05.87 | 1100 |

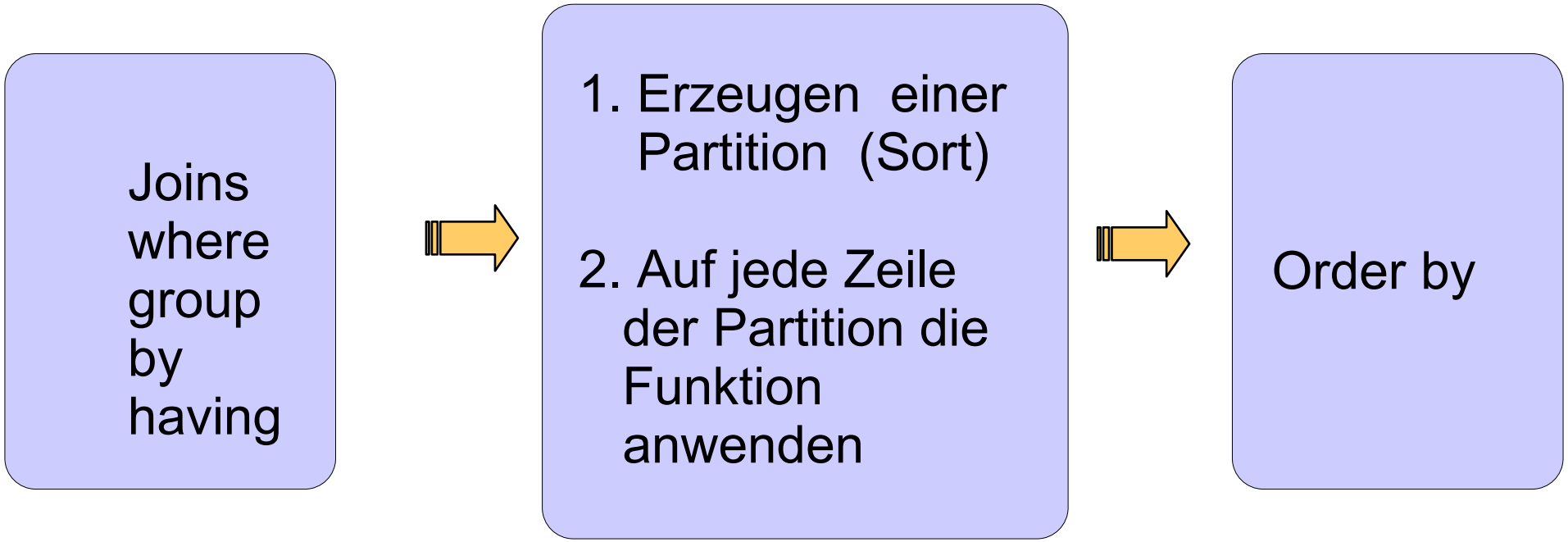
Mittelwert

| ENAME | AVG |
|---|---|
| SMITH | 1201 |
| ALLEN | 1217 |
| WARD | 1942 |
| JONES | 2359 |
| BLAKE | 2759 |
| CLARK | 2267 |
| TURNER | 1734 |
| MARTIN | 1751 |
| KING | 1567 |
| JAMES | 2150 |
| FORD | 1750 |
| MILLER | 2433 |
| SCOTT | 1800 |
| ADAMS | 2050 |

Mittelwert : (1601 +1251 +2975) / 3 = 1942

# SQL-Ausführungsplan

## Ablauf




```
Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=4 Card=14 Bytes=140)
   1    0   WINDOW (SORT) (Cost=4 Card=14 Bytes=140)
   2    1     TABLE ACCESS (FULL) OF 'EMP' (Cost=2 Card=14 Bytes=140)
```

# ANA-Funktionen (1)

- Reporting
  - z.B. prozentualer Anteil des Umsatzes eines Monats auf das Jahr gesehen
- Windowing
  - z.B. Auswerten von je +/- 100 Tagen  vom jeweiligen Tag
- Ranking
  - z.B.  zeige die 3 ersten Mitarbeiter je Region und Produkt

# ANA-Funktionen (2)

- Lag/Lead
  - z.B. Unterschied zwischen zwei Bereichen/Quartalen
  - Lücken in Daten finden
- Statistics
  - z.B. lineare Regression

# ANA-Funktionen selbst erstellen

- SELF-DEFINED ANALYTIC FUNCTIONS
- Siehe:
- http://www.oracle.com/technology/oramag/oracle/06-jul/o46sql.html
- http://www.oracle-developer.net/display.php?id=215

# Report-Funktionen

- Sum()
- Avg()
- Max()
- Min()
- Count()
- Stddev()
- Variance()
- Ratio_to_Report()

# Prozentualer Vergleich

- Prozentualen Anteil auf die Gesamtmenge
  - Blocks pro Thread im Monat / Prozent auf gesamt im Monat

```
Select thread#
     , (TO_CHAR( completion_time, 'yyyymm' )) as Monat
     , sum( blocks ) as blocks
     , ratio_to_report( sum( blocks ))
        over(partition by TO_CHAR( completion_time, 'yyyymm' ))
        as Knoten_Prozent
  from v$archived_log
 group by (TO_CHAR( completion_time, 'yyyymm' )),thread#
 order by thread#
```

```
 THREAD#   |MONAT     |BLOCKS       |KNOTEN_PROZENT
-----------------------------------------------------
1          |200801    |173500720    |0,17
2          |200801    |114996078    |0,12
3          |200801    |274980380    |0,28
4          |200801    |106618654    |0,11
5          |200801    |149133772    |0,15
6          |200801    |177172714    |0,18
```

# Reporting Beispiel ratio_to_report

- Fragestellung: wie hoch ist der Anteil eines einzelnen Gehalts an der Summe der Gehälter ?

- ANA-Lösung:

```
select ename, deptno, sal,
       ratio_to_report(sal) over () as Prozent
from emp
order by sal
```

3 consistent gets

- Standard:

```
select   ename, deptno, sal
       , sal/Gesamt as Prozent
  from emp, (select sum(sal) as Gesamt
               from emp)
```

```
ENAME          DEPTNO        SAL    PROZENT
---------- ---------- ---------- ----------
SMITH              20        800 ,030153405
JAMES              30        951 ,035844861
ADAMS              20       1100 ,041460932
WARD               30       1251 ,0471
.....
```

7 consistent gets

# Reporting Beispiel Sum (1)

- Fragestellung: wie hoch ist das Gehalt eines Angestellten im Durchschnitt

- ANA-Lösung:

```
select ename, deptno, sal,
         avg(sal) over () as durchschnitt
  from emp
```

- Standard:

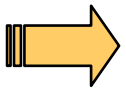
```
select ename, deptno, sal, durchschnitt
  from emp, (select avg(sal) as durchschnitt
               from emp)
```

```
ENAME          DEPTNO        SAL DURCHSCHNITT
---------- ---------- ---------- ------------
SMITH              20        800   1895,07143
ALLEN              30       1601   1895,07143
WARD               30       1251   1895,07143
.....
```

# Reporting Beispiel Sum (2)

- Ausführungsplan
- ANA-Lösung:

```
select ename, deptno, sal,
       avg(sal) over () as durchschnitt
  from emp
```



```
Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=14 Bytes=140)
   1    0   WINDOW (BUFFER) (Cost=2 Card=14 Bytes=140)
   2    1     TABLE ACCESS (BY INDEX ROWID) OF 'EMP' (Cost=2 Card=14 Bytes=140)
   3    2       INDEX (FULL SCAN) OF 'PK_EMP' (UNIQUE) (Cost=1 Card=14)
```

2 consistent gets

- Standard:

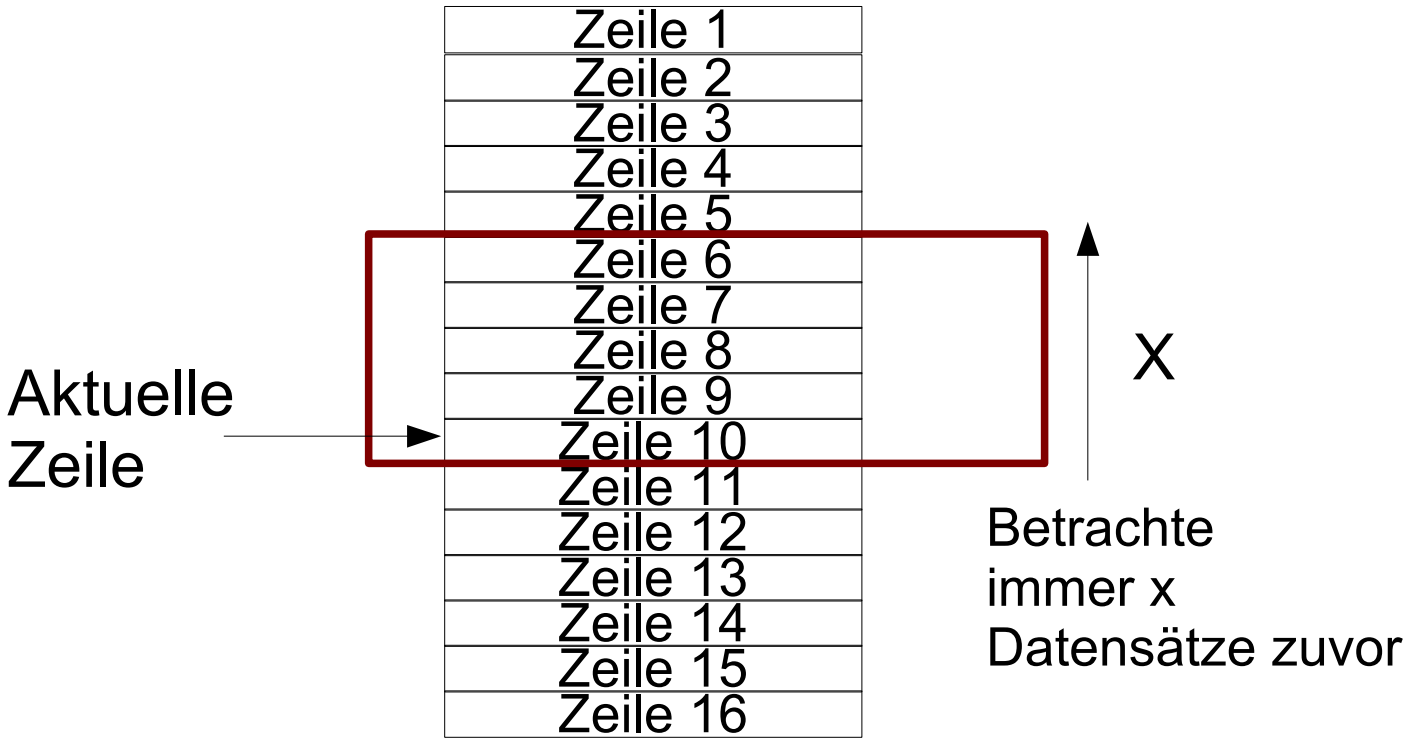
```
select ename, deptno, sal, durchschnitt
  from emp, (select avg(sal) as durchschnitt
               from emp)
```

```
Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=4 Card=14 Bytes=322)
   1    0   MERGE JOIN (CARTESIAN) (Cost=4 Card=14 Bytes=322)
   2    1     VIEW (Cost=2 Card=1 Bytes=13)
   3    2       SORT (AGGREGATE)
   4    3         TABLE ACCESS (FULL) OF 'EMP' (Cost=2 Card=14 Bytes=42)
   5    1     TABLE ACCESS (FULL) OF 'EMP' (Cost=2 Card=14 Bytes=140)
```

7 consistent gets

# Window-Funktionen

- Sum()
- Avg()
- Max()
- Min()
- Count()
- Sttdev()
- Variance()
- First_Value()
- Last_Value()



# Moving-Window-Beispiel AVG (1)

- Fragestellung: gesucht ist der Mittelwert der Gehälter von drei Mitarbeitern sortiert nach Einstellungsdatum

```
select ename, hiredate,sal,
       round(
              avg(sal) over (order by hiredate
                             rows between 1 preceding and 1 following)
       )  as AVG
  from emp
```

Sortierung notwendig!

Größe des Windows

| ENAME | HIREDATE | SAL | AVG |
|---|---|---|---|
| SMITH | 17.12.80 | 800 | 1201 |
| ALLEN | 20.02.81 | 1601 | 1217 |
| WARD | 22.02.81 | 1251 | 1942 |
| JONES | 02.04.81 | 2975 | 2359 |
| BLAKE | 01.05.81 | 2851 | 2759 |
| CLARK | 09.06.81 | 2450 | 2267 |
| ..... | | | |
| FORD | 03.12.81 | 3000 | 1750 |
| MILLER | 23.01.82 | 1300 | 2433 |
| SCOTT | 19.04.87 | 3000 | 1800 |
| ADAMS | 23.05.87 | 1100 | 2050 |

# Moving-Window-Beispiel 2

- Fragestellung: Gehalt eines Angestellten im Vergleich zum Durchschnittsgehalt all derjenigen, die bis zu 6 Monaten vor demjenigen eingestellt wurden.

```
select ename,sal, hiredate,

       avg(sal) over (order by hiredate
                      range  interval '6' MONTH preceding)
from emp
```

Sortierung notwendig!

Größe des Windows beachte die letzten 6 Monate

```
ENAME              SAL HIREDATE DURCHSCHNITT
---------- ---------- -------- ------------
SMITH              800 17.12.80          800
ALLEN             1601 20.02.81       1200,5
WARD              1251 22.02.81   1217,33333
JONES             2975 02.04.81      1656,75
BLAKE             2851 01.05.81       1895,6
CLARK             2450 09.06.81         1988
TURNER            1501 08.09.81      2444,25
.............
```

# Moving-Window-Beispiel 3

- Fragestellung: Gehalt eines Angestellten im Vergleich zum Durchschnittsgehalt derjenigen, die bis zu 6 Monaten vor diesem eingestellt wurden, gestaffelt nach Abteilung

```
select ename,deptno,sal, hiredate,

       avg(sal) over (partition by deptno
                          order by hiredate
                          range  interval '6' MONTH preceding)
from emp
```

Gruppiert nach Abteilung

Sortierung notwendig!

Größe des Windows, beachte die letzten 6 Monate

| ENAME | DEPTNO | SAL | HIREDATE | DUCHSCHNITT |
|---|---|---|---|---|
| CLARK | 10 | 2450 | 09.06.81 | 2450 |
| KING | 10 | 2500 | 17.11.81 | 2475 |
| MILLER | 10 | 1300 | 23.01.82 | 1900 |
| SMITH | 20 | 800 | 17.12.80 | 800 |
| JONES | 20 | 2975 | 02.04.81 | 1887,5 |
| FORD | 20 | 3000 | 03.12.81 | 3000 |
| SCOTT | 20 | 3000 | 19.04.87 | 3000 |
| ADAMS | 20 | 1100 | 23.05.87 | 2050 |
| ALLEN | 30 | 1601 | 20.02.81 | 1601 |
| ..... | | | | |

# Window-Beispiel Sum (1)

- Fragestellung: laufende Summe der Gehälter in Emp
- ANA-Lösung:

```
select ename, deptno, sal,
       sum(sal) over (order by sal,rownum)
       as LaufendesGehalt
  from emp
```

Eindeutigkeit Sicherstellen!

Auf richtige Sortierung achten

- Ergebnis:

| ENAME | DEPTNO | SAL | LAUFENDESGEHALT |
|---|---|---|---|
| SMITH | 20 | 800 | 800 |
| JAMES | 30 | 951 | 1751 |
| ADAMS | 20 | 1100 | 2851 |
| WARD | 30 | 1251 | 4102 |
| MARTIN | 30 | 1251 | 5353 |
| MILLER | 10 | 1300 | 6653 |
| ..... | | | |

# Ranking-Funktionen

- rank()
- dense_rank()
- row_number()
- cume_dist()
- percent_rank()
- Ntile()
- ROW_NUMBER()

# Ranking-Beispiel 1

- Fragestellung: wie ist die Reihenfolge der Gehälter in Emp ?
- ANA-Lösung:

```
select empno, ename, deptno,  sal ,
           rank () over (order by sal) as rang
  from emp
 order by rang
```

3 consistent gets

- Standard:

```
select ename , empno, sal, t.ro
  from emp e,
       (select sal sa,
               min(rownum) ro
          from (select sal
                  from emp
                 order by sal
               )
         group by sal
       )  t
 where e.sal  = t.sa
 order by sal
```

6 consistent gets

| EMPNO | ENAME | DEPTNO | SAL | RANG |
|---|---|---|---|---|
| 7369 | SMITH | 20 | 800 | 1 |
| 7900 | JAMES | 30 | 951 | 2 |
| ..... | | | | |
| 7934 | MILLER | 10 | 1300 | 6 |
| 7844 | TURNER | 30 | 1501 | 7 |
| 7499 | ALLEN | 30 | 1601 | 8 |
| ..... | | | | |

# Ranking-Beispiel 2/1

- Fragestellung: doppelte Zeilen finden
- Problem:

| DEPTNO | DNAME | LOC |
|---|---|---|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |

# Ranking-Beispiel 2/3

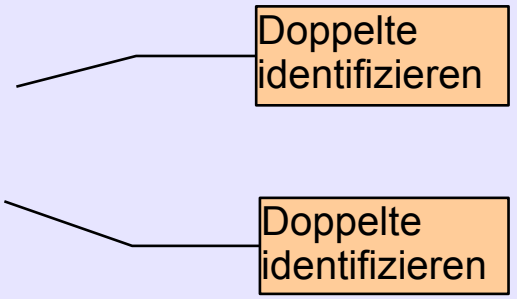
- Fragestellung: doppelte Zeilen finden
- Finden:

```
select deptno
     , rowid

, ROW_NUMBER( )
          over( partition by deptno order by deptno ) rank

 from scott.dept
```

```
  DEPTNO ROWID                  RN
-------- ------------------ --------
      10 AAAfMiAAEAAA638AAA         1
      10 AAAfMiAAEAAA63/AAA         2
      20 AAAfMiAAEAAA63/AAB         1
      20 AAAfMiAAEAAA638AAB         2
      30 AAAfMiAAEAAA63/AAC         1
      40 AAAfMiAAEAAA63/AAD         1
```



# Ranking-Beispiel 2/4

- Fragestellung: doppelte Zeilen finden
- Löschen:

```
delete from scott.dept2
       where rowid in(
 select rowid from ( select rowid

, ROW_NUMBER( )
         over( partition by deptno order by deptno ) rank

from scott.dept2 )
where rank > 1 );
```

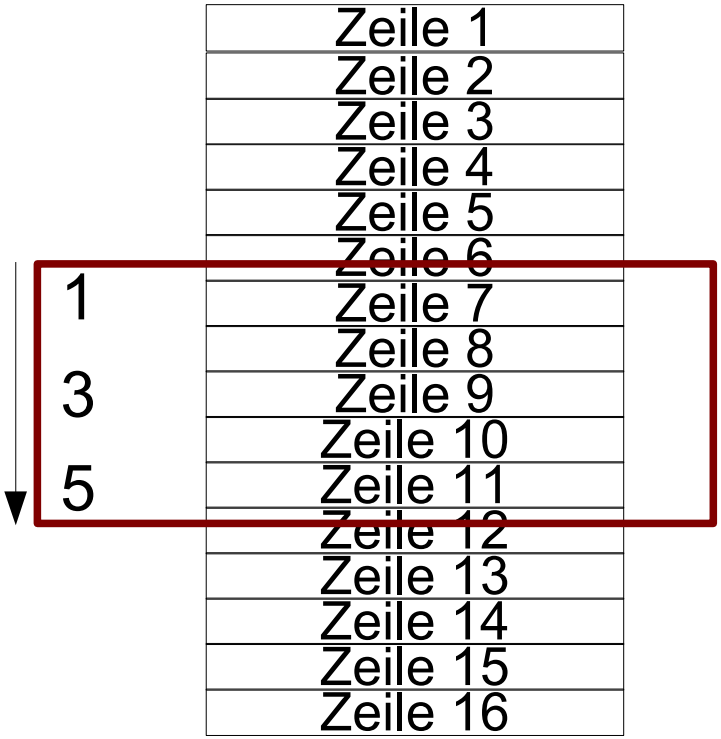
# Lag, Lead Functions

- Lag() (nachlauf) und  Lead() (erster)
  - Lücken finden über den Vorgängerwert

LAG() = 1

Aktuelle  Zeile

LEAD() = 5



# Beispiel: Lücken finden (1)

- Mit Hilfe von Lag/Lead

```
create table w (id number, wert varchar2(1));
```

```
insert into w values (1,'A');
insert into w values (4,'C');
insert into w values (4,'C');
insert into w values (7,'D');
insert into w values (9,'F');
```

# Beispiel: Lücken finden (2)

- Mit Hilfe von Lag

```
GPICONSULT:DEV>select * from
w;
      ID W
------- -
       1 A
       3 B
       4 C
       7 D
       9 F
```

```
GPICONSULT:DEV>select wert, lag(id,1,0) over (order by ID) from w;
W LAG(ID,1,0)OVER(ORDERBYID)
- --------------------------
A                          0
B                          1
C                          3
D                          4
F                          7
```

Wert in der Zeile zuvor

# Beispiel: Lücken finden (2)

- Mit Hilfe von Lead

```
GPICONSULT:DEV>select * from
w;
      ID W
------- -
       1 A
       3 B
       4 C
       7 D
       9 F
```

```
GPICONSULT:DEV>select wert, lead(id,1,0) over (order by ID) from
w;

W LEAD(ID,1,0)OVER(ORDERBYID)
- --------------------------
A                           3
B                           4
C                           7
D                           9
F                           0
```

Wert danach

# Beispiel: Lücken im AUD$ finden

- Mit Hilfe von Lag/Lead – Idee mit interner ROW ID Suchen

```
with aquery as
( select dbms_rowid.rowid_row_number(rowid)
         as after_gap
       , lag( dbms_rowid.rowid_row_number(rowid) , 1, 0 )
         over( order by dbms_rowid.rowid_row_number(rowid)  )
         as before_gap
    from AUD$ )
select   before_gap
       , after_gap
    from aquery
   where before_gap != 0 and after_gap - before_gap > 1
order by before_gap
```

# Problem mit Rollup, Cube

- Rollup für die Berechnung von Untersummen

```
select deptno, job, sum(sal), rank () over (order by sal) as rang
  from emp
 group by rollup(deptno, job);
```

| DEPTNO | JOB | SUM(SAL) | Rang |
|---|---|---|---|
| 10 | CLERK | 1300 | 1 |
| 10 | MANAGER | 2450 | 2 |
| 10 | PRESIDENT | 2500 | 3 |
| 10 | | 6250 | 4 |
| 20 | CLERK | 1900 | 5 |
| 20 | ANALYST | 6000 | 6 |
| 20 | MANAGER | 2975 | 7 |
| 20 | | 10875 | 8 |
| 30 | CLERK | 951 | 9 |
| 30 | MANAGER | 2851 | 10 |
| 30 | SALESMAN | 5604 | 11 |
| 30 | | 9406 | 12 |
| | | 26531 | 13 |

Rollup (Gruppenumbruch) ist im Resultset und wird mit eingerechnet!

# Zusammenfassung

- Analytic Functions
  - <func>() over (partition by ... order by...)

  - Arbeitet auf der Ergebnismenge einer Abfrage
  - Diverse Funktionen für Reporting und Analyse
  - Hohe Performace-Gewinne sind möglich
  - Eigene Funktionen möglich

# Wo gibt es mehr

- Diverse Splitter auf vielen Webseiten
- Oracle-Original-Dokumentation

# Oracle SQL*Net

**Kontakt:**

Gunther Pippèrr

http://www.pipperr.de


## Ihr Ansprechpartner für:

- Oracle Workshops und Training
- Oracle-Projekte mit Forms/Reports
- Java- und XML-Projekte
- Schnittstellen-Entwicklung
- Oracle-Lizenzen
- Remote Wartung und Administration
- Oracle Security

# Statistische Funktionen

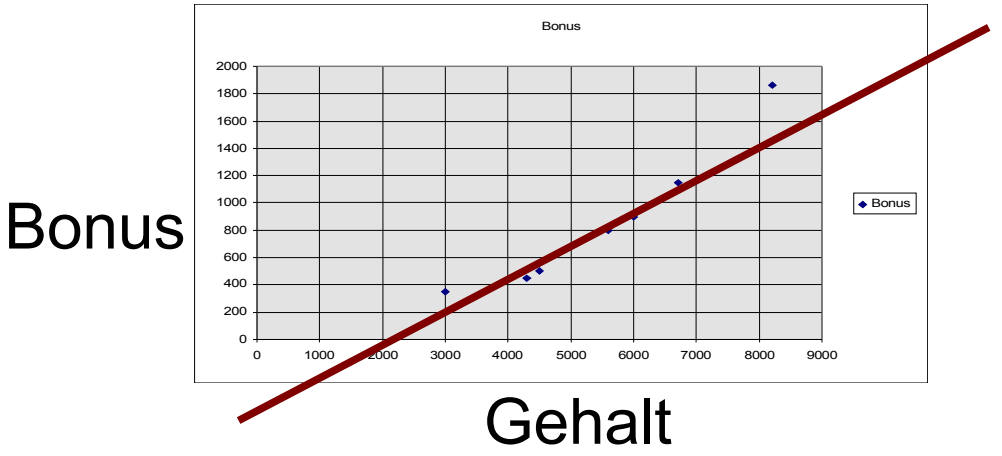
- Var_Pop
  - Population Variance
- Stddev_Pop
- Var_Samp
  - Sample Variance
- Stddev_Samp
- Covar_Pop
  - Population Covariance  of a Set of pairs
- Covar_Samp
  - Sample Covariance of a Set of pairs
- Corr
  - Correlation Coefficient

# Übersicht der wichtigsten Funktionen

- rollup(), cube()
- rank(), dense_rank()
- cume_dist(), percent_rank(), ntile()
- sum(), avg(), max(), min(), count()
- stddev(), variance()
- ratio_to_report()
- lag(), lead()
- Statistik: Var_Pop(), Var_Samp(), Stddev_Pop(), Stddev_Samp(), Covar_Pop(), Covar_Samp()
- Lineare Regression

# Lineare Regression

- Regr_Count #rows used in fit
- Regr_Avgx Average x Value
- Regr_Avgy Average y Value
- Regr_Slope Slope of Curve
- Regr_Intercept y-Axis intercept of Curve
- Regr_R2 r-square Goodness of Fit
- Regr_SXX #rows * var_pop(x)
- Regr_SYY #rows * var_pop(y)
- Regr_SXY #rows * covar_pop(x,y)



# Rollup, Cube

- Rollup für die Berechnung von Untersummen

```
select deptno, job, sum(sal)
  from emp
 group by rollup(deptno, job);
```

```
DEPTNO JOB           SUM(SAL)
---------- --------- ----------
        10 CLERK           1300
        10 MANAGER         2450
        10 PRESIDENT       2500
        10                 6250
        20 CLERK           1900
        20 ANALYST         6000
        20 MANAGER         2975
        20                10875
        30 CLERK            951
        30 MANAGER         2851
        30 SALESMAN        5604
        30                 9406
                          26531
```

# Rollup, Cube, Grouping Set

- CUBE für die Berechnung von Untersummen

```
select deptno, job ,sum(sal),grouping(job),grouping(deptno)
from emp
group by cube (deptno, job)
```

| DEPTNO | JOB | SUM(SAL) | GROUPING(JOB) | GROUPING(DEPTNO) |
|---|---|---|---|---|
| | | 26531 | 1 | 1 |
| | CLERK | 4151 | 0 | 1 |
| | ANALYST | 6000 | 0 | 1 |
| | MANAGER | 8276 | 0 | 1 |
| | SALESMAN | 5604 | 0 | 1 |
| | PRESIDENT | 2500 | 0 | 1 |
| 10 | | 6250 | 1 | 0 |
| 10 | CLERK | 1300 | 0 | 0 |
| 10 | MANAGER | 2450 | 0 | 0 |
| 10 | PRESIDENT | 2500 | 0 | 0 |
| 20 | | 10875 | 1 | 0 |
| 20 | CLERK | 1900 | 0 | 0 |
| 20 | ANALYST | 6000 | 0 | 0 |
| 20 | MANAGER | 2975 | 0 | 0 |
| 30 | | 9406 | 1 | 0 |
| 30 | CLERK | 951 | 0 | 0 |
| 30 | MANAGER | 2851 | 0 | 0 |
| 30 | SALESMAN | 5604 | 0 | 0 |

# Prozentualer Vergleich

- Prozentualer Anteil auf die Gesamtmenge
  - Blocks pro Monat / Prozent auf Gesamt

```
select   sum( blocks ) as blocks

       , ratio_to_report( sum( blocks )) over() as Prozent

    from v$archived_log

group by (TO_CHAR( completion_time, 'yyyymm' ))
```

```
 BLOCKS          PROZENT
------- ---------------
1007100             ,06
5944893             ,37
6552050             ,41
2549620             ,16
```