Analytic SQL

Oracle Analytic SQL

- Anderer Name: „Window functions“
- Ab 8i
- Einfache Ansätze für komplexe Problemstellungen
  - Mit Standard-SQL meist nicht oder nicht elegant zu lösen
  - Effizientere Abarbeitung als mit Standard-SQL
  - Vermeiden von PL/SQL-Code
- Anwendung:
  - Ranking
  - Aggregierung
  - Zeilenweise Verarbeitung
  - Statistik
  - „Was-wäre-wenn“-Szenarien

Anwendungsbeispiele

- „Running totals“
  - Zeilenweise Berechnung der kumulativen Umsätze, bei der jeder Datensatz die Summe der vorhergehenden Datensätze beinhaltet
- Anteil innerhalb einer Gruppe
  - Berechnung des Anteils eines Gehalts bezogen auf die Gesamtsumme innerhalb einer Abteilung
- Top-N-Abfragen
  - Finde die N umsatzstärksten Produkte pro Produktgruppe
- Gleitender Durchschnitt
  - Berechne den Durchschnittswert jeweils über die vorhergehenden N Datensätze
- Ranglisten

Syntax

- Analytic-Function (<Argument1>,<Argument2>, <Argument3>)
  OVER (
    <Query-Partition-Clause>
    <Order-By-Clause>
    <Windowing-Clause>
)

- OVER - identifiziert Analytic Function (Unterschied zu GROUP BY)

- PARTITION BY – aggregiert Ergebnismengen in Gruppen (Partitionen)

- ORDER BY – sortiert Daten innerhalb der Partitionen

- WINDOWING – logische Offsets innerhalb Teilergebnissen (Datensätze oder Bereiche)

Beispiel „Running Totals“

```
 SELECT ename, deptno, sal,
        (SELECT SUM (sal) FROM emp e1
          WHERE e1.deptno < emp.deptno
                OR (e1.deptno = emp.deptno AND e1.ename <= emp.ename)) running_total,
        (SELECT SUM (sal) FROM emp e2
          WHERE e2.deptno = emp.deptno AND e2.ename <= emp.ename) dept_total,
        (SELECT COUNT (ename) FROM emp e3
          WHERE e3.deptno = emp.deptno AND e3.ename <= emp.ename) seq
    FROM emp
ORDER BY deptno, ename
```

```
SELECT ename, deptno, sal,
        SUM (sal) OVER (ORDER BY deptno, ename) running_total,
        SUM (sal) OVER (PARTITION BY deptno ORDER BY ename) dept_total,
        ROW_NUMBER () OVER (PARTITION BY deptno ORDER BY ename) seq
    FROM emp
ORDER BY deptno, ename
```

Beispiel

Erweiterte Syntax

- <ROWS | RANGE> BETWEEN ... AND

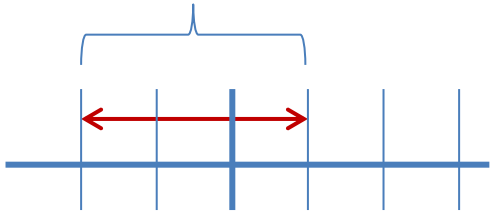
  UNBOUNDED PRECEDING – start of partition

  UNBOUNDED FOLLOWING – end of partition

  CURRENT ROW

  value_expr < PRECEDING| FOLLOWING>

- Beispiele
  - ROWS BETWEEN 1 PRECEDING and 1 FOLLOWING
  - ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  - RANGE BETWEEN INTERVAL '7' day PRECEDING AND CURRENT ROW
  - RANGE BETWEEN 10 PRECEDING AND 10 FOLLOWING



Standard-Aggregatfunktionen

- ROW_NUMBER()
- SUM(), MIN(), MAX(), AVG()
- LAG()
- LEAD()
- RANK()
- DENSE_RANK()
- PERCENT_RANK()
- NTILE()
- FIRST_VALUE()
- LAST_VALUE()
- FIRST()
- LAST()
- STATISTICAL FUNCTIONS

Gruppenbildung

- Partition By
- Aufteilung der Ergebnismenge in Gruppen
- Partitionierung ist beliebiger Ausdruck, jedoch keine Gruppenfunktion zulässig
- Einige Funktionen unterstützen Window-Klausel zur weiteren Begrenzung

```
SELECT employee_id, department_id,
COUNT (*) OVER (PARTITION BY department_id) DEPT_COUNT
FROM employees
```

Beispiel

Unterschied zu Group By

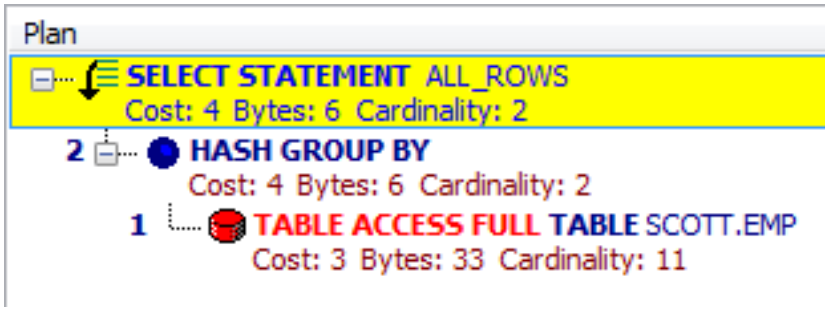
- GROUP BY
  - Spaltenliste darf nur gruppierte Spalten sowie Spalten mit Gruppenfunktion enthalten
  - Gruppierung erfolgt nur nach Spalten in Group By-Ausdruck

- Analytic Functions
  - Gruppenwerte können mehrfach erscheinen
  - Nutzung von Spalten ohne Gruppenfunktion möglich

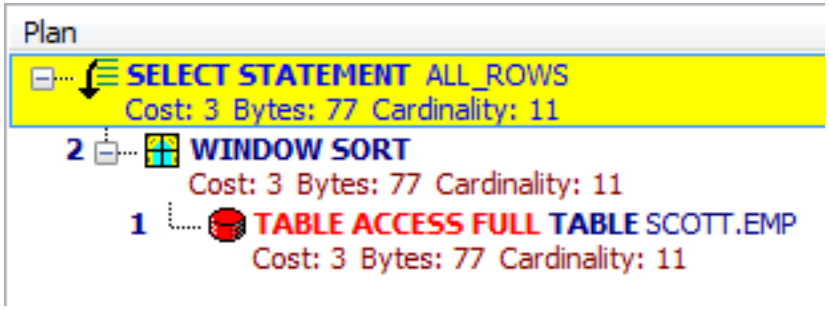
Execution Plans

Beispiel

```
SELECT deptno, COUNT(*) DEPT_COUNT
FROM emp
WHERE deptno IN (20, 30)
GROUP BY deptno;
```




```
SELECT deptno, empno,
COUNT(*) OVER (PARTITION BY deptno) CNT
FROM emp
WHERE deptno IN (20, 30)
ORDER BY 1;
```



Sortierung innerhalb Partitionen

- Sortierung innerhalb Partitionen **<>** Sortierung des Gesamtergebnisses

- (PARTITION BY …
  ORDER BY …)

- Sortierung der Partitionen kann Einfluss auf Ergebnis haben:

```
SELECT ename,
       deptno,
       sal,
       SUM (sal) OVER (ORDER BY ename, deptno) sum_ename_deptno,
       SUM (sal) OVER (ORDER BY deptno, ename) sum_deptno_ename
  FROM emp
ORDER BY ename, deptno
```

Beispiel

Tuning

- Läuft ein Vielfaches schneller als normales SQL
  - Andere Möglichkeiten als reines SQL
- Tuning wie üblich:
  - Indizierung
  - Partitioning
  - Parallelisierung

Beispiel Ranking

- RANK()
  - Evtl. mit Lücken
- DENSE_RANK()
  - Ohne Lücken

LAST_NAME	SALARY	NORMAL_RANK	DENSE_RANK
King	24000	1	1
Kochhar	17000	2	2
De Haan	17000	2	2
Russell	14000	4	3
Partners	13500	5	4
Hartstein	13000	6	5
Greenberg	12008	7	6
Higgins	12008	7	6
Errazuriz	12000	9	7
Ozer	11500	10	8

```
SELECT ename, sal,
        RANK () OVER (ORDER BY sal DESC) normal_rank,
        DENSE_RANK () OVER (ORDER BY sal DESC) DENSE_RANK
FROM emp;
```

Beispiel

essential BYTES

Beispiel LEAD und LAG

- LEAD: bestimmte vorhergehende Datensätze für die Berechnung verwenden
- LAG: bestimmte folgende Datensätze für die Berechnung verwenden

- LEAD (<sql_expr>, <offset>, <default>) OVER (<analytic_clause>)
- LAG (<sql_expr>, <offset>, <default>) OVER (<analytic_clause>)

```
SELECT deptno, empno, sal,
       LEAD (sal, 1, 0)
         OVER (PARTITION BY deptno ORDER BY sal DESC NULLS LAST)
         NEXT_LOWER_SAL,
       LAG (sal, 1, 0)
         OVER (PARTITION BY deptno ORDER BY sal DESC NULLS LAST)
         PREV_HIGHER_SAL
  FROM emp
 WHERE deptno IN (10, 20)
ORDER BY deptno, sal DESC;
```

Beispiel

essential BYTES

Weitere Beispiele

- Gleitender Durchschnitt
  - Über 3 Datensätze

```
SELECT satznummer, wert,
        avg(wert) OVER (ORDER BY satznummer
                        ROWS BETWEEN 1 PRECEDING
                        AND 1 FOLLOWING)
FROM tabelle
```

  - Über 3 Tage

```
SELECT tag, wert,
        avg(wert) OVER (ORDER BY BEGIN_INTERVAL_TIME
                        RANGE BETWEEN interval ‚1‘ day PRECEDING
                        AND ‚1‘ day FOLLOWING)
FROM tabelle
```

Weitere Beispiele

- Summenbildung, Totale und Subtotale, Anteil

```
SELECT DISTINCT prod_id,
                total, subtotal, ROUND (subtotal / total * 100, 2) anteil
   FROM (SELECT prod_id,
                SUM (quantity_sold) OVER () total,
                SUM (quantity_sold) OVER (PARTITION BY prod_id) subtotal
           FROM sales)
ORDER BY anteil DESC
```

Beispiel

Weitere Beispiele

- Daten auf verschiedene „Töpfe“ („buckets“) verteilen
  - Mit einer festgelegten Anzahl von Elementen pro Topf

```
SELECT prod_list_price, prod_name,
          CEIL (ROW_NUMBER () OVER (ORDER BY prod_list_price) / 4) grp
FROM product
```

- Mit einer bestimmten Anzahl von Töpfen

```
SELECT prod_list_price, prod_name,
          NTILE (10) OVER (ORDER BY prod_list_price) grp,
FROM products
```

Beispiel

Fragen ?

Kontakt

Essential Bytes GmbH & Co. KG

Markus Schmidt

Steinebühlstraße 30

77749 Hohberg

mschmidt@essential-bytes.de

http://www.essential-bytes.de