Grouping Sets und Pivotierung

GROUPING SET

Was ist das?

- Unterklausel, welche auf der GROUP BY Klausel aufbaut
- Ermöglicht die Definition mehrerer Gruppierungen in einer Abfrage
- Alternative zur Verwendung von UNION ALL, um mehrere Ausgaben (alle mit unterschiedlicher GROUP BY-Klausel) in einem Resultset zu kombinieren

```
SELECT <column ist with aggregate(s)>
FROM <source>
GROUP BY
GROUPING SETS(
        (<column_name>), -- eine oder mehrere Spalten
        (<column_name>), -- eine oder mehrere Spalten
        () – wenn alle Zeilen aggregiert werden → leere klammern
);
```

Abfragen

```
SELECT ca.CategoryName, c.CompanyName, SUM(od.Quantity) AS TotalQty
FROM Customers c
INNER JOIN Orders o ON c.CustomerID = o.CustomerID
INNER JOIN [Order Details] od ON o.OrderID = od.OrderID
INNER JOIN Products p ON od.ProductID = p.ProductID
INNER JOIN Categories ca ON ca.CategoryID = p.CategoryID
GROUP BY
GROUPING SETS ((ca.CategoryName),(c.CompanyName), ());
```

einmal gruppiert nach CategoryName

einmal gruppiert nach CompanyName

einmal ohne Gruppierungskriterien

Ergebnis

Ergebnisse | Meldungen

	CategoryName	CompanyName	TotalQty
85	NULL	Wartian Herkku	737
86	NULL	Wellington Importadora	267
87	NULL	White Clover Markets	1063
88	NULL	Wilman Kala	148
89	NULL	Wolski Zajazd	205
90	NULL	NULL	51317
91	Beverages	NULL	9532
92	Condiments	NULL	5298
93	Confections	NULL	7906
94	Dairy Products	NULL	9149

gruppiert nach CompanyName

ohne Gruppierungskriterien

gruppiert nach CategoryName

- **Eine** Tabelle nach mehreren Spalten gruppiert zusammengefasst

CUBE

- Erstellt alle möglichen Kombinationen von Gruppierungen auf Basis einer angegebenen Spaltenliste

```
SELECT <column_list_with_aggregate(s)>
FROM <source>
GROUP BY CUBE (<column_name>, <column_name>,...);
```

ROLLUP

- Erstellt hierarchiegebundene Kombinationen von GROUPING SETS. Grundlage der Hierarchie ist die Reihenfolge der eingegebenen Spaltenliste

```
SELECT <column_list_with_aggregate(s)>
FROM <source>
GROUP BY ROLLUP (<column_name>, <column_name>,...);
```

CUBE vs. ROLLUP

```
SELECT Category, Cust, SUM(qty)
FROM Sales.CategorySales
GROUP BY CUBE(Category, Cust);
```

- Erstellt folgende Groupingsets:

1. (Category, Cust)
2. (Category)
3. (Cust)
4. ()

```
SELECT Category, Cust, SUM(qty)
FROM Sales.CategorySales
GROUP BY ROLLUP(Category, Cust);
```

- Erstellt folgende Groupingsets:

1. (Category, Cust)
2. (Category)
3. ()

GROUPING

- Mehrere Groupingsets können bei der Ermittlung der einzelnen Zeilenquellen problematisch sein
  - NULL-Werte können aus den Quelldaten stammen oder ein Platzhalter im Groupingset sein
  - Die GROUPING-Funktion gibt eine 0 zurück, wenn die Spalte in der Gruppierung verwendet wurde und eine 1, wenn nicht

```
SELECT GROUPING(Category) AS grpCat, GROUPING(Cust) AS grpCust,
       Category, Cust, SUM(Qty) AS TotalQty
FROM Sales.CategorySales
GROUP BY CUBE(Category, Cust)
ORDER BY Category, Cust;
```

GROUPING_ID

- GROUPING_ID können mehrere Spalten übergeben werden
  - liefert eine 0 zurück, wenn alle Spalten in der Aggregation vorhanden sind
  - Wenn nicht alle Spalten in der Aggregation vorhanden sind, wird ein Bitwert zurückgegeben

```
GROUPING_ID(Spalte1, Spalte2, Spalte3, Spalte4) ...
```

2^3 2^2 2^1 2^0

- Für nicht benutzte Spalten wird das entsprechende Bit auf 1 gesetzt (von rechts ausgehend!)
- Beispiel: Spalte 1 und 4 gruppiert, Spalten 2 und 3 nicht → GROUPING_ID = 6
- Beispiel: Spalte 2 gruppiert, Spalten 1, 3 und 4 nicht → GROUPING_ID = 11

Pivotieren

- „Aus den Werten einer gruppierten Spalte können mehrere Spalten gemacht werden“
  - ➢ Werte aus einer Spalte können auf verschiedene neue Spalten verteilt werden
- Pivotieren umfasst drei Phasen:
  1. Die Gruppierung bestimmt, welches Element im Resultset eine Zeile erhält
  2. Beim Verteilen werden die einzelnen Werte pivotisiert
  3. Es wird eine Aggregation durchgeführt (z.B. SUM())

Beispiel

Gruppierungselement

```
SELECT name, [1996], [1997], [1998]
FROM ( SELECT ca.CategoryName, od.Quantity, YEAR(o.OrderDate) FROM Orders o
       INNER JOIN [Order Details] od ON o.OrderID = od.OrderID
       INNER JOIN Products p ON od.ProductID = p.ProductID
       INNER JOIN Categories ca ON ca.CategoryID = p.CategoryID) AS Tab(name, menge, jahr)
PIVOT(SUM(menge) FOR jahr IN([1996], [1997], [1998])) AS pvt;
```

Aggregation

Tabellierungselemente

Ergebnisse | Meldungen

	Kategorie	Jahr1	Jahr2	Jahr3
1	Beverages	1842	3996	3694
2	Condiments	962	2895	1441
3	Confections	1357	4137	2412
4	Dairy Products	2086	4374	2689
5	Grains/Cereals	549	2636	1377
6	Meat/Poultry	950	2189	1060

Best Practise

- In Pivot-Klausel nur das Aggregations- und die Tabellierungselemente angeben
  - ➢ Gruppierungselement sind die Spalten, die übrig bleiben
- Pivotierung sollte daher **nicht** auf der Originaltabelle durchgeführt werden
- Nur eine Aggregatfunktion erlaubt
- COUNT(*) ist nicht erlaubt
- IN-Klausel enthält statische Liste

UNPIVOT

- Beim Entpivotieren werden Spalten zu Zeilen
- Werte werden aus einer Quellzeile auf mehrere Zielzeilen verteilt

```
SELECT Kategorie, menge, jahr
FROM vw_pivot
UNPIVOT(menge FOR jahr IN ([1996], [1997], [1998])) AS Unpvt;
```

- Auch hier werden drei Dinge benötigt:
  1. Quellspalten, welche entpivotiert werden sollen
  2. Name der neuen Spalte mit Werten
  3. Name der neuen Spalte mit Namen

Ergebnisse | Meldungen

	Kategorie	menge	jahr
1	Beverages	1842	1996
2	Beverages	3996	1997
3	Beverages	3694	1998
4	Condiments	962	1996
5	Condiments	2895	1997
6	Condiments	1441	1998