

Group By, Having and multi-dimensional Grouping

Birgitta Hauser

Diplom-Betriebswirt (BA)
Software and Database Architect

Hauser@SSS-Software.de



Landsberg am Lech - Schmalzturm (1295)



Agenda

SELECT-Statement

GROUP BY Clause

- Aggregate Functions
 - Aggregate Functions handling distinct values
 - Aggregate Functions handling NULL values
 - Case Clause and Aggregate Functions
- LISTAGG Aggregate Function

HAVING Clause

Multi-dimensional Grouping

- Rollup and Cube Extension
- Grouping Sets
- Grouping Aggregate Function



SELECT Statement



SELECT Statement

WITH - Common Table Expressions (CTE)

Full Select

```
SELECT Columns/Fields (incl.scalar/UD functions)
From Files/Tables/Views/UDTFs incl. Joins
Where Conditions
Group By Clause
Having Clause
Start With Connect By (Release 7.1 PTF)
Order By Clause (Release 6.1)
Fetch First x Rows Only (Release 6.1)
```

Sub-Select

Merge several Sub-Selects using:

• UNION / EXCEPT / INTERSECT

```
SELECT Columns/Fields (incl.scalar/UD functions)
From Files/Tables/Views/UDTFs incl. Joins
Where Conditions
Group By Clause
Having Clause
Start With Connect By (Release 7.1 PTF)
Order By Clause (Release 6.1)
Fetch First x Rows Only (Release 6.1)
```

Sub-Select

ORDER BY Clauses

Fetch First x Rows, Limit/Offset, For Update Of, Optimize For x Rows

SELECT Statement



Group By



GROUP BY Clause

Used to arrange identical data into groups

- For multiple identical data is only a **single row** returned
- Often used in composition with **aggregate functions** for accumulating results

Positioned in a SELECT-Statement

- After the **WHERE** Condition
- After the **FROM** Clause (if there are no WHERE conditions)



GROUP BY Clause - Syntax

```
SELECT List Columns/Expression, AggregateFunctions
FROM Schema.Table or View
GROUP BY List Columns/Expressions
```

Condensing Data from multiple Rows

- **All Columns/Expressions without Aggregate Function (Grouping Expression) have to be repeated** after Group BY
- **Generated Names** for columns/Expr. in the Select List are **not allowed** → same Sub-Select

```
Select Year(SalesDate) SalesYear, CustNo,
Sum(Amount) Total, Count(*) Positions
From Sales
Group By Year(SalesDate), CustNo
Order By SalesYear, CustNo;
```

SALESYEAR	CUSTNO	TOTAL	POSITIONS
2008	10001	115,00	2
2008	10002	1350,00	1
2008	10003	535,00	3

• Grouping Expression must be **repeated** in the GROUP BY Clause

• Order By Clause Outside the Sub-Select → **Generated names** can be used



Aggregate Functions Before Release 7.3

Aggregate Functions	
Function Name	Description
AVG()	Average of a set of numbers
Count()	Number of rows/values in a set of rows/values
Count_Big()	Number of rows/values in a set of rows/values Similar to COUNT but the result can be greater than the maximum value of integer
Max()	Maximum value in a set of values in a group
Min()	Minimum value in a set of values in a group
Sum()	Sum of a set of numbers
StdDev()	biased standard deviation ($/n$) of a set of numbers
StdDev_Samp()	sample standard deviation ($/n-1$) of a set of numbers
Variance()	biased variance ($/n$) of a set of numbers
Variance_Samp()	sample variance ($/n-1$) of a set of numbers

17.06.2020

OMNI User Group 2020 – Group By, Having and Multi-dimensional Grouping - Birgitta Hauser

Page 10



Aggregate Functions New Release 7.3

Enhanced
RELEASE **7.3**

Aggregate Functions	
Function Name	Description
Correlation()	Coefficient of correlation of a set of number pairs
Covariance()	(Population) Covariance of a set of number pairs
Covariance_Samp()	Unbiased Sample Covariance ($n-1$) of a set of number pairs
Median()	Median of a set of numbers
Percentile_Cont()	Value that corresponds to the specified percentile given a sort specification by using a continuous distribution model
Percentile_Disc()	Value that corresponds to the specified percentile given a sort specification by using using a discrete distribution model.
Regression_Functions	
Regr_Count()	number of non-null number pairs used to fit the regression line
Regr_Intercept()	y-intercept of the regression line ("b" in the equation $y = a * x + b$)
Regr_R2()	coefficient of determination ("R-squared" or "goodness-of-fit") for the regression
Regr_Slope()	Slope of the line ("a" in the equation $y = a * x + b$)
Regr_AVGX()	can be used to compute various diagnostic statistics needed for the evaluation of the quality and statistical validity of the regression model
Regr_AVGY()	
Regr_SXX()	
Regr_SXY()	
Regr_SYY()	

17.06.2020

OMNI User Group 2020 – Group By, Having and Multi-dimensional Grouping - Birgitta Hauser

Page 11



GROUP BY Clause - Examples

```

Select CustNo, Count(*) "NbrRows", Sum(Amount) "Total" ,
       Cast(Avg(Amount) as Dec(11, 2)) "Average",
       Min(Amount) "Minimum", Max(Amount) "Maximum"
From Sales
Group By CustNo
Order By CustNO;

```

CUSTNO	NbrRows	Total	Average	Minimum	Maximum
10001	19	3031,14	159,53	20,00	450,85
10002	8	2986,25	373,28	20,00	1350,00
10003	18	6680,61	371,14	35,00	1555,75
10004	14	3143,95	224,56	45,00	539,75
10005	6	4051,95	675,32	225,00	1587,50
10006	22	19425,70	882,98	122,23	3368,75

Accumulating Sales per Customer

- **COUNT(*)** Counts the number of Rows per Customer
- **SUM()** Summarizes the **AMOUNT** of all Customer rows
- **AVG()** Calculates the Average of all Customer rows
CAST() formats the Float result returned by the **AVG()** function
- **MIN()** Returns the Minimum **AMOUNT** of the Customer
- **MAX()** Returns the Maximum **AMOUNT** of the Customer



GROUP BY Clause - Examples

```

Select Year(SalesDate) SalesYear, CustNo,
       Count(*) "NbrRows", Sum(Amount) "Total" ,
       Cast(Avg(Amount) as Dec(11, 2)) "Average",
       Min(Amount) "Minimum", Max(Amount) "Maximum"
From Sales
Where ItemNo Between '5100' and '5300'
Group By Year(SalesDate), CustNo
Order By CustNO, SalesYear;

```

SALESYEAR	CUSTNO	NbrRows	Total	Average	Minimum	Maximum
2008	10001	2	115,00	57,50	55,00	60,00
2009	10001	15	2634,20	175,61	20,00	450,85
2010	10001	2	281,94	140,97	140,97	140,97
2008	10002	1	1350,00	1350,00	1350,00	1350,00
2009	10002	3	489,90	163,30	20,00	444,50
2008	10003	1	350,00	350,00	350,00	350,00
2009	10003	3	853,10	284,36	180,00	444,50
2009	10004	8	1963,55	245,44	150,00	431,80
2008	10005	1	310,00	310,00	310,00	310,00
2009	10005	3	904,45	301,48	225,00	393,70
2010	10006	4	2247,30	561,82	495,00	628,65

Grouping Expression

- **GROUP BY** and **ORDER BY** **different** sequence

- **WHERE** Condition – **Column** not included in the Grouping Expression



Aggregate Functions and Distinct Values



Distinct

Eliminates duplicate rows of the final result table

- **DISTINCT** must be specified immediately **after SELECT**

Example: Determine all customers from the Order Header table with at least one order that is not yet delivered. Using **DISTINCT** each customer is only returned **once**.

```
Select Distinct CustNo
From OrderHdrx
Where DelDate between '2009-01-01'
and '2009-12-31';
```

CUSTNO
20004
10003
10002
20005

```
Select CustNo
From OrderHdrx
Where DelDate between '2009-01-01'
and '2009-12-31';
```

CUSTNO
20004
20004
20004
20005
20005
10003
10003
10003
10002
10002

• Without **DISTINCT**
→ Duplicate Customer No



Aggregate Functions with DISTINCT

Distinct can be specified within **Aggregate Functions**

- Duplicates are only considered **once**

Example: Determine the number of different customers from the Order Header Table with at least one order within a specific period

```
Select Distinct CustNo  
From OrderHdrx  
Where DelDate between '2009-01-01'  
and '2009-12-31';
```

CUSTNO
20004
10003
10002
20005



```
Select Count(Distinct CustNo) "Nbr Customer"  
From OrderHdrx;;
```

Nbr Customer
4



Aggregate Functions with DISTINCT over multiple Columns - Example

- Determine the number of Customer No/Item No Pairs (Total 16 Pairs)

```
Select Distinct CustNo, ItemNo  
From Sales  
Order By CustNo, ItemNo;
```

CUSTNO	ITEMNO
10001	5100
10001	5200
10001	5300
10002	5100
10002	5200
10002	5400
10003	5200
10003	5400
10004	5200
10004	5300
10004	5400
10005	5200
10005	5300
10005	5400
10006	5200
10006	5400

- **Listing multiple Columns** within the COUNT() Function returns an **Error**

```
Select Count(Distinct CustNo, ItemNo)  
From Sales;
```

```
[ Mon Aug 14 12:03:05 CEST 2017 ] Run Selected...  
Select Count(Distinct CustNo, ItemNo) From Sales  
* SQL State: 42605  
Vendor Code: -170  
Message: [SQL0170] Number of arguments for function COUNT not valid.
```

- **Concatenating multiple columns** in the COUNT() Function returns the **expected result**

```
Select Count(Distinct CustNo concat ItemNo) "NbrCustItem"  
From Sales;
```

NbrCustItem
16



Aggregate Functions with DISTINCT - Examples

```
Select Year(SalesDate) SalesYear,  
       Count(*) "Nbr Positions",  
       Count(Distinct CustNo) "Nbr Customer",  
       Count(Distinct ItemNo) "Nbr Items",  
       Count(Distinct CustNo concat ItemNo) "Nbr Customer/Items"  
From Sales  
Group By Year(SalesDate)  
Order By SalesYear
```

SALESYEAR	Nbr Positions	Nbr Customer	Nbr Items	Nbr Customer/Items
2008	9	5	3	6
2009	53	5	4	14
2010	25	3	3	4



NULL Values



NULL Predicate

What are NULL Values?

- Values **out of** the **Valid Range**
 - Neither Blank nor Zero nor x'00'
 - Selecting everything between high and low value → will **NOT** return NULL values

NULL Value in the Database

- **Separate** flag set to *On/*Off → Must be checked **separately**
 - **SQL:** using the **NULL predicate** `IS NULL / IS NOT NULL`
 - **RPG:** Built-In-Function: `%NullInd()`
 - Embedded SQL: **Indicator Variable**

```
Select *  
From OrderHdrX  
Where DelDate > Current Date  
or DelDate is NULL
```



Convert Default Values into NULL Values

`COALESCE(Column, Default1, Default2, ... DefaultN) or IFNULL(Column, Default)`

Converts a NULL Value into a **Default Value**

- Can be used for all data types
- **COALESCE** is **more powerful** than **IFNULL** → Multiple Default Values can be specified

```
Select h.Company, h.OrderNo, h.DelDate,  
Coalesce(ItemNo, 'No Order Position') Item,  
Coalesce(OrderQty, 0) OrderQty  
from OrderHdrX h left outer join OrderDetX d  
on h.Company = d.Company and h.OrderNo = d.OrderNo
```

COMPANY	ORDERNO	DELDATE	ITEM	ORDERQTY
10	BNR2009-10-15/1	2009-08-20	CF001	10
10	BNR2009-10-15/1	2009-08-20	CF003	15
10	BNR2009-10-15/1	2009-08-20	HG001	3
10	BNR2009-10-15/1	2009-08-20	BS002	20
10	BNR2009-10-23/5	2009-08-28	No Order Position	0
10	BNR2009-10-30/2	2009-09-09	BS001	50

• Convert a Default value into a NULL value



Convert NULL Values into Default Values

`NULLIF(Column, ToReplace)`

Converts a **specified Value** into a NULL Value

- Can be used for all data types

```
Select Company, OrderNo,  
       Count(*) "TotalPos", Count(NULLIf(DelQty, 0)) "PosWithDelQty",  
       Avg(DelQty) "AvgDelQty", Avg(NULLIf(DelQty, 0)) "AvgDelQty > 0"  
From OrderDetX  
Group By Company, OrderNo
```

• Convert a specified value into a NULL value

COMPANY	ORDERNO	TotalPos	PosWithDelQty	AvgDelQty	AvgDelQty > 0
10 BNR2009-12-20/2		6	5	6	7
10 BNR2009-12-23/7		3	0	0	-
10 BNR2009-10-15/2		5	5	36	36
10 BNR2009-10-15/1		4	3	10	14

• As an aside: NULL Values in Aggregate Functions **are ignored!**

17.06.2020

OMNI User Group 2020 – Group By, Having and Multi-dimensional Grouping - Birgitta Hauser

Page 23



NULL Values and Aggregate Functions

17.06.2020

OMNI User Group 2020 – Group By, Having and Multi-dimensional Grouping - Birgitta Hauser

Page 24



NULL Values and Aggregat Functions

NULL values are ignored by aggregat functions

- Important for COUNT(), AVG(), MIN(), MAX()
- Example: 6 rows with 2 NULL values → Count(Column) = 4
- COUNT(*) Counts ALL rows even if there is a row with only NULLS

```

Select Company, OrderNo,
       Count(*) "TotlaPos", Count(NULLIF(DeQty, 0)) "PoswithDeQty",
       Avg(DeQty) "AvgDeQty", Avg(NULLIF(DeQty, 0)) "AvgDeQty > 0"
From OrderDetx
Group By Company, OrderNo
    
```

COMPANY	ORDERNO	TotlaPos	PoswithDeQty	AvgDeQty	AvgDeQty > 0
10BNR2009-12-20/2		6	5	6	7
10BNR2009-12-23/7		3	0	0	-
10BNR2009-10-15/2		5	5	36	36
10BNR2009-10-15/1		4	3	10	14
10BNR2009-12-15/1		4	3	21	28

- Positions without delivery quantity are considered
- Positions without delivery quantity are ignored



NULL Values in Aggregate Functions are ignored - Example

```

Select *
From NULLFile;
    
```

MYINT	MYCHAR	MYDEC	MYDATE	MYGRAPH2
1	-	-	-	-
2	AAA	-	-	-
1	BBB	-	-	BBB
1	-	123,00	-	-
2	-	456,00	-	-
2	AAA	777,00	-	-
3	CCC	444,00	-	CCC
1	-	-	2007-02-21	-
1	XXX	-	2007-01-31	XXX
2	-	1000,00	2007-05-15	BBB
2	AAA	-	2007-05-15	-
3	AAA	-	2007-01-07	AAA
2	AAA	2000,00	2007-01-07	BBB
2	XXX	900,00	2007-01-07	BBB
2	AAA	700,00	-	-
2	AAA	300,00	-	-

```

Select Count(*) "All",
       Count(MyInt) "Count MyInt", Count(MyDec) "Count MyDec",
       Count(MyChar) "Count MyChar", Count(MyGraph2) "Count MyGraph2"
From NULLFile;
    
```

All	Count MyInt	Count MyDec	Count MyChar	Count MyGraph2
16	16	9	11	7



Having

17.06.2020

OMNI User Group 2020 – Group By, Having and Multi-dimensional Grouping - Birgitta Hauser

Page 28



Having Clause

```
SELECT    List Columns/Expression, AggregateFunctions
FROM      Schema.Table or View
GROUP BY  List Columns/Expression
HAVING    Where Conditions for AggregateFunctions
```

Having = WHERE Conditions for Aggregate Functions

- WHERE conditions only allowed for **Column Values** but **NOT** for **aggregated Values**
- Examples: Select customers with a total amount > x Euro
 Select all orders with more than y positions

17.06.2020

OMNI User Group 2020 – Group By, Having and Multi-dimensional Grouping - Birgitta Hauser

Page 29



Having Clause

```
SELECT    List Columns/Expression, AggregateFunctions
FROM      Schema.Table or View
GROUP BY  List Columns/Expression
HAVING    Where Conditions for AggregateFunctions
```

Compare the result of aggregate functions

- Comparison Operators

- = (Equal) > (Greater than) >= (Greater equal)
- <> (Not equal) < (Lower than) <= (Lower equal)

- Predicates

- **IN** (List values) **BETWEEN** (Range)



Having Clause

```
SELECT    List Columns/Expression, AggregateFunctions
FROM      Schema.Table or View
GROUP BY  List Columns/Expression
HAVING    Where Conditions for AggregateFunctions
```

Multiple Conditions can be specified

- Linked through **logical Operators**

- **AND** = Logical AND **OR** = Logical OR **NOT** = Logical NOT

Parenthesis can be set



Having Conditions Examples

```
Select Company, OrderNo, Count(*) Positions  
from OrderDetx  
Group By Company, OrderNo  
Having Count(*) > 4;
```

- Determine all orders with more than 4 positions

COMPANY	ORDERNO	POSITIONS
10	BNR2009-12-20/2	6
10	BNR2009-10-15/2	5
10	BNR2009-12-15/2	5
10	BNR2009-10-20/2	6

```
Select Year(SalesDate) SalesYear, CustNo, Sum(Amount) Total  
From Sales  
Group By Year(SalesDate), CustNo  
Having Sum(Amount) > 3000  
and Count(*) > 10;
```

- Determine all Customer with an annual amount > 3000 and more than 10 Positions

SALESYEAR	CUSTNO	TOTAL
2009	10003	4589,86
2010	10006	19425,70



Multi-dimensional Grouping



Multi-dimensional Grouping

New Extensions for the Group By Clause:

- `Cube(Grouping Columns/Fields)`
- `RollUp(Grouping Columns/Fields)`
- `Grouping Sets(Grouping Columns/Fields)`

Cube and Rollup can be specified within the Grouping Sets()



Multi-dimensional Grouping - RollUp() Extention

RollUp() Extention

- Generate **Sub-Totals** based on the columns specified with `RollUP()`

Example: `Group By RollUp (Year, Month, Day)`

- Generates the **following Sub-Totals:**
 - Sub-Total per Year, Month, Day
 - Sub-Total per Year, Month,
 - Sub-Total per Year
 - Grand Total
- **Sequence** of the result set depends on the **ORDER BY** Clause



Multi-dimensional Grouping RollUp() Extention - Example

```
Select Year(SalesDate) SalesYear, CustNo, Sum(Amount) Total
From Sales
Group By RollUp(Year(SalesDate), CustNo)
Order By SalesYear, CustNo
```

SALESYEAR	CUSTNO	TOTAL
2008	10001	115,00
2008	10002	1350,00
2008	10003	535,00
2008	10004	470,00
2008	10005	310,00
2008	-	2780,00
2009	10001	2634,20
2009	10002	1636,25
2009	10003	4589,86
2009	10004	2673,95
2009	10005	3741,95
2009	-	15276,21
2010	10001	281,94
2010	10003	1555,75
2010	10006	19425,70
2010	-	21263,39
--	--	39319,60

• Sales per Year and Customer

• Sub-Totals:
 ▪ Year/Customer
 ▪ Year

• Grand Total



Mult-dimensional Grouping Cube Extention

Extention Cube

- Generates **Sub-Totals** for **each possible composition** of columns specified in **Group By Clause**
 → All Sub-Totals created by RollUp() + “cross tabulation”

Example: **Group By Cube(Year, Customer, Item)**

- Generates the following Sub-Totals:
 - Sub-Total per Year, Customer, Item
 - Sub-Total per Year, Customer
 - Sub-Total per Year, Item
 - Sub-Total per Customer, Item
 - Sub-Total per Year
 - Sub-Total per Customer
 - Sub-Total per Item
 - Grand-Total
- Sequence** of the Result Set depends on the **ORDER BY** Clause



Multi-dimensional Grouping Cube() versus RollUp() - Examples

```

select Year(SalesDate) SalesYear, CustNo, ItemNo,
       Sum(Amount) Total
From Sales
where SalesDate between '2008-10-01'
       and '2009-03-31'
       and CustNo in ('10001', '10002')
Group By Cube(Year(SalesDate), CustNo, Item)
Order By SalesYear, CustNo, Item

```

SALESYEAR	CUSTNO	ITEMNO	TOTAL
2008	10001	King, Stephen - Es	115,00
2008	10001	-	115,00
2008	10002	King, Stephen - Es	1350,00
2008	10002	-	1350,00
2008	-	King, Stephen - Es	1465,00
2008	-	-	1465,00
2009	10001	Grisham, John - Die Akte	180,00
2009	10001	King, Stephen - Drei	160,00
2009	10001	-	340,00
2009	-	Grisham, John - Die Akte	180,00
2009	-	King, Stephen - Drei	160,00
2009	-	-	340,00
-	10001	Grisham, John - Die Akte	180,00
-	10001	King, Stephen - Drei	160,00
-	10001	King, Stephen - Es	115,00
-	10001	-	455,00
-	10002	King, Stephen - Es	1350,00
-	10002	-	1350,00
-	-	Grisham, John - Die Akte	180,00
-	-	King, Stephen - Drei	160,00
-	-	King, Stephen - Es	1465,00
-	-	-	1805,00

```

select Year(SalesDate) SalesYear, CustNo, Item,
       Sum(Amount) Total
From Sales
where SalesDate between '2008-10-01'
       and '2009-03-31'
       and CustNo in ('10001', '10002')
Group By RollUp(Year(SalesDate), CustNo, Item)
Order By SalesYear, CustNo, Item

```

SALESYEAR	CUSTNO	ITEM	TOTAL
2008	10001	King, Stephen - Es	115,00
2008	10001	-	115,00
2008	10002	King, Stephen - Es	1350,00
2008	10002	-	1350,00
2008	-	-	1465,00
2009	10001	Grisham, John - Die Akte	180,00
2009	10001	King, Stephen - Drei	160,00
2009	10001	-	340,00
2009	-	-	340,00
-	-	-	1805,00

• Cross Tabulation



Multi-dimensional Grouping – Grouping Sets () Extention

Grouping Sets() Extention

- Define **any** grouping level/sub-totals you want

Example: `Grouping Sets((A, B, C), (B, C), ())`

- Generates the following sub-totals:
 - Sub-Total per A, B, C
 - Sub-Total per B, C
 - Grand Total
- **Sequence** of the Result Set depends on the **ORDER BY** Clause



Multi-dimensional Grouping Grouping Sets() - Example

```
select Year(SalesDate) SalesYear, CustNo, Item, Sum(Amount) Total
From Sales
Where CustNo between 10001 and 10003
Group By Grouping Sets((Year(SalesDate), CustNo),
                       (Year(SalesDate), Item),
                       (Year(SalesDate)), ())
Order By 1, 3, 4, 2
```

SALESYEAR	CUSTNO	ITEM	TOTAL
2008	-	King,Stephen - Drei	350,00
2008	-	King,Stephen - Es	1465,00
2008	-	King,Stephen - Shining	185,00
2008	10001	-	115,00
2008	10003	-	535,00
2008	10002	-	1350,00
2008	-	-	2000,00
2009	-	Grisham,John - Die Akte	896,65
2009	-	King,Stephen - Drei	1261,70
2009	-	King,Stephen - Es	1818,85
2009	-	King,Stephen - Shining	4883,11
2009	10002	-	1636,25
2009	10001	-	2634,20
2009	10003	-	4589,86
2009	-	-	8860,31
2010	-	King,Stephen - Es	281,94
2010	-	King,Stephen - Shining	1555,75
2010	10001	-	281,94
2010	10003	-	1555,75
2010	-	-	1837,69
--	--	--	12698,00

• Sub-Total SalesYear/Item

• Sub-Total Year/Customer

• Sub-Total SalesYear

• Grand Total



Multi-Dimensional Grouping - Aggregat-Function Grouping

Grouping(ColumnName)

Aggregate-Function: Grouping(Column)

- Can only be used in composition with **multi-dimensional Grouping**

Identifies NULL Values in Sub-Total Rows

- **1** = NULL value in the specified column in the sub-total row
- **0** = No NULL value in the specified column



Multi-dimensional Grouping - Aggregate Function Grouping - Example

```

With x as (Select Year(SalesDate) SalesYear, CustNo, Amount
           from Sales)
Select Case When Grouping(SalesYear) = 1
         Then 'Grand Total'
         When Grouping(CustNo) = 1
         Then 'Total Year ' concat Varchar(SalesYear)
         Else '' End,
       SalesYear, CustNo, Sum(Amount) as Total
from x
Group By Rollup(SalesYear, CustNo)
Order By SalesYear, CustNo
    
```

- Generate Summary Textes based on the **GROUPING** Aggregate Function

00001	SALESYEAR	CUSTNO	TOTAL
	2008	10001	115,00
	2008	10002	1350,00
	2008	10003	535,00
	2008	10004	470,00
	2008	10005	310,00
Total Year 2008	2008 -		2780,00
	2009	10001	2634,20
	2009	10002	1636,25
	2009	10003	4389,86
	2009	10004	2673,95
	2009	10005	3741,95
Total Year 2009	2009 -		15276,21
	2010	10001	281,94
	2010	10003	1555,75
	2010	10006	19425,70
Total Year 2010	2010 -		21263,39
Grand Total	--		39319,60



Multi-dimensional Grouping - Aggregate Function Grouping - Example

- Display Sub-Totals and Grand Total only

```

With x as (Select Year(SalesDate) SalesYear, CustNo, Amount
           from Sales)
Select Case When Grouping(SalesYear) = 1
         Then 'Grand Total'
         When Grouping(CustNo) = 1
         Then 'Total Year ' concat Varchar(SalesYear)
         Else '' End,
       SalesYear, CustNo, Sum(Amount) as Total
from x
Group By Rollup(SalesYear, CustNo)
Having Grouping(CustNo) = 1
Order By SalesYear, CustNo
    
```

- Based on a **Having** Condition in composition with the **Grouping** Aggregate Function

00001	SALESYEAR	CUSTNO	TOTAL
Total Year 2008	2008 -		2780,00
Total Year 2009	2009 -		15276,21
Total Year 2010	2010 -		21263,39
Grand Total	--		39319,60



Any Questions?



Landsberg am Lech



Special Thanks to

Holger Scherer – RZKH Rechenzentrum Kreuznach

- For providing an IBM i-System enabling the creation of the samples/code used in my presentations
- <http://www.rzkh.de>



■ Your data is save! ... in the bunker



References

IBM i Information Center

- DB2 for i SQL Reference
http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_74/rzajp/rzajppdf.pdf?lang=en
- Embedded SQL programming
http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_74/db2/rbafzpdf.pdf?lang=en
- RPG Reference
https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_74/rzasd/sc092508.pdf?view=kc

IBM Redbooks

- Who Knew You Could Do That with RPG IV? – Modern RPG for the Modern Programmer
<http://www.redbooks.ibm.com/abstracts/sg245402.html?Open>
- Modernizing IBM eServer iSeries Application Data Access – A Roadmap Cornerstone
<http://www.redbooks.ibm.com/abstracts/sg246393.html?Open>
- Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between
<http://www.redbooks.ibm.com/abstracts/sg248185.html?Open>



Speaker's Biography

Birgitta Hauser
Diplom-Betriebswirt (BA)
Database and Software Architect

Birgitta Hauser worked on the IBM i and its predecessors since 1992. She graduated with a business economics diploma, and started programming on the AS/400 in 1992. She worked and works as traditional RPG Programmer but also as Database and Software Engineer, focusing on IBM i application and database modernization.

Currently she is self-employed and works in Consulting and Application and Database Modernization on IBM i and Db2 for i. Since July, 2019 she is occasionally working for Fresche Solutions Inc. (Montréal) as a contractor.

She also works in education as a trainer for RPG and SQL developers.

Since 2002 she has frequently spoken at the COMMON User Groups and other IBM i and Power Conferences in Germany, other European Countries, USA and Canada.

In addition, she is co-author of two IBM Redbooks and also the author of several articles and papers focusing on RPG and SQL for the ITP Verlag (a German publisher), IT Jungle Guru and IBM DeveloperWorks.

In 2015 she received the John Earl Speaker Scholarship Award. In 2018 she received the Al Barsa Memorial Scholarship Award.



IBM Champion 2020

17.06.2020

OMNI User Group 2020 – Group By, Having and Multi-dimensional Grouping - Birgitta Hauser

Page 59



Thank you

Group By, Having and multi-dimensional Grouping?

Yes I can!

If you are interested in more detailed individual Workshops on-site or remote,
Please contact me directly

Birgitta Hauser
Diplom-Betriebswirt (BA)
Database and Software Architect
Hauser@SSS-Software.de

