

Do you know – What you can do with a Simple SQL Statement?



Birgitta Hauser

TOOLMAKER Advanced Efficiency GmbH

bha@toolmaker.de / Hauser@SSS-Software.de

Tel. (+49) 08191 968-0

www.toolmaker.de

Agenda

SELECT – Statement

CASE Clause

OLAP Ranking Functions

Join Tables and Views

Union, Intersect and Except

Common Table Expressions

- Recursive Common Table Expressions
- Hierarchical Query Clause

Multi-Dimensional Grouping



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 2

SELECT Statement



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 3

SELECT Statement

WITH – Common Table Expressions (CTE)

Full Select

```
SELECT Columns/Fields (incl.scalar/UD functions)
From Files/Tables/Views incl. Joins and UDTFs
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 incl. Joins and UDTFs
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, For Update Of, Optimize For x Rows



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 4

Case Clause



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 6

CASE Expression

Allows to act in **different** situations **differently**

■ Kind of IF-THEN-ELSE-Statement

```
CASE Condition
  WHEN Value1 THEN ThisValue
  WHEN Value2 THEN ThatValue
  ELSE OtherValue
END
```

```
Select Case Month(SalesDate)
  when 1 Then 'January'
  when 2 Then 'February'
  Else 'Other Month'
End
From Sales
```

ELSE-Branch optional

```
CASE WHEN Condition1 THEN ThisValue
      WHEN Condition2 THEN ThatValue
      ELSE OtherValue
END
```

```
Select Case when Amount < 100 Then 'Low'
              when Amount < 500 Then 'Avg'
              when Amount >= 500 Then 'High'
End
From Sales
```



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 7

CASE Expression

Display different values depending on a column value

```
Select Company, OrderNo, OrderPos, ItemNo, OrderQty, DelQty, Status,  
       case Status when 'CP' then 'Completed'  
                when 'CL' then 'Cancelled'  
                when 'EN' then 'Entered'  
                when 'PD' then 'Partly Delivered'  
                else 'Unknown Status' end as StsText  
from OrderDetX
```

Merge Values from different columns

```
Select Company, OrderNO, OrderPos, ItemNo, OrderQty, DelQty, Status,  
       case when Status = 'CP' then DelQty  
            when Status = 'CL' then OrderQty  
            when Status = 'EN' then OrderQty  
            when Status = 'PD' then DelQty  
            else 0 end as Quantity  
from OrderDetX  
;;
```



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 8

CASE Expression

Case Expression used in the WHERE Clause

```
Select *  
  from OrderHdrx  
 where OrderType = case When CustNo >= '20000' then 'DO'  
                        When CustNo = '10002' then 'UO'  
                        Else OrderType  
                    End
```

Order By CustNo

COMPANY	ORDERNO	CUSTNO	ORDERTYPE	DELDATE	DELTERMS	STATUS
10	BNR2009-12-15/2	10002	UO	12/04/2009	EXW	EN
10	BNR2009-12-20/2	10002	UO	12/15/2009	EXW	EN
10	BNR2009-12-15/1	10003	DO	12/01/2009	CPT	EN
10	BNR2009-12-23/5	10003	EX	12/09/2009	CPT	EN
10	BNR2009-12-30/2	10003	EX	12/21/2009	CPT	EN
10	BNR2009-10-15/1	20004	DO	08/20/2009	CPT	EN
10	BNR2009-10-23/5	20004	DO	08/28/2009	CPT	EN
10	BNR2009-10-30/2	20004	DO	09/09/2009	CPT	EN
10	BNR2009-10-20/2	20005	DO	09/03/2009	EXW	EN

COM...	ORDERNO	CUSTNO	ORDERTYPE	DELDATE	DELTERMS	STATUS
10	BNR2009-10-15/1	20004	DO	09/20/2009	CPT	EN
10	BNR2009-10-23/5	20004	DO	08/28/2009	CPT	EN
10	BNR2009-10-30/2	20004	DO	09/09/2009	CPT	EN
10	BNR2009-10-15/2	20005	EX	09/23/2009	EXW	EN
10	BNR2009-10-20/2	20005	DO	09/03/2009	EXW	EN
10	BNR2009-12-15/1	10003	DO	12/01/2009	CPT	EN
10	BNR2009-12-23/5	10003	EX	12/09/2009	CPT	EN
10	BNR2009-12-30/2	10003	EX	12/21/2009	CPT	EN
10	BNR2009-12-15/2	10002	UO	12/04/2009	EXW	EN
10	BNR2009-12-20/2	10002	UO	12/15/2009	EXW	EN
10	TRG2010-09-07/1	10002	DO	09/10/2010	EXW	EN



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 9

Case Expression

```

Select OrderQty, DelQty, Status,
  case when Status = 'CP' then DelQty
        when Status = 'CL' then OrderQty
        when Status = 'EN' then OrderQty
        when Status = 'PD' then DelQty
        else 0 end as Quantity,
  case when Status in ('CL', 'EN') then ' is Ordered Quantity'
        when Status in ('CP', 'PD') then
          Case When OrderQty = DelQty then ' is completely delivered'
                When OrderQty > DelQty then ' is partly delivered'
                Else 'is over delivered' End
        else '' end as Description,
a.*
from OrderDetX a

```

ORDERQTY	DELQTY	STATUS	QUANTITY	DESCRIPTION	COMPANY	ORDERNO
10	10	CP	10	is completely delivered	10	BNR2009-10-15/1
15	12	CP	12	is partly delivered	10	BNR2009-10-15/1
3	0	CL	3	is Ordered Quantity	10	BNR2009-10-15/1
20	20	CP	20	is completely delivered	10	BNR2009-10-15/1
50	0	EN	50	is Ordered Quantity	10	BNR2009-10-30/2
30	0	EN	30	is Ordered Quantity	10	BNR2009-10-30/2
15	0	EN	15	is Ordered Quantity	10	BNR2009-10-30/2
50	45	PD	45	is partly delivered	10	BNR2009-10-15/2
75	75	CP	75	is completely delivered	10	BNR2009-10-15/2



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 10

CASE Expression

Pivot-Table

```

Select CustNo, Year(SalesDate) SalesYear,
  sum(case when Month(SalesDate)= 1 then Amount else 0 end) as Jan,
  sum(case when Month(SalesDate)= 2 then Amount else 0 end) as Feb,
  sum(case when Month(SalesDate)= 3 then Amount else 0 end) as Mar,
  sum(case when Month(SalesDate)= 4 then Amount else 0 end) as Apr,
  sum(case when Month(SalesDate)= 5 then Amount else 0 end) as May,
  sum(case when Month(SalesDate)= 6 then Amount else 0 end) as Jun,
  sum(case when Month(SalesDate)= 7 then Amount else 0 end) as Jul,
  sum(case when Month(SalesDate)= 8 then Amount else 0 end) as Aug,
  sum(case when Month(SalesDate)= 9 then Amount else 0 end) as Sep,
  sum(case when Month(SalesDate)= 10 then Amount else 0 end) as Oct,
  sum(case when Month(SalesDate)= 11 then Amount else 0 end) as Nov,
  sum(case when Month(SalesDate)= 12 then Amount else 0 end) as Dec,
  sum(Amount) as Total
from Sales
group by CustNo, Year(SalesDate)
order by CustNo, 2

```

CUSTNO	SALESYEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	TOTAL
10001	2008	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	55,00	60,00	115,00
10001	2009	160,00	0,00	180,00	299,40	0,00	0,00	240,40	814,60	63,50	425,45	450,85	0,00	2634,20
10001	2010	140,97	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	140,97
10002	2008	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
10002	2009	0,00	0,00	0,00	0,00	0,00	20,00	250,00	280,40	0,00	762,00	323,85	0,00	1636,25
10002	2010	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	35,00	500,00	535,00
10003	2009	333,00	200,00	200,00	1057,91	0,00	0,00	254,00	1749,00	224,45	0,00	571,50	0,00	4569,86
10003	2010	1555,75	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1555,75
10004	2008	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	45,00	425,00	470,00
10004	2009	340,00	0,00	50,00	1121,55	0,00	155,00	220,00	450,85	57,15	0,00	279,40	0,00	2673,95
10005	2008	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	310,00	0,00	310,00
10005	2009	0,00	225,00	1250,00	0,00	0,00	285,75	0,00	1587,50	0,00	393,70	0,00	0,00	3741,95
10006	2010	915,75	550,00	550,00	0,00	0,00	0,00	698,50	4809,75	617,23	0,00	1571,62	0,00	9712,85



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 11

OLAP-Ranking Functions



OLAP - Ranking Functions

OnLine Analytical Processing (OLAP) Functions

Generate Running Numbers or Rank Orders

Can not be used within the same Sub-Select

- **Allowed** within the **SELECT** List and **ORDER BY** Clause
- **Not allowed** within the **WHERE** Conditions or **FROM** Clause
- Select 3rd – 5th rank is only possible in composition with a Common Table Expression (CTE) or Nested Sub-Select

Only executed by the SQE (SQL Query Engine)

- **Can not** be used if the query is executed by the **CQE** (Classic QE)



OLAP - Ranking Functions



ROW_NUMBER() OVER() – Generate Running Numbers

- No Duplicate → a **sequential** row number is computed

RANK() and DENSE_RANK() – Determine Ranks

- Computes an ordinal rank: **Identical** values → **same** ranking
- RANK() OVER() after identical ranks → **Gaps** within the ranking
 - Example: Two ranked 3rd → next rank is 5, **not** 4th rank
- DENSE_RANK() OVER() after identical ranks → **No Gaps** within the ranking
 - Example: Two ranked 3rd → next rank is 4



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 14

OLAP Ranking Functions



```
Select Row_Number() Over()  
      CustNo, Amount  
from SalesCustY  
Where SalesYear = 2008
```



ROWNR	CUSTNO	AMOUNT
1	10004	470,00
2	10003	535,00
3	10005	310,00
4	10002	1350,00
5	10001	115,00

- Sequence of the running number depends on the **ORDER BY** Clause

```
Select Row_Number() Over()  
      CustNo, Amount  
from SalesCustY  
Where SalesYear = 2008  
Order By CustNo
```



ROWNR	CUSTNO	AMOUNT
1	10001	115,00
2	10002	1350,00
3	10003	535,00
4	10004	470,00
5	10005	310,00



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 15

OLAP - Ranking Functions

Window-Order-Clause



Window-Order-Clause - OVER(ORDER BY Clause)

- Defines the **sequence** of the running number or ranks which may differ from the sequence specified in the ORDER BY Clause
- Only allowed in **composition** with the **OLAP Ranking functions**
 - ROW_NUMBER(), RANK(), DENSE_RANK()
 - Must be specified for each **OLAP function**, immediately after the function
- Syntax **analog** ORDER-BY clause
 - **Several key fields** can be listed
 - **Ascending** and **Descending** Sequence allowed (ASC / DESC)



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 16

OLAP-Ranking Functions

Examples



```
Select Row_Number() over(Order By Amount Desc) RowNbr,  
       CustNo, Amount  
From SalesCustY  
Where SalesYear = 2008  
Order By Amount Desc  
;
```

ROWNBR	CUSTNO	AMOUNT
1	10002	1350,00
2	10003	535,00
3	10004	470,00
4	10005	310,00
5	10001	115,00

- **Generating running numbers based on the OVER(ORDER BY) Clause**

```
Select Row_Number() over(Order By Amount Desc) RowNbr,  
       CustNo, Amount  
From SalesCustY  
Where SalesYear = 2008  
Order By CustNo  
;
```

ROWNBR	CUSTNO	AMOUNT
5	10001	115,00
1	10002	1350,00
2	10003	535,00
3	10004	470,00
4	10005	310,00



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 17

OLAP-Ranking Functions Rank() versus Dense_Rank()

```
-- Rank(), Dense_Rank()
-----
Select * from SalesEmp;

Select Employee, Amount,
       Rank() Over(Order By Amount Desc) as Rank,
       Dense_Rank() Over(Order By Amount Desc) as DenseRank
From SalesEmp
Where SalesYear = 2008
;8
```

EMPLOYEE	AMOUNT	RANK	DENSERANK
Müller	120000,00	1	1
Bauer	110000,00	2	2
Jäger	110000,00	2	2
Huber	100000,00	4	3
Schmidt	100000,00	4	3
Meier	80000,00	6	4
Fischer	80000,00	6	4

Two second ranks:

- **Rank()** → No 3rd Rank, Next Rank = 4
- **Dense_Rank()** → Next Rank = 3



OLAP-Ranking-Functions Window-Partition-Clause

Window-Partition-Clause – Over(Partition By) Clause

- Must be specified within the **OVER(ORDER BY)** Clause
 - Immediately **after OVER**(→ Order By must be specified **after**)
 - Several column names can be specified
- Numbering is **reset** at **Level Change**
 - Example: Rank Sales Persons for several years



OLAP - Ranking Functions Window-Partition-Clause - Example

```
-- Partition By
-----
Select SalesYear, Row_Number() over(Partition By SalesYear Order By Amount Desc) RowNbr,
      CustNo, Amount
From SalesCustY
Order By SalesYear, Amount Desc
;|b
```

SALESYEAR	ROWNR	CUSTNO	AMOUNT
2008	1	10002	1350,00
2008	2	10003	535,00
2008	3	10004	470,00
2008	4	10005	310,00
2008	5	10001	115,00
2009	1	10003	4589,86
2009	2	10005	3741,95
2009	3	10004	2673,95
2009	4	10001	2634,20
2009	5	10002	1636,25
2010	1	10006	9712,85
2010	2	10003	1555,75
2010	3	10001	140,97

■ Reset numbering depending on the year



OLAP - Ranking Functions Restrictions

OLAP Functions are **not allowed** in **WHERE** Conditions!

Selection running numbers or ranks

■ Nested Sub-Selects

■ Common-Table-Expressions (CTE)

- Example: Determine the customers with the 5th to 10th highest Sales

```
select *
  from (select a.*, Dense_Rank() over(Order By Amount Desc) as DenseRank
        from SalesEmp a
        where SalesYear = 2009) as b
where DenseRank between 2 and 4
```

Nested Subselect

```
with x as (select a.*,
                 Dense_Rank() over(Order By Amount Desc) as DenseRank
          from SalesEmp a
          where SalesYear = 2009)
select * from x
where DenseRank between 2 and 4
```

Common Table Expression



Join Tables and Views



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 22

Join Tables/Views

Type Of Join	Description
(Inner) Join:	Only rows that are found in both tables/views are returned
Left (Outer) Join:	All rows from the first (left) table/view are returned even if there is no match in the second table/view
	If there is no equivalent in the second table, NULL values will be returned
Right (Outer) Join:	All rows from the second (right) table/view are returned even if there is no match in the first table/view
	If there is no equivalent in the second table, NULL values will be returned
Cross Join:	All rows of the first table/view are connected to all rows of the second table (cartesian product)
(Left) Exception Join:	All rows from the first (left) table/view are returned where in the second (right) table/view no match is found
Right Exception Join:	All rows from the second (right) table/view are returned where in the first (left) table/view no match is found
Full (Outer) Join:	All rows of both tables are returned independent if there is a match or not



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 23

Join Tables/Views – Inner Join



Inner Join: all rows that are available in both tables

```
Select Company, OrderNo, a.CustNo, CustName1, DelDate
  from OrderHdrX, AddressX
 where OrderHdrX.CustNo = AddressX.CustNo
```

Join occurs in the
WHERE Condition

```
Select Company, OrderNo, a.CustNo, CustName1, DelDate
  from OrderHdrX a, AddressX b
 where a.CustNo = b.CustNo
```

```
select  a.Company, a.OrderNo, a.CustNo, DelDate,
        ItemNo,   OrderQty,  DelQty
  from    OrderHdrX a inner join OrderDetX b
        on a.Company = b.Company and a.OrderNo = B.OrderNo
 where DelDate between '2009-12-01' and '2009-12-31'
```

Join occurs in the
FROM Clause

```
select Company, OrderNo, CustNo, DelDate,
        ItemNo,   OrderQty,  DelQty
  from OrderHdrX a inner join OrderDetX b using (Company, OrderNo)
 where DelDate between '2009-12-01' and '2009-12-31'
```



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 24

Join Table/Views with USING Clause



USING clause = *named-columns-join*

- Is a **shorthand way** of defining the **join condition**
- The specified **column** names must **exist in both** tables/views
- The result table contains **only the columns from the first** table/view
- Any column specified in the USING clause **cannot be qualified** in the **select-list** of the query
 - Before release 7.1 the columns had to be qualified!
 - For Right Outer or Right Exception Joins the **column values** are **NULL** and it is **NOT possible** to qualify or specify the **columns from the 2nd table**

Changed
Release 6.1



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 25

Join Tables/Views - Self Join

A table/view can be **joined** with **itself**

Key fields can be joined by any of the following operators

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

```
Select a.CustNo, a.SalesYear "Prev.Year", a.Amount "Amount Prev.Year", b.SalesYear, b.Amount,
b.Amount - a.Amount as Diff
from SalesCustY a join SalesCustY b
on a.SalesYear = b.SalesYear-1 and a.CustNo = b.CustNo
Order by a.CustNo, a.SalesYear
;;
```

Join year with the previous year

CUSTNO	Prev.Year	Amount Prev.Year	SALESYEAR	AMOUNT	DIFF
10001	2008	115,00	2009	2634,20	2519,20
10001	2009	2634,20	2010	140,97	-2493,23
10002	2008	1350,00	2009	1636,25	286,25
10003	2008	535,00	2009	4589,86	4054,86
10003	2009	4589,86	2010	1555,75	-3034,11
10004	2008	470,00	2009	2673,95	2203,95
10005	2008	310,00	2009	3741,95	3431,95



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 26

Join Tables/Views - Self Join

Cumulated Totals

```
-- Cumulated Sums
Select a.CustNo, a.SalesYear, a.Amount, sum(b.Amount) as Cumul
from SalesCustY a inner join SalesCustY b
on a.CustNo = b.CustNo and a.SalesYear >= b.SalesYear
Group By a.CustNo, a.SalesYear, a.Amount
Order By a.CustNo, a.SalesYear, a.Amount
;B
```

CUSTNO	SALESYEAR	AMOUNT	CUMUL
10001	2008	115,00	115,00
10001	2009	2634,20	2749,20
10001	2010	140,97	2890,17
10002	2008	1350,00	1350,00
10002	2009	1636,25	2986,25
10003	2008	535,00	535,00
10003	2009	4589,86	5124,86
10003	2010	1555,75	6680,61
10004	2008	470,00	470,00
10004	2009	2673,95	3143,95
10005	2008	310,00	310,00
10005	2009	3741,95	4051,95
10006	2010	9712,85	9712,85



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 27

Join Tables/Views - Left / Right Outer Join



Left Outer Join → All rows from the 1st table/view

- All rows from the second table where a match is found

```
select a.*, b.*
  from OrderHdrX a left outer join OrderDetX b
    on a.Company = b.Company
   and a.OrderNo = b.OrderNo
```

Right Outer Join → All rows from the 2nd table/view

- All rows from the first table where a match is found

```
select a.*, b.*
  from OrderHdrX a right outer join OrderDetX b
    on a.Company = b.Company
   and a.OrderNo = b.OrderNo
```



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 28

Join Tables/Views - Left/Right (Outer) Join



Compare sales of the current year with the sales of the previous year

```
select a.CustNo, a.SalesYear "Prev.Year", a.Amount "Amount Prev.Year",
       b.SalesYear, b.Amount,
       b.Amount - a.Amount as Diff
  from SalesCustY a Left outer join SalesCustY b
    on a.SalesYear = b.SalesYear-1 and a.CustNo = b.CustNo
 order by a.CustNo, a.SalesYear
 ;;
```

CUSTNO	Prev.Year	Amount	Prev.Year	SALESYEAR	AMOUNT	DIFF
10001	2008	115,00				
10001	2009	2634,20				
10001	2010	140,97				
10002	2008	1350,00				
10002	2009	1636,25				
10003	2008	535,00				
10003	2009	4589,86				
10003	2010	1555,75				
10004	2008	470,00				
10004	2009	2673,95				
10005	2008	310,00				
10005	2009	3741,95				
10006	2010	9712,85				

```
select b.CustNo, b.SalesYear-1 "Prev.Year", a.Amount "Amount Prev.Year",
       b.SalesYear, b.Amount,
       b.Amount - a.Amount as Diff
  from SalesCustY a Right outer join SalesCustY b
    on a.SalesYear = b.SalesYear-1 and a.CustNo = b.CustNo
 order by a.CustNo, a.SalesYear
 ;;
```

CUSTNO	Prev.Year	Amount	Prev.Year	SALESYEAR	AMOUNT	DIFF
10001	...	2008	115,00	2009	2634,20	2519,20
10001	...	2009	2634,20	2010	140,97	-2493,23
10002	...	2008	1350,00	2009	1636,25	286,25
10003	...	2008	535,00	2009	4589,86	4054,86
10003	...	2009	4589,86	2010	1555,75	-3034,11
10004	...	2008	470,00	2009	2673,95	2203,95
10005	...	2008	310,00	2009	3741,95	3431,95
10004	...	2007	-	2008	470,00	-
10003	...	2007	-	2008	535,00	-
10006	...	2009	-	2010	9712,85	-
10005	...	2007	-	2008	310,00	-
10002	...	2007	-	2008	1350,00	-
10001	...	2007	-	2008	115,00	-



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 29

Join Tables/Views - Union Full Outer Join before Release 6.1



```
Select a.CustNo, a.SalesYear "Prev.Year", a.Amount "Amount Prev.Year",
      b.Amount - a.Amount as Diff
from SalesCustY a Left outer join SalesCustY b
on a.SalesYear = b.SalesYear-1 and a.CustNo = b.CustNo

Union

Select a.CustNo, a.SalesYear "Prev.Year", a.Amount "Amount Prev.Year",
      b.Amount - a.Amount as Diff
from SalesCustY a Right outer join SalesCustY b
on a.SalesYear = b.SalesYear-1 and a.CustNo = b.CustNo

Order by 1, 2
```

**Merge a Left Outer Join
and a Right Outer Join or a
Right Exception Join by
using UNION (Distinct)**

CUSTNO	Prev.Year	Amount Prev.Year	SALESYEAR	AMOUNT	DIFF
10001	2008	115,00	2009	2634,20	2519,20
10001	2009	2634,20	2010	140,97	-2493,23
10001	2010	140,97	-	-	-
10002	2008	1350,00	2009	1636,25	286,25
10002	2009	1636,25	-	-	-
10003	2008	535,00	2009	4589,86	4054,86
10003	2009	4589,86	2010	1555,75	-3034,11
10003	2010	1555,75	-	-	-
10004	2008	470,00	2009	2673,95	2203,95
10004	2009	2673,95	-	-	-
10005	2008	310,00	2009	3741,95	3431,95
10005	2009	3741,95	-	-	-
10006	2010	9712,85	-	-	-
-	-	-	2008	470,00	-
-	-	-	2008	535,00	-
-	-	-	2010	9712,85	-
-	-	-	2008	310,00	-
-	-	-	2008	1350,00	-
-	-	-	2008	115,00	-



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 30

Join Tables/Views Full Outer Join – Release 6.1



```
-- Full Outer Join
Select Case When a.CustNo is NULL Then b.CustNo Else a.CustNo End "Customer",
      Case When a.SalesYear is NULL Then b.SalesYear-1 Else a.SalesYear End "Prev.Year",
      Case When a.Amount is NULL Then 0 Else a.Amount End "Amount Prev.Year",
      Case When b.SalesYear is NULL Then a.SalesYear+1 Else b.SalesYear End "Curr.Year",
      Case When b.Amount is NULL Then 0 Else b.Amount End "Amount Curr.Year",
      Case When a.Amount is NULL Then b.Amount
      When b.Amount is NULL Then 0 - a.Amount
      Else b.Amount - a.Amount End "Difference"
from SalesCustY a Full Outer Join SalesCustY b
on a.SalesYear = b.SalesYear-1 and a.CustNo = b.CustNo

Order By "Customer", "Prev.Year"
```

Full Outer Join
■ Null Values are converted
into Default Values

Customer	Prev.Year	Amount Prev.Year	Curr.Year	Amount Curr.Year	Difference
10001	2007	0,00	2008	115,00	-115,00
10001	2008	115,00	2009	2634,20	2519,20
10001	2009	2634,20	2010	140,97	-2493,23
10001	2010	140,97	2011	0,00	-140,97
10002	2007	0,00	2008	1350,00	1350,00
10002	2008	1350,00	2009	1636,25	286,25
10002	2009	1636,25	2010	0,00	-1636,25
10003	2007	0,00	2008	535,00	535,00
10003	2008	535,00	2009	4589,86	4054,86
10003	2009	4589,86	2010	1555,75	-3034,11
10003	2010	1555,75	2011	0,00	-1555,75
10004	2007	0,00	2008	470,00	470,00
10004	2008	470,00	2009	2673,95	2203,95
10004	2009	2673,95	2010	0,00	-2673,95
10005	2007	0,00	2008	310,00	310,00
10005	2008	310,00	2009	3741,95	3431,95
10005	2009	3741,95	2010	0,00	-3741,95
10006	2009	0,00	2010	9712,85	9712,85
10006	2010	9712,85	2011	0,00	-9712,85



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 31

Compare Full Outer Join



UNION – Left/Right outer Joins

Actual Runtime Info...

Optimization Time	895
Run Time (ms)	896
Statement Open Tl...	292
Statement Fetch Tl...	457
Statement Close Tl...	48
Rows Fetched	19
Total Times Query ...	1
Total Time For All R...	897
Synchronous Data...	0.0
Asynchronous Data...	0.0
Page Faults	0.0
Temporary Storage...	1
Total Times Tempora...	0.0
Reason Temporary R...	Host Variable Value Ch...

Information about S...

Statement Number	12
Statement Function	Select
Statement Operation	Open
Statement Type	Dynamic
Statement Name	STMT0015
Statement Outcome	Successful
SQL Return Code	0
SQLSTATE	00000
Cursor Name	CRSR0015

Full Outer Join

Actual Runtime Info...

Optimization Time (ms)	899
Run Time (ms)	904
Statement Open Tm...	304
Statement Fetch Tm...	Not Available
Statement Close Tm...	Not Available
Rows Fetched	0.0
Total Times Query W...	1
Total Time For All R...	905
Synchronous Data...	4
Asynchronous Data...	0.0
Page Faults	4
Temporary Storage...	1
Total Times Tempora...	0.0
Reason Temporary R...	No Ending Temporary

Information about SQ...

Statement Number	22
Statement Function	Select
Statement Operation	Open
Statement Type	Dynamic
Statement Name	STMT0020
Statement Outcome	Successful
SQL Return Code	0
SQLSTATE	00000
Cursor Name	CRSR0020

Join Tables/Views - Exception Join



Exception Join → All rows from the left table without match in the second Table

```
Select a.*
from OrderHdrX a exception join OrderDetX b
on      a.Company = b.Company
and a.OrderNo = b.OrderNo
```

Predicate EXISTS → Returns true if rows exist

```
Select a.*
from OrderHdrX a
where Not Exists (Select b.Company
from orderDetX b
where      a.Company = b.Company
and a.OrderNo = b.OrderNo)
```



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 33

Join Tables/Views - Cross Join

Cross Join → cartesian product

- Join Tables/Views with a single or a few rows
- No columns for to join available

```
SELECT CustNo, sum(Amount) "Total Customer", Total "Grant Total"  
FROM Sales cross join SalesTot  
GROUP BY CustNo, Total
```

```
SELECT CustNo, sum(Amount) as "Total Customer", Total "Grant Total",  
Sum(Amount)*100,00/Total as "Sales %"  
FROM Sales cross join SalesTot  
GROUP BY CustNo, Total
```

CUSTNO	Total Customer	Grant Total	Sales %
10006	9712,85	29465,78	32,96
10003	6680,61	29465,78	22,67
10005	4051,95	29465,78	13,75
10004	3143,95	29465,78	10,66
10001	2890,17	29465,78	9,80
10002	2986,25	29465,78	10,13



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 34

Union, Intersect and Except



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 35

Union – Clause

A Union-Clause allows to **combine several SELECT** Statements and execute them as if it was a **single one**.

Example: As soon as an order is completed it is copied to a history table. If a specific order is searched, both tables (OrderHeader and History Table) must be searched.

Prerequisites for UNION Clauses

- **Identical Number of Columns** in the **SELECT** List
- **Compatible data types** in the selected columns

Add **ALL** or **DISTINCT** → controls the output of **Duplicates**

- **UNION (DISTINCT):** Returns **No Duplicates**
DISTINCT is Default and optional
Order is required → Find duplicates → **Performance!**
- **UNION ALL:** Returns **Duplicates**



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 36

Compare Union/Union ALL

```
-- Without Duplicates
Select * From AddressX
Union
Select * From AddressX_Y
Order By 1, 2
;B
```

CUSTNO	CUSTNAME1	CUSTNAME2
00100	Pallhuber und Söhne	P. & Söhne
00110	Bahnleitner Gemischtwaren	
00120	Ebacko Nordrhein Westfalen	
00130	Deutscher-Paket-Dienst	DPD
00140	Neckermann	
00150	Radio-Müller	
00160	Media Markt	
00170	Sound Acoustics GmbH & Co KG	
00180	Radio Eckhard OHG	
00190	Fischer Art of Sound & Light GmbH	
00200	Rhein-Main-Sieg Getränke GmbH & Co.	
00210	Cyber Tronic GmbH	
00220	Hermann Müller TV KG	
00230	OBI-Baumarkt AG	
00240	Erwin Schmitt Gartenzubehör GmbH	
00250	Alfred Meyer Gartenzubehör KG	
00260	Haushaltsgeräte Hans Schmidt KG	

- **UNION / UNION DISTINCT**
→ Without duplicates
- **UNION ALL**
→ All rows

```
-- With Duplicates
Select * From AddressX
Union ALL
Select * From AddressX_X
Order By 1, 2
;B
```

CUSTNO	CUSTNAME1	CUSTNAME2
00100	Pallhuber und Söhne	P. & Söhne
00100	Pallhuber und Söhne	P. & Söhne
00110	Bahnleitner Gemischtwaren	
00110	Bahnleitner Gemischtwaren	
00120	Ebacko Nordrhein Westfalen	
00120	Ebacko Nordrhein Westfalen	
00130	Deutscher-Paket-Dienst	DPD
00130	Deutscher-Paket-Dienst	DPD
00140	Neckermann	
00140	Neckermann	
00150	Radio-Müller	
00150	Radio-Müller	
00160	Media Markt	
00160	Media Markt	
00170	Sound Acoustics GmbH & Co KG	
00170	Sound Acoustics GmbH & Co KG	
00180	Radio Eckhard OHG	
00180	Radio Eckhard OHG	



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 37

Toolmaker
ADVANCED EFFICIENCY

Merge a Left Outer Join and a Right Outer Join or a Right Exception Join by using UNION (Distinct)

CUSTNO	Prev.Year	Amount	Prev.Year	SALESYEAR	AMOUNT	DIFF
10001	2008	115,00	2009	2009	2634,20	2519,20
10001	2009	2634,20	2010	2010	140,97	-2493,23
10001	2010	140,97	-	-	-	-
10002	2008	1350,00	2009	2009	1636,25	286,25
10002	2009	1636,25	-	-	-	-
10003	2008	535,00	2009	2009	4589,86	4054,86
10003	2009	4589,86	2010	2010	1555,75	-3034,11
10003	2010	1555,75	-	-	-	-
10004	2008	470,00	2009	2009	2673,95	2203,95
10004	2009	2673,95	-	-	-	-
10005	2008	310,00	2009	2009	3741,95	3431,95
10005	2009	3741,95	-	-	-	-
10006	2010	9712,85	-	-	-	-
-	-	-	2008	2008	470,00	-
-	-	-	2008	2008	535,00	-
-	-	-	2010	2010	9712,85	-
-	-	-	2008	2008	310,00	-
-	-	-	2008	2008	1350,00	-
-	-	-	2008	2008	115,00	-

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 38

Toolmaker
ADVANCED EFFICIENCY

00001	00002	00003	00004	00005
20004	King, Stephen - Shining	03/28/2009	510	
20004	King, Stephen - Shining	04/14/2009	539	
20004	King, Stephen - Shining	08/12/2009	550	
20004	King, Stephen - Shining	09/29/2009	570	
Total Item King, Stephen - Shining	King, Stephen - Shining -		1169	
Total Customer 10009			1169	
20005	Grisham, John - Die Akte	02/28/2009	250	
20005	Grisham, John - Die Akte	03/07/2009	250	
20005	Grisham, John - Die Akte	10/30/2009	300	
20005	King, Stephen - Dret	11/21/2009	310	
20005	King, Stephen - Dret	01/27/2010	350	
20005	King, Stephen - Shining	03/28/2009	510	
20005	King, Stephen - Shining	04/14/2009	539	
20005	King, Stephen - Shining	08/12/2009	550	
20005	King, Stephen - Shining	09/29/2009	570	
Total Item King, Stephen - Shining	King, Stephen - Shining -		2019	
Total Customer 10009			2019	
20006	King, Stephen - Dret	09/16/2010	490	
20006	King, Stephen - Dret	11/24/2010	510	
20006	King, Stephen - Dret	01/29/2011	511	
20006	King, Stephen - Shining	02/28/2009	510	
20006	King, Stephen - Shining	03/07/2009	539	
20006	King, Stephen - Shining	07/11/2010	690	
20006	King, Stephen - Shining	08/12/2010	690	
20006	King, Stephen - Shining	08/28/2010	760	
20006	King, Stephen - Shining	09/08/2010	760	
20006	King, Stephen - Shining	13/07/2010	760	
Total Item King, Stephen - Shining	King, Stephen - Shining -		3720	
Total Customer 10006			3720	

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 39

INTERSECT Clause (Release V5R3)

Composition of several Sub-Selects

- Display all rows that are **identical** in both tables/views
- Prerequisites analog UNION
 - **Identical** Number of Columns in the **SELECT** List
 - **Compatible data types** in the selected columns

Can only be executed by the **SQL Query Engine (SQE)**

Current Table	
CustNo	Sales
10001	10.000,00
10002	20.000,00
10003	5.000,00
10004	7.000,00
10005	12.000,00
10006	15.000,00

Saved Table	
CustNo	Sales
10001	10.000,00
10002	20.000,00
10003	4.500,00
10004	6.000,00
10006	15.000,00

```
Select * From Current  
Intersect  
Select * From Saved
```

Result	
CustNo	Sales
10001	10.000,00
10002	20.000,00
10006	15.000,00



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 40

EXCEPT Clause (Release V5R3)

Composition of several Sub-Selects

- Display all rows that are in the **1st table/view** but **not** in the 2nd one
- Prerequisites analog UNION
 - **Identical** number of columns in the **SELECT** List
 - **Compatible data types** in the selected columns

Can only be executed by the **SQL Query Engine (SQE)**

Current Table	
CustNo	Sales
10001	10.000,00
10002	20.000,00
10003	5.000,00
10004	7.000,00
10005	12.000,00
10006	15.000,00

Saved Table	
CustNo	Sales
10001	10.000,00
10002	20.000,00
10003	4.500,00
10004	6.000,00
10006	15.000,00

```
Select * From Current  
Except  
Select * From Saved
```

Result	
CustNo	Sales
10003	5.000,00
10004	7.000,00
10005	12.000,00



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 41

INTERSECT

Determine all rows that are **identical (rows and column values)** in both tables/views

```
Select * From AddressX  
Intersect  
Select * From AddressX_Y  
Order By CustNo
```

Determine Duplicates

EXCEPT

Determine all rows where is **no equivalent** in the second table

```
( Select * From AddressX  
Except  
Select * From AddressX_Y)  
Union  
( Select * from AddressX_Y  
Except  
Select * from AddressX)  
Order By CustNo
```

Compare 2 tables/views
→ All rows all column values



Common Table Expressions



Common Table Expressions (CTE)

Part of Select Statements

- Introduced with release V4R4M0
- Must be specified before the Full-Select → **With CTEs ... Select**

Pseudo-Views only valid for the current SELECT Statement

- Alternate for (nested) Sub-Selects
- Complex SQL-Queries can be splitted into smaller parts
- CTE can be used in **other CTEs**
- **Order By** and **Fetch First X Rows** only are allowed within CTEs



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 44

Common Table Expressions Example



with a as (Select * from Sales
where SalesDate between '2009-01-01' and '2009-12-31'),
x as (select CustNo as xCustNo, sum(Amount) as TotalCust
from a
group by CustNo),
y as (select sum(Amount) as Total from a)
select a.*, TotalCust, Total
from a join x on CustNo = xCustNo
cross join y
Order By CustNo, ItemNo, SalesDate
;!

CUSTNO	ITEMNO	ITEM	SALESDATE	AMOUNT	TOTALCUST	TOTAL
10001	5100	King,Stephen - Es	04/15/2009	76,25	2634,20	15276,21
10001	5100	King,Stephen - Es	04/17/2009	20,00	2634,20	15276,21
10001	5100	King,Stephen - Es	07/21/2009	25,40	2634,20	15276,21
10001	5100	King,Stephen - Es	08/10/2009	120,00	2634,20	15276,21
10001	5100	King,Stephen - Es	08/22/2009	111,00	2634,20	15276,21
10001	5100	King,Stephen - Es	08/31/2009	355,00	2634,20	15276,21
10001	5100	King,Stephen - Es	09/04/2009	63,55	2634,20	15276,21
10001	5100	King,Stephen - Es	10/23/2009	152,40	2634,20	15276,21
10001	5100	King,Stephen - Es	11/12/2009	450,85	2634,20	15276,21
10001	5200	King,Stephen - DreI	01/30/2009	160,00	2634,20	15276,21
10001	5200	King,Stephen - DreI	04/25/2009	203,20	2634,20	15276,21
10001	5300	Grisham,John - Die Akte	03/23/2009	180,00	2634,20	15276,21
10001	5300	Grisham,John - Die Akte	07/22/2009	215,00	2634,20	15276,21
10001	5300	Grisham,John - Die Akte	08/26/2009	228,60	2634,20	15276,21
10001	5300	Grisham,John - Die Akte	10/04/2009	273,05	2634,20	15276,21
10002	5100	King,Stephen - Es	10/08/2009	444,50	1636,25	15276,21
10002	5200	King,Stephen - DreI	06/22/2009	20,00	1636,25	15276,21
10002	5200	King,Stephen - DreI	08/05/2009	25,40	1636,25	15276,21
10002	5400	King,Stephen - Shining	07/21/2009	250,00	1636,25	15276,21
10002	5400	King,Stephen - Shining	08/26/2009	255,00	1636,25	15276,21
10002	5400	King,Stephen - Shining	10/03/2009	317,50	1636,25	15276,21
10002	5400	King,Stephen - Shining	11/07/2009	323,85	1636,25	15276,21
10003	5200	King,Stephen - DreI	04/09/2009	444,50	4589,86	15276,21
10003	5200	King,Stephen - DreI	09/04/2009	180,00	4589,86	15276,21
10003	5200	King,Stephen - DreI	11/16/2009	528,45	4589,86	15276,21
10003	5400	King,Stephen - Shining	01/29/2009	533,00	4589,86	15276,21

- CTE A = All sales for 2009
- CTE X = Cumulated sales for 2009 per customer (based on A)
- CTE Y = Cumulated sales for 2009 (based on A)
- Final Select → Join A, X, Y



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 45

Nested Sub-Select vs Common Table Expressions

```
Select Item, count(Item)
from Sales a
group by Item
having count(Item) = (Select max(b.NbrItem)
                      from (Select count(Item)
                          as NbrItem
                          from Sales
                          group by (Item)) as b)
;;
```

Nested Sub-Select

ITEM	NBRITEM
King,Stephen - Shining	35

```
With b as (Select Item, count(Item) as NbrItem
from Sales
group by Item),
c as (Select Max(NbrItem) as MaxItem
from b)
Select b.*
from b join c on MaxItem = NbrItem
;|
```

Common Table Expression

ITEM	NBRITEM
King,Stephen - Shining	35

Determine the item with the most order lines

Common Table Expression

```
with CTEYearSales as (Select Year(SalesDate) as SalesYear, CustNo, Amount
from Sales),
CTETotalCust as (select SalesYear, CustNo,
cast(Sum(Amount) as Dec(7, 2)) as TotalCust
from CTEYearSales
group by SalesYear, CustNo),
CTETotalYear as (select SalesYear as "YEAR",
cast(Sum(TotalCust) as Dec(7, 2)) as TotalYear
from CTETotalCust
group by SalesYear),
CTEPercent as (select SalesYear, CustNo, TotalCust,
Cast(TotalCust/TotalYear * 100 as Dec(9, 4)) as Percent
from CTETotalCust join CTETotalYear
on SalesYear = "YEAR"),
CTETOTAL as (Select Cast(NULL as Char(1)) Order,
SalesYear, 'Total Year' Text, Sum(TotalCust) as TotalCust,
Cast(Round(Sum(Percent), 2) as Dec(9, 4)) as Percent
from CTEPercent
group By SalesYear, 2)
select ' ', SalesYear, CustNo,
Cast(sum(TotalCust) as Dec(9, 2)) as TotalCustomer,
Cast(sum(Percent) as Dec(9, 2)) as Percent
from CTEPercent
Group by SalesYear, CustNo
Union
Select Order, SalesYear, Text, TotalCust, Cast(Percent as Dec(9, 2))
From CTETotal
Order by 2, 1, 3
```

- Calculate sales per customer and year
- Percentage of sales
- Add total per year

Common Table Expression

```
with CTEYearSales as (Select Year(SalesDate) as SalesYear, CustNo, Amount
                        from Sales),
    CTETotalCust as (select SalesYear, CustNo,
                           cast(Sum(Amount) as Dec(7, 2)) as TotalCust
                     from CTEYearSales
                     group by SalesYear, CustNo),
    CTETotalYear as (select SalesYear as "YEAR",
                           cast(Sum(TotalCust) as Dec(7, 2)) as TotalYear
                     from CTETotalCust
                     group by SalesYear),
    CTEPercent as (select 00001, SALESYEAR, CUSTNO,
                          TOTALCUSTOMER, PERCENT
                    from CTETotalYear
                    group by SalesYear, CustNo),
    CTETOTAL as (Select
                  Cast(sum(TotalCust) as
                  Cast(sum(Percent) as
                    from CTEPercent
                    Group by SalesYear, CustNo
                Union
                Select Order, SalesYear, Text,
                From CTETotal
                Order by 2, 1, 3
```



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 48

Recursive SQL Queries



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 49

Recursive SQL Queries



Recursion, in mathematics and computer science, is a method of defining functions in which the function being defined is applied within its own definition; specifically it is defining an infinite statement using finite component. The term is also used more generally to describe a process of repeating objects in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested images that occur are a form of infinite recursion. (wikipedia)

Recursion in Tables or Views

■ **Relationship** between information in a **specific column** in a row with information in a **different column** in a different row

■ **Examples** for recursions:

- **Hierarchical relationship:** Organization charts / Bill of material
- **Bi-directional relationship:** Air / train connections / network plans



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 50

Recursive SQL Queries



Recursion, in mathematics and computer science, is a method of defining functions in which the function being defined is applied within its own definition; specifically it is defining an infinite statement using finite component. The term is also used more generally to describe a process of repeating objects in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested images that occur are a form of infinite recursion. (wikipedia)

Release V5R4 → Introduction recursive queries

■ **Recursive Queries** can only be executed by the **SQE**

■ **Recursive Common Table Expressions** (RCTE)

- Join a Common Table Expression using **UNION ALL** with **itself**

■ **Recursive Views**

- Implement a Recursive Common Table Expression within a View

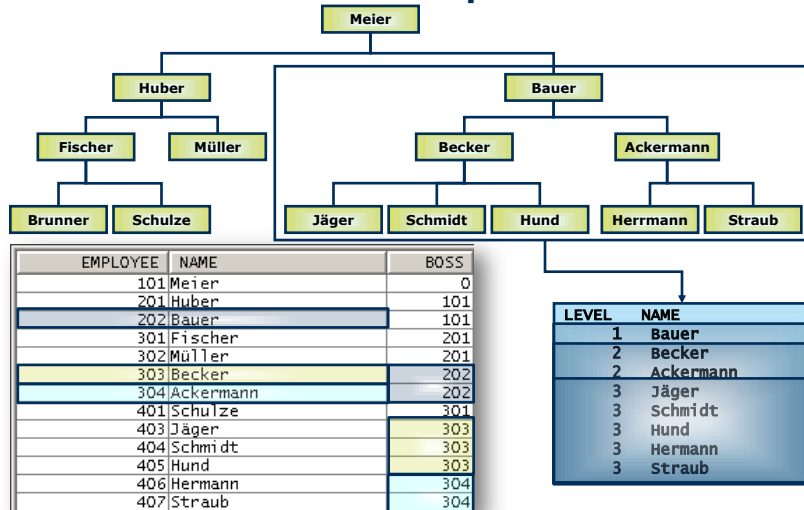


23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 51

Recursive SQL-Queries Hierarchical Relationship



Recursive SQL Queries Hierarchical Relationships

Initializing Select

→ Part of the Common Table Expression CTE

- Declare the **start position**
- Is only **executed once** → will **not** be considered for recursion !

Iterative/Recursive Select

→ Part of the Common Table Expression

- **UNION ALL** → combines all result rows
- Merges the CTE with itself → **Parent/Child relationship**

Final/Main Select

- Processes the **Common Table Expression**

Recursive SQL Queries Hierarchical Relationships

With StaffList (Level, Employee, Name, Boss)

Common Table Expression

```
as (
  -- Initialization
  Select 1, Employee, Name, Boss
  from Staff
  where Name = 'Bauer'
```

Initialization

```

  -- Recursion
  Union All
  Select Level+1, o.Employee, o.Name, o.Boss
  from Staff o join StaffList p
  on o.Boss = p.Employee)
```

Recursion/Iteration

```
-- Final/Main Select
select *
  from StaffList
 Order By Level
```

Final/Main Select Statement

LEVEL	EMPLOYEE	NAME	BOSS
1	202	Bauer	...
2	303	Becker	...
2	304	Ackermann	...
3	403	Jäger	...
3	404	Schmidt	...
3	405	Hund	...
3	406	Hermann	...
3	407	Straub	...



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 55

Recursive SQL Hierarchical Relationships - Examples

```
With StaffList (Level, Employee, Name, Boss)
as (Select 1, Employee, Name, Cast('' as VarChar(2000))
  from Staff
  where Name = 'Bauer'
  Union All
  Select Level+1, o.Employee, o.Name,
    Cast(Trim(p.boss) concat ', ' concat
      Trim(p.Name) as VarChar(2000))
  from StaffList p join Staff o
  on p.Employee = o.Boss)
select Level, Employee,
  Cast(Name as VarChar(20)) Name,
  Trim(Trim(Leading ', ' From Trim(Boss))) "Who are the bosses"
  from StaffList
 Order By Level
```

**Recursion
→ List all Managers**

LEVEL	EMPLOYEE	NAME	Who are my bosses
1	202	Bauer	
2	303	Becker	Bauer
2	304	Ackermann	Bauer
3	403	Jäger	Becker, Bauer
3	404	Schmidt	Becker, Bauer
3	405	Hund	Becker, Bauer
3	406	Hermann	Ackermann, Bauer
3	407	Straub	Ackermann, Bauer



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 56

Recursive SQL Hierarchical Relationships - Examples



```
With Counter (Nbr) as (
    Select 100 from SysIbm/SysDummy1
    Union All
    Select Nbr + 10 from Counter
    Where Nbr < 250)
Select * from Counter
```



Generate Running Numbers

NBR
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250



23.04.2012 | Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 57

Recursive SQL Hierarchical Relationships - Examples



```
With Calendar (RunDate) as (
    Values(Cast('2009-11-01' as Date))
    Union All
    Select RunDate + 1 Day from Calendar
    Where RunDate < '2009-11-30'),
    TotalSales as (Select SalesDate, Sum(Amount) as Total
    from Sales
    Group By SalesDate)
Select rundate, coalesce(Total, 0) Total
from Calendar Left Outer join TotalSales on RunDate = SalesDate
```

- List all days in November 2009
- Show Sales per Day

RUNDATE	TOTAL
11/01/2009	0,00
11/02/2009	0,00
11/03/2009	0,00
11/04/2009	0,00
11/05/2009	0,00
11/06/2009	0,00
11/07/2009	666,75
11/08/2009	0,00
11/09/2009	0,00
11/10/2009	0,00
11/11/2009	0,00
11/12/2009	450,85
11/13/2009	0,00
11/14/2009	0,00
11/15/2009	0,00
11/16/2009	228,60
11/17/2009	0,00
11/18/2009	0,00
11/19/2009	0,00
11/20/2009	0,00
11/21/2009	0,00
11/22/2009	0,00
11/23/2009	0,00
11/24/2009	0,00
11/25/2009	0,00
11/26/2009	0,00
11/27/2009	0,00
11/28/2009	0,00
11/29/2009	0,00
11/30/2009	279,40



23.04.2012 | Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 58

Recursive SQL

Hierarchical Relationships - Examples



```
With AllNbr (Level)
as (
    Values(10000)
    Union All
    Select Level + 1 from AllNbr
    Where Level < 10010),
Cust as (Select Distinct CustNo from Sales)
Select *
from AllNbr exception join Cust on Level = CustNo
;1
```

Find missing customer no.

LEVEL	CUSTNO
10000	-
10007	-
10008	-
10009	-
10010	-



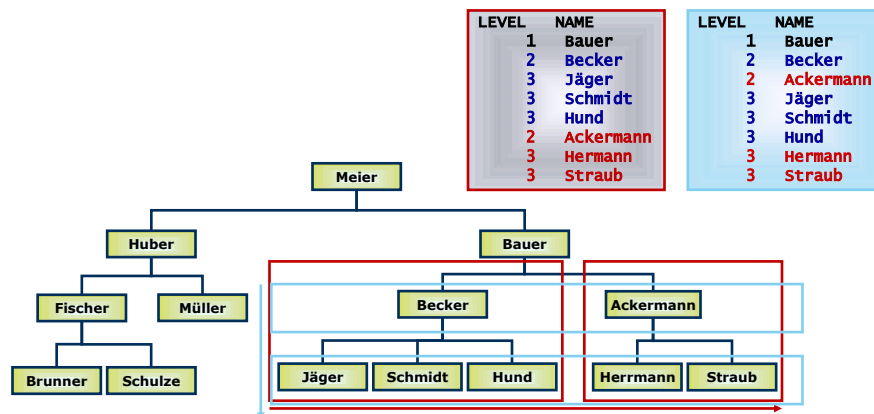
23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 59

Recursive SQL Queries

Hierarchical Relationships



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 60

Recursive SQL Queries

Search Clause



Search Clause

- Must be specified **after** the **Iterative Select** within the **CTE**

Options:

- **BREADTH FIRST:** **Default**
Decomposition/Order according to the **Level**
- **DEPTH FIRST:** Decomposition of the **complete branches**
Result must **be stored temporary** and **ordered at the end** of the execution
→ An **additional Order Column** will be generated
→ Column must be specified in the **Order By Clause**
Only DEPTH FIRST → Not BREADTH FIRST
→ **May result in a bad performance!**



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 61

Recursive SQL-Queries

Search Clause



```
With StaffList (Level, Employee, Name, Boss)
```

```
as (-- Initialization
    Select 1, Employee, Name, Boss
    from Staff
    where Name = 'Bauer'
```

Initialization

```
-- Recursion
Union All
Select Level+1, o.Employee, o.Name, o.Boss
from Staff o join StaffList p
on o.Boss = p.Employee)
```

Recursion/Iteration

```
-- Search Depth Clause
Search Depth first by Employee set SortSeq
```

Search Depth Clause
■ with Sort field

```
-- Final/Main SQL-Statement
select *
from StaffList
Order By SortSeq
```

Final/Main Select Statement

Breadth first

LEVEL	EMPLOYEE	NAME	BOSS
1	202	Bauer	101
2	303	Becker	202
2	304	Ackermann	202
3	403	Jäger	303
3	404	Schmidt	303
3	405	Hund	303
3	406	Hermann	304
3	407	Straub	304

Depth first

LEVEL	EMPLOYEE	NAME	BOSS
1	202	Bauer	101
2	303	Becker	202
3	403	Jäger	303
3	404	Schmidt	303
3	405	Hund	303
2	304	Ackermann	202
3	406	Hermann	304
3	407	Straub	304



23.04.2012

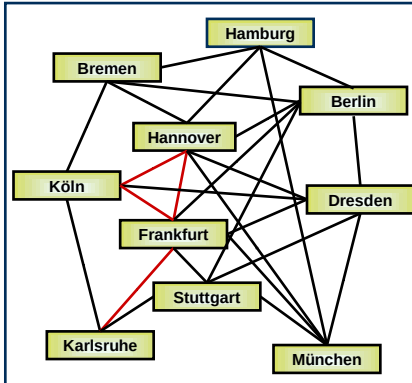
Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 62

Recursive SQL Queries Two-Way Relationships



Examples: Air/Train Connections / Network Plans



- Problem: **Cycles!** → Infinite loops
- Check with **CYCLE Clause**

CYCLE Clause

- In CTE after the **iterative Select**
- **Define Pseudo Columns**
 - For use in **Selection + Order By**
- **Predefine Return Value for**
 - Cyle
 - No Cycle
- Cycle detected
→ **Stop decomposition** branch

Attention: For Two-Way Relationships CYCLE-Clause is mandatory



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 63

Recursive SQL Queries Two-Way Relationships



```
WITH destinations (departure, arrival, itinerary)
```

```
as (
  -- Initialization
  SELECT f.departure, f.arrival,
        CAST (f.departure concat '->' concat f.arrival AS VARCHAR(200))
  FROM flights f
  WHERE f.departure = 'Frankfurt'
```

Initializing Select

```

  -- Recursion
  UNION ALL
  SELECT r.departure, b.arrival,
        CAST (r.itinerary concat '->' concat b.arrival AS VARCHAR(200))
  FROM destinations r, flights b
  WHERE r.arrival = b.departure)
```

Iteration/Recursion

```

  -- Cycle Clause
  CYCLE arrival SET cyclic_data TO '1' DEFAULT '0'
```

Cycle Clause
1=Cycle / 0 = No Cyle

```

-- Main/Final Select Statement
SELECT departure, arrival, itinerary, cyclic_data
FROM destinations
order by departure, arrival
;;
```

Final/Main Select



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 64

Recursive SQL Queries Two-Way Relationships

DEPARTURE	ARR...	ITINERARY	CYCLIC_DATA
Frankfurt	Berlin	Frankfurt->Berlin	
Frankfurt	Berlin	Frankfurt->Hannover->Berlin	
Frankfurt	Berlin	Frankfurt->Stuttgart->Berlin	
Frankfurt	Berlin	Frankfurt->München->Berlin	
Frankfurt	Berlin	Frankfurt->Hannover->Bremen->Berlin	0
Frankfurt	Berlin	Frankfurt->Hannover->Frankfurt->Berlin	0
Frankfurt	Berlin	Frankfurt->Hannover->Hamburg->Berlin	0
Frankfurt	Berlin	Frankfurt->Köln->Frankfurt->Berlin	0
Frankfurt	Berlin	Frankfurt->Köln->Hannover->Berlin	0
Frankfurt	Berlin	Frankfurt->Köln->Bremen->Berlin	0
Frankfurt	Berlin	Frankfurt->Karlsruhe->Frankfurt->Berlin	0
Frankfurt	Berlin	Frankfurt->Karlsruhe->Stuttgart->Berlin	0
Frankfurt	Berlin	Frankfurt->Stuttgart->Frankfurt->Berlin	0
Frankfurt	Berlin	Frankfurt->Stuttgart->München->Berlin	0
Frankfurt	Berlin	Frankfurt->München->Stuttgart->Berlin	0
Frankfurt	Berlin	Frankfurt->München->Frankfurt->Berlin	0
Frankfurt	Berlin	Frankfurt->München->Hannover->Berlin	0
Frankfurt	Berlin	Frankfurt->Berlin->Hamburg->Berlin	1
Frankfurt	Berlin	Frankfurt->Berlin->Bremen->Berlin	1
Frankfurt	Berlin	Frankfurt->Berlin->Hannover->Berlin	1
Frankfurt	Berlin	Frankfurt->Berlin->München->Berlin	1
Frankfurt	Berlin	Frankfurt->Berlin->Frankfurt->Berlin	1
Frankfurt	Berlin	Frankfurt->Berlin->Stuttgart->Berlin	1
Frankfurt	Berlin	Frankfurt->Hannover->Köln->Frankfurt->Berlin	0

**Initializing SELECT –
Not Checked for recursion!**



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 65

Hierarchical Query Clause



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 66

Recursive Queries



Recursive Common Table Expressions (CTE)

- Available since Release V5R4
- **Common Table Expression** is used to
 - define the **Starting Point** → **Initial Select**
 - describe the **Recursion** → By using an **UNION ALL** clause and **merging the CTE with itself**
→ **Iterative Select**

Hierarchical Query Clause

- Shipped with PTF SF99701 Version 9 (Release 7.1)
- **CONNECT BY** clause
 - to define the **association** between **parent and child** rows
- **CONNECT BY** clause is **easier to understand** than Recursive CTEs but **does not cover everything** that is possible with recursive CTEs

**Now PTF
Release 7.1**



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 67

Hierarchical Query Clause



Hierarchical Query Clause is part of a Sub-Select

- Must be specified after the WHERE, GROUP BY, HAVING clauses

Hierarchical Query Clause is composed of 2 parts

- **START WITH** **Starting Point**
(= **Initial Select** in recursive CTEs)
- **CONNECT BY** **Linkage** between parent and child rows
(= **Iterative Select** in recursive CTEs)

Special Order By Clause

- **ORDER SIBLINGS BY** for **sorting** the returned **child rows**



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 68

Hierarchical Query Clause Start With ... Connect By - Example

```
Select 101 as Boss, Employee, Name
From Staff
Start With Boss = 101
Connect By Employee = Boss;
```

BOSS	EMPLOYEE	NAME
101	201	Huber ...
101	202	Bauer ...

```
Select 303 as Boss, Employee, Name
From Staff
Start With Boss = 303
Connect By Employee = Boss;
```

BOSS	EMPLOYEE	NAME
303	403	Jäger ...
303	404	Schmidt ...
303	405	Hund ...

■ Retrieve **direct** subordinates
for manager 101 and 303

Select * from Staff - 62.176.136.71(E...

EMPLOYEE	NAME	BOSS
101	Meier ...	0
201	Huber ...	101
202	Bauer ...	101
301	Fischer ...	201
302	Müller ...	201
303	Becker ...	202
304	Ackermann ...	202
401	Schulze ...	301
403	Jäger ...	303
404	Schmidt ...	303
405	Hund ...	303
406	Hermann ...	304
407	Straub ...	304

```
Select 101 Boss, Employee, Name
from Staff
Where Boss = 101;
```



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 69

Hierarchical Query Clause PRIOR

PRIOR expression

Can be specified in composition with CONNECT BY

- If **PRIOR** is specified **ALL** sub-ordinated elements are joined
 - Otherwise **only the next level** will be determined
- A **PRIOR** operator has a **higher precedence** than **any** infix operator
 - If the expression contains other operators (such as + or concat) parenthesis must be set



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 70

Hierarchical Query Clause Start With ... Connect By Prior - Example



```
Select 202 Boss, Employee, Name
From Staff
Start With Boss = 202
Connect By Prior Employee = Boss;
```

BOSS	EMPLOYEE	NAME
202	303	Becker
202	405	Hund
202	404	Schmidt
202	403	Jäger
202	304	Ackermann
202	407	Straub
202	406	Hermann

- Retrieve all subordinates for manager 202 Bauer

Select * from Staff - 62.176.136.71(E...

EMPLOYEE	NAME	BOSS
101	Meier	...
201	Huber	...
202	Bauer	...
301	Fischer	...
302	Müller	...
303	Becker	...
304	Ackermann	...
401	Schulze	...
403	Jäger	...
404	Schmidt	...
405	Hund	...
406	Hermann	...
407	Straub	...



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 71

Hierarchical Query Clause Order Siblings By



Order Siblings By

- Special Order By clause
- Enforces ordering a set of rows produced by the same parent

```
Select 202 Boss, Employee, Name
From Staff
Start With Boss = 202
Connect By Prior Employee = Boss
Order Siblings by Employee;
```

BOSS	EMPLOYEE	NAME
202	303	Becker
202	403	Jäger
202	404	Schmidt
202	405	Hund
202	304	Ackermann
202	406	Hermann
202	407	Straub

```
Select 202 Boss, Employee, Name
From Staff
Start With Boss = 202
Connect By Prior Employee = Boss
Order Siblings by Name;
```

BOSS	EMPLOYEE	NAME
202	304	Ackermann
202	406	Hermann
202	407	Straub
202	303	Becker
202	405	Hund
202	403	Jäger
202	404	Schmidt



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 72

Hierarchical Queries

CONNECT_BY_ROOT



CONNECT_BY_ROOT (expression)

CONNECT_BY_ROOT only for Hierarchical Queries

- Returns the **row's Root Ancestor** for **every row** in the hierarchy
- Must be specified within the **SELECT** or **WHERE** clause
- **Can not** be specified within the **START WITH** or **CONNECT BY** clause

```

Select Connect_By_Root Boss Boss, Employee, Name
  From Staff
 Start With Boss = 202
 Connect By Prior Employee = Boss
 Order Siblings by Employee;
  
```

BOSS	EMPLOYEE	NAME
202	303	Becker
202	403	Jäger
202	404	Schmidt
202	405	Hund
202	304	Ackermann
202	406	Hermann
202	407	Straub



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 73

Hierarchical Query Clause

CONNECT_BY_ROOT



```

Select Connect_By_Root (b.Boss concat ' ' concat
                        (Select Name
                          from Staff a
                          where a.Employee = b.Boss))
      Employee, Name
  From Staff b
 Start With Boss in (201, 202) or Name like '%mann%'
 Connect By Prior Employee = Boss;
  
```

■ **Connect_By_Root**
in composition with
a sub-select

■ **Multiple Starting
Points**

Boss	EMPLOYEE	NAME
201 Huber	...	301 Fischer
201 Huber	...	401 Schulze
201 Huber	...	302 Müller
202 Bauer	...	303 Becker
202 Bauer	...	405 Hund
202 Bauer	...	404 Schmidt
202 Bauer	...	403 Jäger
202 Bauer	...	304 Ackermann
202 Bauer	...	407 Straub
202 Bauer	...	406 Hermann
304 Ackermann	...	406 Hermann



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 74

Hierarchical Query Clause Pseudo Columns



A pseudo column is an **identifier** with

- A given **name** and a predefined **functionality**
- Can be used within the **assigned context** like **any column** in a table or view

For hierarchical queries **3 Pseudo Columns** are available

- **Level** Returns the **recursive step no** in the hierarchy at which the row was produced
 1 → Rows produced by **START WITH**
 >= 2 → Rows produced by **CONNECT BY**
- **Connect_By_isCycle** For **bi-directional relationships** to detect a **cycle**
 1 → Row **is part of a cycle** in the hierarchy
 0 → Row is not a part of a cycle
- **Connect_By_isLeaf** 1 → **Hierachical** relationships = **last level**
 → **Bi-directional** relationships = **cycle**
 0 → Neither last level nor cycle



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 75

Hierarchical Queries Pseudo Column Level



```
Select Connect_By_Root Boss "Boss",  
       Level, Employee, Name  
From Staff  
Start With Boss = 101  
Connect By Prior Employee = Boss  
Order By Level, Employee;
```

- Force **Search Breadth First** by using pseudo column **Level**

Boss	LEVEL	EMPLOYEE	NAME
101	1	201	Huber ...
101	1	202	Bauer ...
101	2	301	Fischer ...
101	2	302	Müller ...
101	2	303	Becker ...
101	2	304	Ackermann ...
101	3	401	Schulze ...
101	3	403	Jäger ...
101	3	404	Schmidt ...
101	3	405	Hund ...
101	3	406	Hermann ...
101	3	407	Straub ...



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 76

Hierarchical Queries Pseudo Column Level

```
Select Employee, Name, Level,
       Space((Level - 1) * 3) concat '/' concat Employee
       concat ' - ' concat Trim(Name) x
from Staff
Start With Boss = 101
Connect By Prior Employee = Boss
Order Siblings By Employee;
```

EMPLOYEE	NAME	LEVEL	X
201	Huber	...	1/201 - Huber
301	Fischer	...	2/301 - Fischer
401	Schulze	...	3/401 - Schulze
302	Müller	...	2/302 - Müller
202	Bauer	...	1/202 - Bauer
303	Becker	...	2/303 - Becker
403	Jäger	...	3/403 - Jäger
404	Schmidt	...	3/404 - Schmidt
405	Hund	...	3/405 - Hund
304	Ackermann	...	2/304 - Ackermann
406	Hermann	...	3/406 - Hermann
407	Straub	...	3/407 - Straub



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 77

Hierarchical Query Clause SYS_CONNECT_BY_PATH

**SYS_CONNECT_BY_PATH (String-Expression1 ,
String-Expression2)**

Builds a string containing a path from the root to this row

- Can only be used in the **context of hierarchical queries**
Can neither be used with START WITH nor CONNECT BY
- String/Path is returned as **CLOB (1M)**
- *String-Expression1:*
(Character) String **representing the row**
- *String-Expression2:*
String expression that **serves as separator**

No function to concatenate/calculate numeric values available

- For calculation a **Recursive Common Table Expression (RCTE)** or an **User Defined Function (UDF)** must be used



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 78

Hierarchical Query Clause 2-way Relationships - Example



```
Select Connect_By_Root Departure, Arrival, Level,
       Connect_By_Root Departure concat Sys_Connect_By_Path(Arrival, ' - ')
from flights
Where   Arrival = 'Berlin'
       and Connect_By_isCycle = 0
       and Level <= 3
Start With Departure = 'Frankfurt'
Connect By NoCycle Prior Arrival = Departure
Order By Level
```

Composing **CONNECT_BY_ROOT**
and **SYS_CONNECT_BY_PATH**

Returns all connections up to Level = 3 between Frankfurt and Berlin

00001	ARR...	LEVEL	00004
Frankfurt	Berlin	1	Frankfurt - Berlin
Frankfurt	Berlin	2	Frankfurt - Hannover - Berlin
Frankfurt	Berlin	2	Frankfurt - Stuttgart - Berlin
Frankfurt	Berlin	2	Frankfurt - München - Berlin
Frankfurt	Berlin	3	Frankfurt - Hannover - Hamburg - Berlin
Frankfurt	Berlin	3	Frankfurt - Hannover - Frankfurt - Berlin
Frankfurt	Berlin	3	Frankfurt - Hannover - Bremen - Berlin
Frankfurt	Berlin	3	Frankfurt - Köln - Bremen - Berlin
Frankfurt	Berlin	3	Frankfurt - Köln - Hannover - Berlin
Frankfurt	Berlin	3	Frankfurt - Köln - Frankfurt - Berlin
Frankfurt	Berlin	3	Frankfurt - Karlsruhe - Stuttgart - Berlin
Frankfurt	Berlin	3	Frankfurt - Karlsruhe - Frankfurt - Berlin
Frankfurt	Berlin	3	Frankfurt - Stuttgart - München - Berlin
Frankfurt	Berlin	3	Frankfurt - Stuttgart - Frankfurt - Berlin
Frankfurt	Berlin	3	Frankfurt - München - Hannover - Berlin
Frankfurt	Berlin	3	Frankfurt - München - Frankfurt - Berlin
Frankfurt	Berlin	3	Frankfurt - München - Stuttgart - Berlin

START WITH
is **not** checked
to determine a
cycle



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 79

Hierarchical Query Clause 2-way Relationships - Example



```
Select Connect_By_Root Departure, Arrival, Level,
       Connect_By_Root Departure concat Sys_Connect_By_Path(Arrival, ' - ')
from flights
Where   Arrival = 'Berlin'
       and connect_By_isCycle = 0
       and Level <= 3
Start With Departure = 'Frankfurt'
Connect By NoCycle Prior      Arrival = Departure
                           and Arrival <> 'Frankfurt'
Order By Level
```

■ Skip starting point



00001	ARRIVAL	LEVEL	00004
Frankfurt	Berlin	1	Frankfurt - Berlin
Frankfurt	Berlin	2	Frankfurt - Hannover - Berlin
Frankfurt	Berlin	2	Frankfurt - Stuttgart - Berlin
Frankfurt	Berlin	2	Frankfurt - München - Berlin
Frankfurt	Berlin	3	Frankfurt - Hannover - Hamburg - Berlin
Frankfurt	Berlin	3	Frankfurt - Hannover - Bremen - Berlin
Frankfurt	Berlin	3	Frankfurt - Köln - Bremen - Berlin
Frankfurt	Berlin	3	Frankfurt - Köln - Hannover - Berlin
Frankfurt	Berlin	3	Frankfurt - Karlsruhe - Stuttgart - Berlin
Frankfurt	Berlin	3	Frankfurt - Stuttgart - München - Berlin
Frankfurt	Berlin	3	Frankfurt - München - Hannover - Berlin
Frankfurt	Berlin	3	Frankfurt - München - Stuttgart - Berlin



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 80

Multidimensional Grouping



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 81

Multi-dimensional Grouping Release 6.1

New Extentions for the Group By Clause:

- **Cube(Grouping Columns/Fields)**
- **RollUp(Grouping Columes/Fields)**
- **Grouping Sets(Grouping Columns/Fields)**

Execution only with the **SQL Query Engine (SQE)**

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



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 82

Multi-dimensional Grouping Extention RollUp()



Extention RollUp()

- 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



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 83

Multi-dimensional Grouping Extention RollUp() - Example



```
select Year(SalesDate) SalesYear, Month(SalesDate) SalesMonth,  
       Day(SalesDate) SalesDay, Sum(Amount) Total  
From Sales  
Where SalesDate between '2008-10-01' and '2009-03-31'  
Group By RollUp(Year(SalesDate), Month(SalesDate), Day(SalesDate))  
Order By SalesYear, SalesMonth, SalesDay  
;
```

SALESYEAR	SALESMONTH	SALESDAY	TOTAL
2008	11	1	55,00
2008	11	5	35,00
2008	11	7	45,00
2008	11	15	1350,00
2008	11	30	310,00
2008	11	-	1795,00
2008	12	9	150,00
2008	12	17	350,00
2008	12	22	425,00
2008	12	23	60,00
2008	12	-	985,00
2008	-	-	2780,00
2009	1	29	333,00
2009	1	30	160,00
2009	1	31	340,00
2009	1	-	833,00
2009	2	8	200,00
2009	2	28	225,00
2009	2	-	425,00
2009	3	15	200,00
2009	3	17	1250,00
2009	3	23	180,00
2009	3	28	50,00
2009	3	-	1680,00
2009	-	-	2938,00
-	-	-	5718,00

RollUp() – Sub-Totals:

- Sales Year, Month, Day
- Sales Year, Month
- Sales Year
- Grant Total

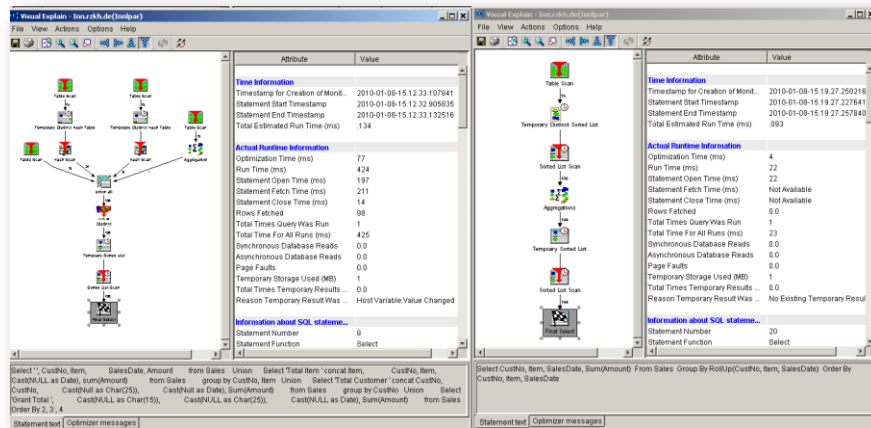


23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 84

Compare - Grouping



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 85

Multidimensional Grouping Extension Cube

Extension Cube

- Generate 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**



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 86

Multi-dimensional Grouping Extention Cube() - Examples



```
select Year(SalesDate) SalesYear, CustNo, Item, Sum(Amount) Total
From Sales
Where SalesDate between '2008-01-01' and '2009-12-31'
and CustNo in ('10001', '10002')
Group By Rollup(Year(SalesDate), CustNo, Item)
--Order By 1, 2, 3
Order By SalesYear, CustNo
```

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
2008	-	King,Stephen - Es	1465,00
2009	10001	Grisham,John - Die Akte	896,65
2009	10001	King,Stephen - Drei	363,20
2009	10001	King,Stephen - Es	1374,35
2009	10001	-	2634,20
2009	10002	King,Stephen - Drei	45,40
2009	10002	King,Stephen - Es	444,50
2009	10002	King,Stephen - Shining	1146,35
2009	10002	-	1636,25
2009	-	-	4270,45
2009	-	Grisham,John - Die Akte	896,65
2009	-	King,Stephen - Drei	408,60
2009	-	King,Stephen - Es	1818,85
2009	-	King,Stephen - Shining	1146,35
2009	-	Grisham,John - Die Akte	896,65
-	10001	King,Stephen - Drei	363,20
-	10001	King,Stephen - Es	1489,35
-	10001	-	2749,20
-	10002	King,Stephen - Drei	45,40
-	10002	King,Stephen - Es	1794,50
-	10002	King,Stephen - Shining	1146,35
-	10002	-	2986,25
-	-	-	5735,45
-	-	Grisham,John - Die Akte	896,65
-	-	King,Stephen - Drei	408,60
-	-	King,Stephen - Es	3283,85
-	-	King,Stephen - Shining	1146,35

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	896,65
2009	10001	King,Stephen - Drei	363,20
2009	10001	King,Stephen - Es	1374,35
2009	10001	-	2634,20
2009	10002	King,Stephen - Drei	45,40
2009	10002	King,Stephen - Es	444,50
2009	10002	King,Stephen - Shining	1146,35
2009	10002	-	1636,25
2009	-	-	4270,45
-	-	-	5735,45

Cross Tabulation



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 87

Multi-dimensional Grouping



Extention Grouping Sets()

- 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

Attention: There is still a bug when using scalar function in composition with Grouping Sets()
 → Works correctly when Nested Sub-Selects or CTEs are used



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 88

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	896,69
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	140,97
2010	-	King, Stephen - Shining	1555,75
2010	10001	-	140,97
2010	10003	-	1555,75
2010	-	-	1696,72
2010	-	-	12557,03

Sub-Total SalesYear/Item

Sub-Total SalesYear/Customer

Sub-Total SalesYear

Grand Total



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 89

Multi-dimensional Grouping Grouping Sets() and Rollup()/Cube()



```
Select Coalesce(Grp1, Grp2) as "Group",
       Week_Iso(SalesDate) as "WeekISO",
       DayOfWeek_ISO(SalesDate) as "DayOfWeek",
       Month(SalesDate) as "Month",
       CustNo, SUM(Amount) as "Total"
From Sales Cross Join
     (Values('Group 1', 'Group 2')) as X(Grp1, Grp2)
Where Year(SalesDate) = 2009
and Week_Iso(SalesDate) between 15 and 19
Group By Grouping Sets((Grp1, Cube(Week_Iso(SalesDate),
                                   DayOfWeek_ISO(SalesDate))),
                       (Grp2, Rollup(Month(SalesDate), CustNo)))
Order By "Group", CustNo, "WeekISO", "DayOfWeek", "Month"
;;
```

Group	WeekISO	DayOfWeek	Month	CUSTNO	Total
Group 1	15	4	-	-	444,50
Group 1	15	-	-	-	444,50
Group 1	16	2	-	-	539,75
Group 1	16	3	-	-	76,20
Group 1	16	5	-	-	20,00
Group 1	16	-	-	-	635,95
Group 1	17	6	-	-	203,20
Group 1	17	7	-	-	422,91
Group 1	17	-	-	-	626,11
Group 1	18	1	-	-	431,80
Group 1	18	3	-	-	150,00
Group 1	18	-	-	-	581,80
Group 1	-	1	-	-	431,80
Group 1	-	2	-	-	539,75
Group 1	-	3	-	-	226,20
Group 1	-	4	-	-	444,50
Group 1	-	5	-	-	20,00
Group 1	-	6	-	-	203,20
Group 1	-	7	-	-	422,91
Group 1	-	-	-	-	2288,36
Group 2	-	-	-	410001	299,40
Group 2	-	-	-	410003	867,41
Group 2	-	-	-	410004	1121,55
Group 2	-	-	-	41	2288,36
Group 2	-	-	-	-	2288,36

Composition Grouping Sets
and Super Groups
(= Cube() and Rollup())



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 90

Multi-Dimensional Grouping Aggregat-Function Grouping



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

Bug or works as designed?

- Currently not yet allowed in composition with scalar functions
- Can only specified once within the same sub-select



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 91

Multi-dimensional Grouping Aggregate Function Grouping - Example



```
With x as (Select Year(SalesDate) SalesYear, CustNo, Amount
            from Sales)
Select Grouping(SalesYear) "GroupingYear",
       Grouping(CustNo)    "GroupingCustNo",
       Case When CustNo is NULL and SalesYear is NULL Then 'Total'
       When CustNo is NULL Then 'Total Year'
       else '' End "TextNULL",
       SalesYear, CustNo, Sum(Amount) as Total
from x
Group By Rollup(SalesYear, CustNo)
Order By SalesYear, CustNo
```

GroupingYear	GroupingCustNo	TextNULL	SALESYEAR	CUSTNO	TOTAL
0	0		2008	10001	115,00
0	0		2008	10002	1350,00
0	0		2008	10003	535,00
0	0		2008	10004	470,00
0	0		2008	10005	310,00
0	1	Total Year	2008	-	2780,00
0	0		2009	10001	2634,20
0	0		2009	10002	1636,25
0	0		2009	10003	4589,86
0	0		2009	10004	2673,95
0	0		2009	10005	3741,95
0	1	Total Year	2009	-	15276,21
0	0		2010	10001	140,97
0	0		2010	10003	1555,75
0	0		2010	10006	9712,85
0	1	Total Year	2010	-	11409,57
1	1	Total	-	-	29465,78



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 92

Calculation



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 102

Calculation

```
select Distinct Amount,
       Amount * 0,95 A1,          0,95 * Amount A2,
       Amount * 95/100 B1,        95/100 * Amount B2,          Amount * (95/100) B3,
       95 * Amount/100 B4,        1/100 * Amount * 95 B5,
       Amount * (1 - 0,05) C1,    Amount * (1 - 5/100) C2,    (1 - 5/100) * Amount C3,
       Amount - Amount * 0,05 D1, Amount - Amount * 5/100 D2, Amount - 5/100 * Amount D3
From Sales
Order By Amount
;B
```

AMOUNT	A1	A2	B1	B2	B3	B4	B5	C1	C2	C3	D1	D2	D3
20,00	19,0000	19,0000	19,00000000000000	0,00	0,00	19,00000000000000	0,00	19,0000	20,00	20,00	19,0000	19,00000000000000	20,00
25,40	24,1300	24,1300	24,13000000000000	0,00	0,00	24,13000000000000	0,00	24,1300	25,40	25,40	24,1300	24,13000000000000	25,40
35,00	33,2500	33,2500	33,25000000000000	0,00	0,00	33,25000000000000	0,00	33,2500	35,00	35,00	33,2500	33,25000000000000	35,00
44,45	42,2275	42,2275	42,22750000000000	0,00	0,00	42,22750000000000	0,00	42,2275	44,45	44,45	42,2275	42,22750000000000	44,45
45,00	42,7500	42,7500	42,75000000000000	0,00	0,00	42,75000000000000	0,00	42,7500	45,00	45,00	42,7500	42,75000000000000	45,00
50,00	47,5000	47,5000	47,50000000000000	0,00	0,00	47,50000000000000	0,00	47,5000	50,00	50,00	47,5000	47,50000000000000	50,00
55,00	52,2500	52,2500	52,25000000000000	0,00	0,00	52,25000000000000	0,00	52,2500	55,00	55,00	52,2500	52,25000000000000	55,00
57,15	54,2925	54,2925	54,29250000000000	0,00	0,00	54,29250000000000	0,00	54,2925	57,15	57,15	54,2925	54,29250000000000	57,15
60,00	57,0000	57,0000	57,00000000000000	0,00	0,00	57,00000000000000	0,00	57,0000	60,00	60,00	57,0000	57,00000000000000	60,00
63,10	60,1150	60,1150	60,11500000000000	0,00	0,00	60,11500000000000	0,00	60,1150	63,10	63,10	60,1150	60,11500000000000	63,10
76,20	72,3900	72,3900	72,39000000000000	0,00	0,00	72,39000000000000	0,00	72,3900	76,20	76,20	72,3900	72,39000000000000	76,20
111,00	105,4500	105,4500	105,45000000000000	0,00	0,00	105,45000000000000	0,00	105,4500	111,00	111,00	105,4500	105,45000000000000	111,00
120,00	114,0000	114,0000	114,00000000000000	0,00	0,00	114,00000000000000	0,00	114,0000	120,00	120,00	114,0000	114,00000000000000	120,00
122,23	116,1185	116,1185	116,11850000000000	0,00	0,00	116,11850000000000	0,00	116,1185	122,23	122,23	116,1185	116,11850000000000	122,23
140,97	133,9215	133,9215	133,92150000000000	0,00	0,00	133,92150000000000	0,00	133,9215	140,97	140,97	133,9215	133,92150000000000	140,97
150,00	142,5000	142,5000	142,50000000000000	0,00	0,00	142,50000000000000	0,00	142,5000	150,00	150,00	142,5000	142,50000000000000	150,00
152,40	144,7800	144,7800	144,78000000000000	0,00	0,00	144,78000000000000	0,00	144,7800	152,40	152,40	144,7800	144,78000000000000	152,40
155,00	147,2500	147,2500	147,25000000000000	0,00	0,00	147,25000000000000	0,00	147,2500	155,00	155,00	147,2500	147,25000000000000	155,00
160,00	152,0000	152,0000	152,00000000000000	0,00	0,00	152,00000000000000	0,00	152,0000	160,00	160,00	152,0000	152,00000000000000	160,00
180,00	171,0000	171,0000	171,00000000000000	0,00	0,00	171,00000000000000	0,00	171,0000	180,00	180,00	171,0000	171,00000000000000	180,00
190,50	180,9750	180,9750	180,97500000000000	0,00	0,00	180,97500000000000	0,00	180,9750	190,50	190,50	180,9750	180,97500000000000	190,50
196,85	187,0075	187,0075	187,00750000000000	0,00	0,00	187,00750000000000	0,00	187,0075	196,85	196,85	187,0075	187,00750000000000	196,85
200,00	190,0000	190,0000	190,00000000000000	0,00	0,00	190,00000000000000	0,00	190,0000	200,00	200,00	190,0000	190,00000000000000	200,00



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 103

Toolmaker
ADVANCED EFFICIENCY

Figure 1: Schematic representation of the experimental design. The figure is divided into three main sections: A (top), B (middle), and C (bottom). Section A shows a timeline of 100 trials, with trials 1-20 in blue, 21-40 in green, 41-60 in red, and 61-80 in yellow. Section B shows a timeline of 100 trials, with trials 1-20 in blue, 21-40 in green, 41-60 in red, and 61-80 in yellow. Section C shows a timeline of 100 trials, with trials 1-20 in blue, 21-40 in green, 41-60 in red, and 61-80 in yellow.

Seite 104

Toolmaker
ADVANCED EFFICIENCY

```
select Distinct Amount,
       Amount * 0,95 A1,
       Amount * 95/100,0 B1,
       95,0 * Amount/100 B4,
       Amount * (1 - 0,05) C1,
       Amount - Amount * 0,05 D1,
       0,95 * Amount A2,
       95/100,0 * Amount B2,
       1,100,0 * Amount * 95 B5,
       Amount * (1 - 5/100,0) C2,
       Amount - Amount * 5/100,0 D2,
       Amount * (95/100,0) B3,
       (1 - 5/100,0) * Amount C3,
       Amount - 5/100,0 * Amount D3
From Sales
Order By Amount
;;
```

- Seite 105

Any Questions ?



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 110

Special Thanks to

Deon (Kit) von Blerk

- For proof-reading/correcting the English translation of these presentations

■ www.changefit.com

changeFIT™
Change Management Systems for IT

Holger Scherer

- For providing an AS/400 (IBM i-System) enabling the creation of the samples/code used in these presentations

■ www.rzkh.de



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 111

References



IBM i information center

■ SQL Reference

<http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/index.jsp?topic=%2Fdb2%2Frbafzintro.htm>

IBM developer works IBM i

<http://www.ibm.com/developerworks/ibmi/>

■ Hierarchical Queries with DB2 Connect By

<http://www.ibm.com/developerworks/ibmi/library/i-db2connectby/index.html>



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 112

Speaker's Biography



Birgitta Hauser has been a Software Engineer since 2008, focusing on RPG, SQL and Web development on IBM i at Toolmaker Advanced Efficiency GmbH in Germany.

She graduated with a business economics diploma, and started programming on the AS/400 in 1992. She also works in consulting and 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 and USA.

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 a German magazine and IBM DeveloperWorks.



23.04.2012

Do you know what you can do with a simple SQL Query? - Birgitta Hauser

Seite 113



May 6 - 9, 2012
Disneyland® Resort
Anaheim, CA



Powerful SQL Queries? Yes i can!
Thank you!



Birgitta Hauser

bha@toolmaker.de

Hauser@SSS-Software.de

Toolmaker Advanced Efficiency GmbH

Westendstraße 34

86916 Kaufering

www.toolmaker.de

Tel. (+49) 08191 968-0

Toolmaker Software AG

Obstgartenstr. 28

8136 Gattikon

www.toolmaker.ch

Tel. (+41) 043 3057323

Do you know what you can do with a simple SQL Query? - Birgitta Hauser