



IBM Systems

DB2 for i SQL OLAP Functions

Mike Cain
DB2 for i Center of Excellence
Rochester, MN USA
mcain@us.ibm.com

Scandinavian
Developer
Conference 2010

OLAP

On Line Analytical Processing

Think: building and delivering information via SQL

Review of the SQL OLAP functions

- RANK
 - DENSE_RANK
 - ROW_NUMBER
- } V5R4
- ROLLUP
 - CUBE
 - GROUPING SETS
 - GROUPING
- } 6.1

RANK and DENSE_RANK

- Specifies that the ordinal rank of a row within the window is computed
- Rows that are not distinct with respect to the ordering within their window are assigned the same rank
- The results of ranking may be defined with or without gaps in the numbers resulting from duplicate values
- **RANK** specifies that the rank of a row is defined as 1 plus the number of rows that strictly precede the row
 - Thus, if two or more rows are not distinct with respect to the ordering, then there will be one or more gaps in the sequential rank numbering
- **DENSE_RANK** specifies that the rank of a row is defined as 1 plus the number of preceding rows that are distinct with respect to the ordering
 - Thus, there will be no gaps in the sequential rank numbering

RANK and DENSE_RANK

- RANK and DENSE_RANK for highlighting data attribute – independent of the result set sorting

```

SELECT empno, lastname, salary+bonus AS TOTAL_SALARY,
RANK() OVER (ORDER BY salary+bonus DESC) AS Salary_Rank
FROM employee
WHERE salary + bonus > 30000
ORDER BY lastname
    
```

EMPNO	LASTNAME	TOTAL_SALARY	SALARY_RANK
000050	GEYER	40975.00	5
000010	HAAS	53750.00	1
200010	HEMMINGER	47500.00	2
000090	HENDERSON	30350.00	11
200220	JOHN	30440.00	9
000030	KWAN	39050.00	6
000110	LUCCHESI	47400.00	3
000220	LUTZ	30440.00	9
000070	PULASKI	36870.00	7
000060	STERN	32750.00	8
000020	THOMPSON	42050.00	4

**Dense_Rank()
Output**

SALARY_RANK
5
1
2
10
9
6
3
9
7
8
4

THE NEW POWER EQUATION |

ROW_NUMBER

- Specifies that a sequential row number is computed for the row within the window defined by the ordering, starting with 1 for the first row
- If the ORDER BY clause is not specified in the window, the row numbers are assigned to the rows in arbitrary order, as returned by the subselect
 - I.e. not according to any ORDER BY clause in the select-statement

ROW_NUMBER

- ROW_NUMBER can be used to arbitrarily assign a number to query results

```
SELECT ROW_NUMBER() OVER (ORDER BY workdept, lastname) AS Nbr,
       lastname, salary
FROM employee
ORDER BY workdept, lastname
```

NBR	LASTNAME	SALARY
1	HAAS	52750.00
2	HEMMINGER	46500.00
3	LUCCHESSI	46500.00
4	O'CONNELL	29250.00
5	ORLANDO	29250.00

```
SELECT workdept, lastname, hiredate,
       ROW_NUMBER() OVER (PARTITION BY workdept ORDER BY hiredate) AS Nbr
FROM employee
ORDER BY workdept, hiredate
```

WORKDEPT	LASTNAME	HIREDATE	NBR
A00	LUCCHESSI	1958-05-16	1
A00	O'CONNELL	1963-12-05	2
A00	HAAS	1965-01-01	3
A00	HEMMINGER	1965-01-01	4
A00	ORLANDO	1972-05-05	5
B01	THOMPSON	1973-10-10	1
C01	QUINTANA	1971-07-28	1
C01	KWAN	1975-04-05	2

THE NEW POWER EQUATION |

ROW_NUMBER

/* Below is an example of an SQL statement using the ROW_NUMBER support. */
 /* Note this particular technique works best when a table contains a single column primary key. */

```
WITH cte_row_number as (
  SELECT emp_mast_pk,
         ROW_NUMBER ()
           OVER (ORDER BY lastname, firstnme, midinit)
           AS cte_rrn
  FROM emp_mast
  WHERE workdept = ?
)
SELECT empno, lastname, firstnme, midinit
FROM emp_mast
JOIN cte_row_number USING (emp_mast_pk)
WHERE cte_rrn BETWEEN ? AND ?
ORDER BY cte_rrn;
```

*Common Table Expression
(emp_mast_pk, cte_rrn)*

Final query to return a range of results

/* The dynamic statement uses the ROW_NUMBER function to assign a row number to the result set of the derived table **cte_row_number**. */
 /* The OVER clause defines the order of the data. */
 /* The outer query filters based on the assigned row number. */
 /* The final result set is ordered by the generated row number (cte_rrn) which is in ascending sequence based on the ROW_NUMBER OVER clause. */

Grouping: What's in a name?

- Industry refers to Grouping Sets and Super Groups (Rollup and Cube) collectively as '**Grouping Sets**'
- Grouping Sets and Super Groups are sometimes referred to as '**Analytical SQL Grouping Functions**'
- The phrase '**Grouping Functions**' can be used to collectively describing Grouping Sets, Rollups, and Cubes

Grouping Sets and Super Groups ROLLUP and CUBE

- Many BI applications and OLAP tools involve hierarchical, multi-dimensional aggregate views of transaction data
 - Users need to view results at multiple levels
 - Users need to view result data from different perspective
 - Current grouping support only allows aggregation data of along a SINGLE dimension

EXAMPLE: *SELECT country, region, store, product, SUM(sales)*
FROM trans
GROUP BY country, region, store, product

1 level

- Limitations result in extra coding for programmers
- The **6.1** grouping and OLAP capabilities allow data to be grouped in multiple ways with a single SQL request
 - ROLLUP
 - CUBE
 - GROUPING SETS

**Less Coding
for Developers!**

THE NEW POWER EQUATION |

ROLLUP

- An extension to the GROUP BY clause that produces a result set containing sub-total rows in addition to the "regular" grouped rows
- Sub-total rows are "super-aggregate" rows that contain further aggregates whose values are derived by applying the same column functions that were used to obtain the grouped rows
- ROLLUP on the GROUP BY clause results in DB2 returning aggregates for each level of the hierarchy implicitly represented in the grouping columns

ROLLUP

- ROLLUP(Country, Region) will result in the data being summarized at the following levels
 - (Country, Region)
 - (Country)
 - () << represents Grand Total

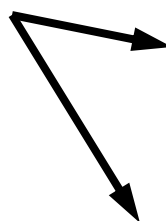
- **Example Query:**

```
SELECT country, region, SUM(sales)  
FROM trans  
GROUP BY ROLLUP (country, region)
```

ROLLUP Output Example

```
SELECT country, region, SUM(sales)
FROM trans
GROUP BY ROLLUP (country, region)
```

*GROUP BY
country, NULL*



Country	Region	Sum(Sales)
Canada	-	100,000
Canada	NW	100,000
U.S.A.	-	3,250,000
U.S.A.	NE	450,000
U.S.A.	NW	940,000
U.S.A.	SE	550,000
U.S.A.	SW	1,310,000
-	-	3,350,000

*GROUP BY
NULL, NULL*



THE NEW POWER EQUATION |

CUBE

- An extension to the GROUP BY clause that produces a result set that contains all the rows of a ROLLUP aggregation, plus contains "cross-tabulation" rows
- Cross-tabulation rows are additional "super-aggregate" rows that are not part of an aggregation with sub-totals
- CUBE on the GROUP BY clause results in DB2 returning aggregates for all possible distinct combinations represented by the grouping columns

CUBE

- CUBE(Country, Region) will result in the data being summarized at the following levels
 - (Country, Region)
 - (Country)
 - (Region)
 - () << represents Grand Total
- Returns results at multiple intersection points
- **Example Query:**
SELECT country, region, SUM(sales)
FROM trans
GROUP BY CUBE(country, region)

CUBE Output Example

```
SELECT country,region, SUM(sales)
FROM trans
GROUP BY CUBE (country, region)
```

GROUP BY NULL, region



GROUP BY NULL, NULL



GROUP BY country, NULL



Country	Region	Sum(Sales)
-	NE	450000
-	NW	1040000
-	SE	550000
-	SW	1310000
-	-	3350000
Canada	-	100000
U.S.A.	-	3250000
Canada	NW	100000
U.S.A.	NE	450000
U.S.A.	NW	940000
U.S.A.	SE	550000
U.S.A.	SW	1310000

THE NEW POWER EQUATION

GROUPING SETS

- Allows multiple grouping clauses to be specified in a single statement
- This can be thought of as the union of two or more groups of rows into a single result set
- GROUPING SET on the GROUP BY clause enables DB2 to return aggregates for multiple sets of grouping columns

GROUPING SETS

- GROUPING SETS((Country, Region), (Country, Store)) will result in the data being summarized at the following levels
 - (Country, Region)
 - (Country, Store)
- CUBE and ROLLUP can be used in combination with Grouping Sets

CAUTION: These types of combinations can result in an exponential growth in the number of grouping sets returned by a query, combine carefully

- **Example Query:**

```
SELECT country, region, SUM(sales)
FROM trans
GROUP BY GROUPING SETS((country, region), (country, store))
```

GROUPING SETS Output Example

```
SELECT country, region, store, SUM(sales)
FROM trans
GROUP BY GROUPING SETS ((country, region), (country, store))
```

GROUP BY
COUNTRY, REGION

Country	Region	Store	Sum(Sales)
Canada	NW	-	100,000
U.S.A.	NE	-	450,000
U.S.A.	NW	-	940,000
U.S.A.	SE	-	550,000
U.S.A.	SW	-	1,310,000
Canada	-	Dougs	100,000
U.S.A.	-	Mariahs	350,000
U.S.A.	-	KMs	770,000
U.S.A.	-	Jennas	400,000
U.S.A.	-	Adrians	500,000
U.S.A.	-	Joshs	300,000
U.S.A.	-	TZs	200,000
U.S.A.	-	Maddies	210,000

GROUP BY
COUNTRY, STORE

THE NEW POWER EQUATION

GROUPING

- The GROUPING function can be used to determine if null values are from underlying user data or DB2 aggregate processing
 - Function returns **1** if grouping column contains NULL value produced by grouping set or super group processing
 - Function returns **0** if grouping column contains “real” GROUP BY value

EXAMPLE: *SELECT country,region, store, GROUPING(store), SUM(sales)
FROM trans
WHERE transYear = 2006
GROUP BY GROUPING SETS ((country, region),(country, store))*

Multiplicative vs. Additive

- CUBE and ROLLUP can appear as part of a '*GROUP BY*' or '*GROUPING SETS*' list
- When CUBE and ROLLUP expressions are combined in one set, they operate like multipliers, forming additional set entries according to the definition of CUBE, ROLLUP
- When the CUBE or ROLLUP are separate sets, they operate in an additive fashion
- The use of parenthesis and the *GROUPING SETS* vs. *GROUP BY* clause will dictate how CUBE and ROLLUP expressions interact with each other

Multiplicative vs. Additive Examples

Set 1

Set 2

Set 3

EXAMPLE 1:

GROUP BY GROUPING SETS ((A, B, C, D), (A, B, C), (A, B))

- Additive grouping sets. *GROUPING SETS* without ROLLUP or CUBE are always additive. The parenthesis in red are required

EXAMPLE 2:

GROUP BY GROUPING SETS (CUBE(A,B), ROLLUP(C,D), E)

- *GROUPING SETS* clause used in both. One set of parenthesis around all sets specifies an additive association. Parenthesizing each set individually also specifies an additive association
- *GROUPING SETS* clause with single parenthesis is Additive
- CUBE(A,B) expands to *GROUPING SETS ((A,B), (A), (B), ())*
- ROLLUP(C,D) expands to *GROUPING SETS ((C,D), (C), ())*
- *GROUP BY* occurs in an additive fashion to form:
 - *GROUPING SETS ((A, B), (A), (B), (), (C, D), (C), (), (E))*

Multiplicative vs. Additive Examples

EXAMPLE 3:

GROUP BY A, B, ROLLUP(C,D)

Or

GROUP BY GROUPING SETS ((A, B, ROLLUP(C,D)))

- Parenthesis in red specifies that those elements are multiplicative in nature if a ROLLUP or CUBE exists.
- **GROUPING SETS** clause with double parenthesis is multiplicative
- **GROUP BY** clause with **ROLLUP** or **CUBE** is multiplicative
- ROLLUP(C,D) expands to GROUPING SETS ((C,D), (C), ())
- GROUP BY occurs in a multiplicative fashion to form:
 - GROUP BY GROUPING SETS ((a, b, c, d), (a, b, c), (a, b))

Multiplicative vs. Additive Examples

EXAMPLE 4:

GROUP BY CUBE(A, B), ROLLUP(C, D), E

OR

GROUP BY GROUPING SETS ((CUBE(A,B), ROLLUP(C,D), E))

- Parenthesis in red specifies that those elements are multiplicative in nature if a ROLLUP or CUBE exists
- CUBE(A,B) expands to GROUPING SETS ((A,B), (A), (B), ())
- ROLLUP(C,D) expands to GROUPING SETS ((C,D), (C), ())
- GROUP BY occurs in a multiplicative fashion to form:
 - *GROUPING SETS ((A, B, C, D, E), (A, B, C, E), (A, B, E), (A, C, D, E), (A, C, E), (A, E), (B, C, D, E), (B, C, E), (B, E), (C, D, E), (C, E), (E))*

12 Sets of results!

THE NEW POWER EQUATION |

Grouping Sets and Super Groups Considerations

- The new grouping technology can result in an exponential growth in the number of results returned by a query
- Performance Considerations
 - SQE query optimizer contains *patented and unique technology* allowing DB2 for i to internally compute multiple aggregates in single pass of data
 - Assist the optimizer by creating indexes that cover all of the grouping columns in addition to any local, equal selection predicates
 - Best Index keys for previous sample query
(transYear, country, region, store)
 - Index Advisor enhanced to support new grouping capabilities too!
- Do more with SQL, and do it faster and more efficiently!
- Only available in 6.1, and when using SQL queries via SQE

Performance Tuning Opportunities

- Optimal access method(s) in the feeder nodes
 - Sound Indexing and statistics Strategy
 - Create radix indexes for common Grouping Set and ROLLUP sets
 - Run Visual Explain to view query rewrites
 - Use index advised to find holes in indexing strategy and temporary index creates
- Sorted Aggregate Hash Table Creations
 - Could be replaced by a permanent index if index encapsulates predicate selection, grouping and aggregation columns
 - SMP / DB Parallelism can greatly improve populate time of the hash table
- Temporary Index Creations
 - Could be replaced by a permanent index if index encapsulates predicate selection, grouping and aggregation columns
 - Can imply that the environment is memory constrained
- Memory Constraints
 - Grouping function queries are generally more memory intensive as they often union multiple trees that create and interrogate temporary objects
 - Use feedback tools to determine the Optimizer's fair share of memory

Grouping Set Index Advised Example

*SELECT REGION, COUNTRY, COUNT(TERRITORY) FROM CUST_DIM WHERE
CONTINENT = 'AMERICA'
GROUP BY GROUPING SETS ((REGION),(COUNTRY))*

Visual Explain - Lp02ut15.rchland.ibm.com(Lp02ut15)

File View Actions Options Help

Attribute	Value
Share of Memory Available(bytes)	433,175,200
Memory Constrained	No
Cumulative Memory Constrained	No
Actual Runtime Information	
Actual Rows Selected Per Plan ...	29,952
Actual Rows Processed Per Pla...	150,000
Actual Plan Step Iterations	1
Actual Total Rows Selected	29,952
Actual Total Rows Processed	150,000
Information About the Plan Perf...	
Contains Predicate	Yes
Scrollable	Yes
Plan Name	Table Scan
Plan Step Type	Logic
Reason Code	Table Scan Cost Is Better
Plan Step Name	Node 34
Statement Text	SELECT CUST_DIM_1.REGION, CUST_DIM_1.COUNTRY, CUST_DIM_1.TERRITORY FROM STAR1G/CUST_DIM CUST_DIM_1 WHERE CUST_DIM_1.CONTINENT='AMERICA'

SELECT REGION, COUNTRY, COUNT(TERRITORY) FROM CUST_DIM WHERE CONTINENT = 'AMERICA' GROUP BY GROUPING SETS ((REGION),(COUNTRY))
OPTIMIZE FOR ALL ROWS

Statement text

Grouping Set Index Advised Example

Index and Statistics Advisor - Lp02ut15.rchland.ibm.com(Lp02ut15)

Index Advisor | Statistics Advisor

It is recommended that the following indexes be created:

Create	Table Name	Schema	Index Type	Columns
<input checked="" type="checkbox"/>	CUST_DIM	STAR1G	Binary Radix	CONTINENT REGION
<input checked="" type="checkbox"/>	CUST_DIM	STAR1G	Binary Radix	CONTINENT COUNTRY

Create ...

OK Help ?

Grouping Set Index Advised Example

Visual Explain - Lp02ut15.rchland.ibm.com(Lp02ut15)

File View Actions Options Help

Attribute	Value
Time Information	
Timestamp for Creation of Monit...	2007-07-02-15.23.35.880875
Statement Start Timestamp	2007-07-02-15.23.35.880875
Statement End Timestamp	2007-07-02-15.23.35.880875
Total Estimated Run Time (ms)	1,345
Actual Runtime Information	
Optimization Time (ms)	12
Run Time (ms)	83
Statement Open Time (ms)	Not Available
Statement Fetch Time (ms)	83
Statement Close Time (ms)	Not Available
Rows Fetched	10
Total Times Query Was Run	1
Total Time For All Runs (ms)	84
Synchronous Database Reads	1
Asynchronous Database Reads	0
Page Faults	1
Information about SQL stateme...	
Statement Number	24,948
Statement Function	Select
Statement Operation	Open
Statement Type	Dynamic
Statement Name	STMT0017
Statement Outcome	Successful
SQL Return Code	0
SQLSTATE	00000
Cursor Name	CRSR0017
Program or Package Name	
Program or Package Library	
Statement Text	SELECT REGION, COUNTRY, COUNT(TERRITORY) FROM CUST_DIM WHERE CONTINENT = 'AMERICA' GROUP BY GROUPING SETS ((REGION),(COUNTRY)) OPTIMIZE FOR ALL ROWS

Statement text

THE NEW POWER EQUATION |

Grouping Set Index Advised Example

SELECT REGION, COUNTRY, COUNT(TERRITORY) FROM CUST_DIM GROUP BY GROUPING SETS ((REGION),(COUNTRY))

The screenshot shows the Visual Explain interface for the query: `SELECT REGION, COUNTRY, COUNT(TERRITORY) FROM CUST_DIM GROUP BY GROUPING SETS ((REGION),(COUNTRY)) OPTIMIZE FOR ALL ROWS`. The plan diagram illustrates a 'Table Scan' of the 'STAR1G.CUST_DIM' table (150,000 rows) feeding into two 'Hash Scan' operations, each using a 'Temporary Distinct Hash Table'. These feeds into a 'Union all' operation, which then leads to the 'Final Select' step.

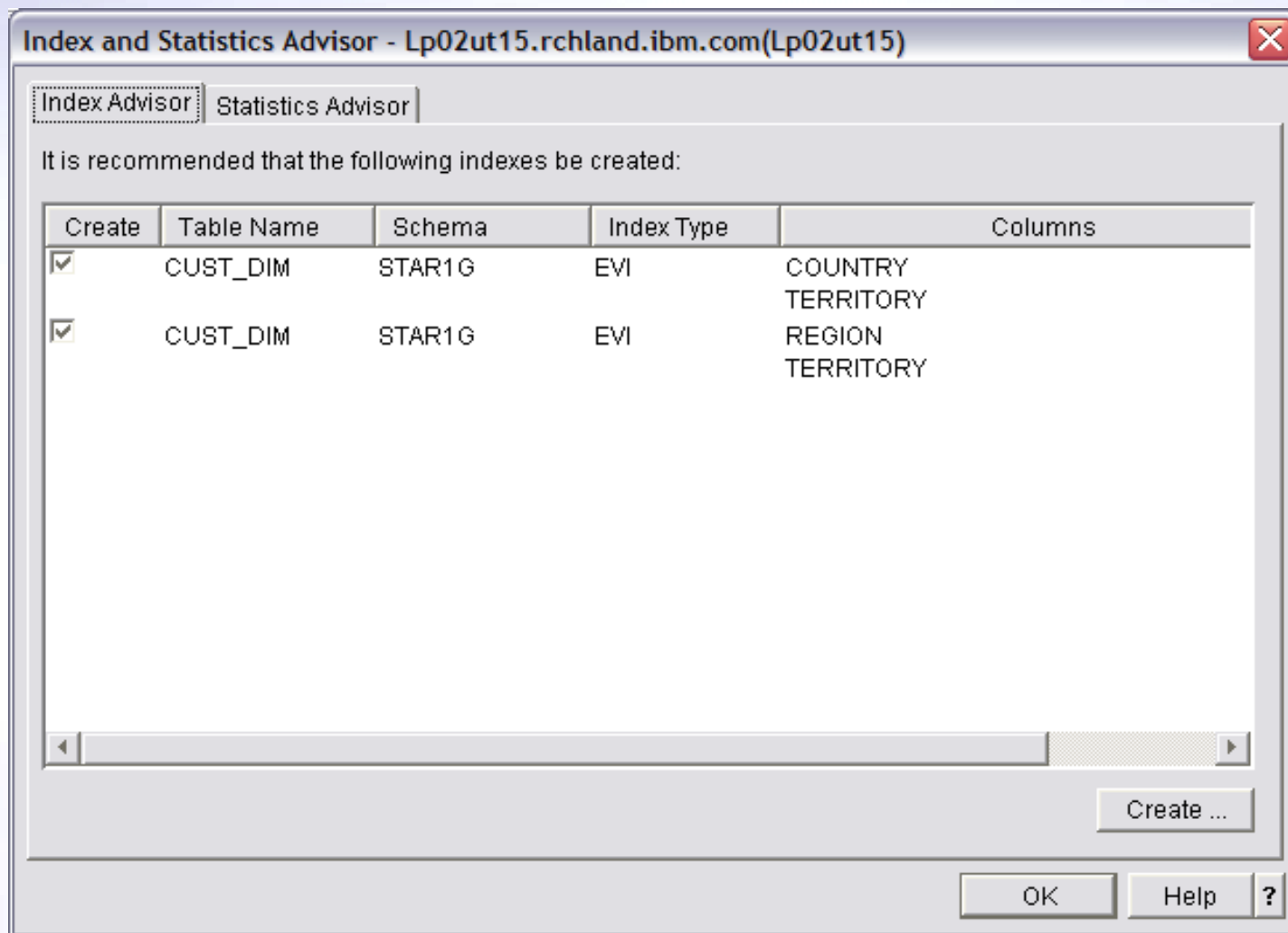
Attribute	Value
CPU Cost(ms)	27.954
I/O Cost(ms)	1,270
I/O Count	254
PreLoad Relation	No
Memory Used(bytes)	33,150,992
Share of Memory Available(bytes)	433,175,200
Memory Constrained	No
Cumulative Memory Constrained	No

Actual Runtime Information	
Actual Rows Selected Per Plan ...	150,000
Actual Rows Processed Per Pla...	150,000
Actual Plan Step Iterations	1
Actual Total Rows Selected	150,000
Actual Total Rows Processed	150,000

Information About the Plan Perf...	
Scrollable	Yes
Plan Name	Table Scan
Plan Step Type	Logic
Reason Code	No Indexes Exist
Plan Step Name	Node 13
Statement Text	SELECT CUST_DIM_1.REGION, CUST_DIM_1.COUNTRY, CUST_DIM_1.TERRITORY FROM STAR1G/CUST_DIM CUST_DIM_1

Statement text: `SELECT REGION, COUNTRY, COUNT(TERRITORY) FROM CUST_DIM GROUP BY GROUPING SETS ((REGION),(COUNTRY)) OPTIMIZE FOR ALL ROWS`

Grouping Set Index Advised Example



Grouping Set Index Advised Example

Visual Explain - Lp02ut15.rchland.ibm.com(Lp02ut15)

File View Actions Options Help

Attribute	Value
Time Information	
Timestamp for Creation of Monit...	2007-07-02-15.57.26.484308
Statement Start Timestamp	2007-07-02-15.57.26.484308
Statement End Timestamp	2007-07-02-15.57.26.484308
Total Estimated Run Time (ms)	.053
Actual Runtime Information	
Optimization Time (ms)	10
Run Time (ms)	0
Statement Open Time (ms)	Not Available
Statement Fetch Time (ms)	0
Statement Close Time (ms)	Not Available
Rows Fetched	30
Total Times Query Was Run	1
Total Time For All Runs (ms)	1
Synchronous Database Reads	0
Asynchronous Database Reads	0
Page Faults	0
Information about SQL stateme...	
Statement Number	24,978
Statement Function	Select
Statement Operation	Open
Statement Type	Dynamic
Statement Name	STMT0035

SELECT REGION, COUNTRY, COUNT(TERRITORY) FROM CUST_DIM GROUP BY GROUPING SETS ((REGION),(COUNTRY)) OPTIMIZE FOR ALL ROWS

Statement text

Q & A

Thank You