(T32)比較GroupingSet、Rollup、Cube。比較Grouping Function、Grouping、IDFunction  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T32)比較GroupingSet、Rollup、Cube。比較Grouping Function、Grouping、IDFunction  
=======================================================================  
0. Summary

-----------

1. GroupingSets

1.1. Create Sample data

1.2. Group By ... Union All...

1.3. Group BY GROUPING SETS ...

1.4. Clean up

-----------

2. Rollup

2.1. Create Sample data

2.2. GROUP BY ROLLUP(C1,C2,...)

2.3. GROUP BY ROLLUP(C1,C2,...)

2.4. Clean up

-----------

3. Cube

3.1. Create Sample data

3.2. GROUP BY CUBE (C1, C2, ..., Cn-1, Cn)

3.3. Clean up

-----------

4. Cube V.S. Rollup

4.1. Create Sample data

4.2. GROUP BY ROLLUP(C1,C2,...)

4.3. GROUP BY CUBE(C1,C2,...)

4.4. ROLLUP and CUBE on a single column is no different.

4.5. Clean up

-----------

5. GroupingFunction

5.1. Create Sample data

5.2. Grouping(columnA)

5.3. Grouping(columnA)

5.4. Grouping(columnA)

5.5. Clean up

-----------

6. Grouping\_IDFunction

=======================================================================  
  
0. Summary

1.

GROUPING SETS   V.S.   ROLLUP  V.S.  CUBE

Reference:

[https://technet.microsoft.com/zh-cn/library/bb522495(v=sql.105).aspx](https://technet.microsoft.com/zh-cn/library/bb522495%28v=sql.105%29.aspx)

[https://technet.microsoft.com/zh-cn/library/bb522631(v=sql.105).aspx](https://technet.microsoft.com/zh-cn/library/bb522631%28v=sql.105%29.aspx)

[https://technet.microsoft.com/zh-cn/library/bb510427(v=sql.105).aspx](https://technet.microsoft.com/zh-cn/library/bb510427%28v=sql.105%29.aspx)

[https://technet.microsoft.com/en-us/library/bb522495(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/bb522495%28v=sql.105%29.aspx)

[https://technet.microsoft.com/en-us/library/bb510427(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/bb510427%28v=sql.105%29.aspx)

1.1.

--GROUP BY GROUPING SETS (

--    (ColumnA, ColumnB, ...), --GroupingSet1

--    (ColumnB ...), --GroupingSet2

--    ...

--)

GROUP BY GROUPING SETS do aggregate operation

and UNION ALL all other aggregate operation.

1.2.

--GROUP BY ROLLUP (C1, C2, ..., Cn-1, Cn)

or

--GROUP BY C1, C2, ..., Cn-1, Cn WITH ROLLUP

ROLLUP do aggregate operation on multiple levels in hierarchy.

1.3.

--GROUP BY CUBE (C1, C2, ..., Cn-1, Cn)

or

--GROUP BY C1, C2, ..., Cn-1, Cn WITH CUBE

CUBE produces the result set

by generating all combinations of columns

specified in GROUP BY CUBE().

---------------------------------------------------------------------

2.

Syntax

--GROUP BY GROUPING SETS (

--    (ColumnA, ColumnB, ...), --GroupingSet1

--    (ColumnB ...), --GroupingSet2

--    ...

--)

2.1.

This is like

--Group By (ColumnA, ColumnB, ...)

--Union ALL

--Group By (ColumnB, ...)

--Union ALL ...

Problem about Union ALL is that

if you Union ALL 4 SELECT ... GROUP BY ...

then the table need to be accessed 4 times.

IF you use GROUP BY GROUPING SETS,

then the table need to be accessed 1 time.

Thus, GROUPING SETS is faster than UNION ALL 4 times of SELECT ... GROUP BY ...

2.2.

--ORDER BY GROUPING(ColumnA);

or

----ORDER BY ColumnA;

The order of the rows of GROUP BY GROUPING SETS in the result set

is not the same as UNION ALL query.

We can use Order By with GROUP BY GROUPING SETS to control the order.

However, we cannot use use Order By with Union ALL.

To control the order use order by as shown below.

---------------------------------------------------------------------

3.

3.1.

The following clauses return the same grand totals:

3.1.1.

--GROUP BY GROUPING SETS ( () )

3.1.2.

--GROUP BY ()

-------------------------------

3.2.

The following clauses return the same single sets

3.2.1.

--GROUP BY GROUPING SETS ( (C1, C2, ..., Cn) )

3.2.2.

--GROUP BY C1, C2, ..., Cn

-------------------------------

3.3.

The following clauses are equivalent:

3.3.1.

--GROUP BY ROLLUP (C1, C2, ..., Cn-1, Cn)

3.3.2.

--GROUP BY C1, C2, ..., Cn-1, Cn WITH ROLLUP

3.3.3.

--GROUP BY GROUPING SETS ( (C1, C2, ..., Cn-1, Cn)

--    ,(C1, C2, ..., Cn-1)

--    ...

--    ,(C1, C2)

--    ,(C1)

--    ,() )

3.3.4.

--SELECT ...FROM...GROUP BY C1, C2, ..., Cn-1, Cn

--UNION ALL

--SELECT ...FROM...GROUP BY C1, C2, ..., Cn-1

--SELECT ...FROM...UNION ALL

--...

--UNION ALL

--SELECT ...FROM...GROUP BY C1, C2

--UNION ALL

--SELECT ...FROM...GROUP BY C1

--UNION ALL

--SELECT ...FROM...

-------------------------------

3.4.

The following clauses are equivalent:

3.4.1.

--GROUP BY CUBE (C1, C2, C3)

3.4.2.

--GROUP BY GROUPING SETS (

--    (C1, C2, C3)

--    ,(C1, C2)

--    ,(C1, C3)

--    ,(C2, C3)

--    ,(C1)

--    ,(C2)

--    ,(C3)

--    ,() )

-------------------------------

3.5.

The following clauses are equivalent:

3.5.1.

--GROUP BY ROLLUP (C1, C2, C3)

3.5.2.

--GROUP BY C1, C2, C3 WITH ROLLUP

3.5.3.

--GROUP BY GROUPING SETS (

--    ,(C1, C2, C3)

--    ,(C1, C2)

--    ,(C1)

--    ,() )

-------------------------------

3.6.

The following clauses are equivalent:

3.6.1.

--GROUP BY ROLLUP(A, (C1, C2, ..., Cn) )

3.6.2.

--GROUP BY ROLLUP( (A), (C1, C2, ..., Cn) )

3.6.3.

--GROUP BY ( (A), (C1, C2, ..., Cn) ) WITH ROLLUP

3.6.3.

--GROUP BY GROUPING SETS (

--    (A, C1, C2, ..., Cn),

--    (A),

--    ()

--)

-------------------------------

3.7.

The following clauses are equivalent:

3.7.1.

--GROUP BY CUBE(A, (C1, C2, ..., Cn) )

3.7.2.

--GROUP BY CUBE( (A), (C1, C2, ..., Cn) )

3.7.3.

--GROUP BY GROUPING SETS (

--    (),

--    (A),

--    (C1, C2, ..., Cn),

--    (A, C1, C2, ..., Cn) )

-------------------------------

--\*\*

3.8.

The following clauses are equivalent:

3.8.1.

GROUP BY A, CUBE (B, C)

3.8.2.

GROUP BY GROUPING SETS (

    (A),

    (A, B),

    (A, C),

    (A, B, C ))

-------------------------------

--\*\*

3.9.

The following clauses are equivalent:

3.9.1.

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

3.9.2.

--GROUP BY GROUPING SETS (

--    (A, B),

--    (A, C) )

-------------------------------

--\*\*

3.10.

The following clauses are equivalent:

3.10.1.

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

3.10.2.

--GROUP BY GROUPING SETS (

--    (A),

--    (B,C),

--    (B),

--    ()

--)

-------------------------------

--\*\*

3.11.

The following clauses are equivalent:

3.11.1.

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

3.11.2.

--GROUP BY GROUPING SETS (

--    A,

--    B,

--    (B,C),

--    (B, C, D)

--    ()

--)

------------------------------------------------------------------------

4.

--Grouping(columnA)

Syntax

--SELECT  ColumnA ,

--        SUM(ColumnB) AS TotalB ,

--        GROUPING(ColumnA) AS 'GroupingColumnA'

--FROM    dbo.TableName

--GROUP BY ROLLUP(ColumnA);

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/grouping-transact-sql>

<https://docs.microsoft.com/zh-cn/sql/t-sql/functions/grouping-transact-sql>

4.1.

if the columnA in a GROUP BY list is aggregated(Count, Sum, Avg, Min, Max)

then Grouping(columnA) return 1, otherwise return 0.

4.2.

When using ROLLUP, CUBE or GROUPING SETS,

the NULL returned might be normal standard null values,

or the NULL returned might be a column placeholder and means all.

4.3.

If SELECT Grouping(columnA) for that row return 0,

it means columnA in a GROUP BY list for that row is normal standard null values.

4.4.

If SELECT Grouping(columnA) for that row return 1,

it means columnA in a GROUP BY list for that row is a column placeholder

from ROLLUP, CUBE or GROUPING SETS, and it means all.

4.5.

Grouping(columnA) can be used in the

SELECT <select> list,

HAVING, and

ORDER BY clauses

when GROUP BY is specified.

------------------------------------------------------------------------

5.

--Grouping(columnA)

E.g.

--SELECT  HouseType ,

--        SUM(SoldPrice) AS TotalSold ,

--        GROUPING(HouseType) AS 'GroupingHouseType'

--FROM    dbo.HouseSold

--GROUP BY ROLLUP(HouseType);

Output as following

--HouseType   TotalSold   GroupingHouseType

--NULL          493000.00        0

--Type1          1320000.00    0

--Type2          1400000.00    0

--NULL          3213000.00    1

The result set shows two NULL values under HouseType Column.

5.1.

--NULL    493000.00        0

The 1st NULL value under HouseType Column

means HouseType Column in a GROUP BY list for that row

is normal standard null values.

It represents the group of null values from the HouseType Column.

Thus, SELECT Grouping(HouseType) for that row will return 0,

5.2.

--NULL    3213000.00    1

The 2nd NULL value under HouseType Column

means HouseType Column in a GROUP BY list for that summary row is a column placeholder

from ROLLUP, CUBE or GROUPING SETS, and it means all.

Thus, SELECT Grouping(HouseType) for that row will return 1,

if the columnA in a GROUP BY list is aggregated(Count, Sum, Avg, Min, Max)

then Grouping(columnA) return 1, otherwise return 0.

------------------------------------------------------------------------

6.

--GROUPING\_ID(C1,C2,...Cn)

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/grouping-id-transact-sql>

<https://docs.microsoft.com/zh-cn/sql/t-sql/functions/grouping-id-transact-sql>

6.1.

Syntax

--SELECT  C1,C2,...Cn ,

--        SUM(ColumnB) AS TotalB ,

--        GROUPING\_ID(C1,C2,...Cn) AS 'GPID'

--FROM    dbo.TableName

--GROUP BY ROLLUP(C1,C2,...Cn);

6.1.1.

GROUPING\_ID(C1,C2,...Cn) function concatenates

all the GOUPING(C1), GOUPING(C2),...GOUPING(Cn) functions,

and then perform the binary string to decimal conversion.

6.1.2.

The column list of GROUPING\_ID(C1,C2,...Cn) must match

the column list of GROUP BY ROLLUP(C1,C2,...Cn).

6.1.3.

GROUPING\_ID(C1,C2,...Cn) function computes the level of grouping.

We normally use GROUPING\_ID(C1,C2,...Cn) in ORDER BY and HAVING clause to

order the ROLLUP or CUBE.

6.1.4.

GROUPING\_ID(C1,C2,...Cn) can be used in the

SELECT <select> list,

HAVING, and

ORDER BY clauses

when GROUP BY is specified.

This usage is same as Grouping(C1) function

6.2.

E.g.

--SELECT  C1,C2,C3,

--        SUM(ColumnB) AS TotalB ,

--        GROUPING\_ID(C1,C2,C3) AS 'GPID'

--FROM    dbo.TableName

--GROUP BY ROLLUP(C1,C2,C3);

GROUPING\_ID(C1,C2,C3) binary string =

CAST(GROUPING(C1) AS NVARCHAR(1)) +

CAST(GROUPING(C2) AS NVARCHAR(1)) +

CAST(GROUPING(C3) AS NVARCHAR(1));

GROUPING\_ID(C1,C2,C3) = convert GROUPING\_ID(C1,C2,C3)BinaryString to decimal.

Grouping(C1), Grouping(C2), or Grouping(C3) will return 1 or 0.

GROUPING\_ID(C1,C2,C3) function concatenates

all the GOUPING(C1), GOUPING(C2),GOUPING(C3) functions,

and then perform the binary string to decimal conversion.

------------------------------------------------------------------------

7.

Reference:

<http://improve.dk/converting-between-base-2-10-and-16-in-t-sql/>

----If function exists then DROP it

--IF ( EXISTS ( SELECT    \*

--              FROM      INFORMATION\_SCHEMA.ROUTINES

--              WHERE     ROUTINE\_TYPE = 'FUNCTION'

--                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

--                        AND SPECIFIC\_NAME = 'fnBinaryStrToDecimal' ) )

--    BEGIN

--        DROP FUNCTION fnBinaryStrToDecimal;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE FUNCTION [dbo].[fnBinaryStrToDecimal] ( @Input VARCHAR(255) )

--RETURNS BIGINT

--AS

--    BEGIN

--        DECLARE @Cnt TINYINT = 1;

--        DECLARE @Len TINYINT = LEN(@Input);

--        DECLARE @Output BIGINT = CAST(SUBSTRING(@Input, @Len, 1) AS BIGINT);

--        WHILE ( @Cnt < @Len )

--            BEGIN

--                SET @Output = @Output

--                    + POWER(CAST(SUBSTRING(@Input, @Len - @Cnt, 1) \* 2 AS BIGINT),

--                            @Cnt);

--                SET @Cnt = @Cnt + 1;

--            END;

--        RETURN @Output;

--    END;

--GO -- Run the previous command and begins new batch

--PRINT dbo.fnBinaryStrToDecimal('111')

=====================================================

1. GroupingSets

--========================================================================

--T032\_01\_GroupingSets

--========================================================================

1.1. Create Sample data

--========================================================================

--T032\_01\_01

--Create Sample data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSold

(

  Id INT IDENTITY(1, 1)

         PRIMARY KEY ,

  HouseStreetAddress NVARCHAR(100) ,

  HouseSuburb NVARCHAR(100) ,

  SoldPrice MONEY ,

  HouseType NVARCHAR(100)

);

GO -- Run the previous command and begins new batch

INSERT  dbo.HouseSold

VALUES  ( N'A1 Street', N'Suburb2', 400000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B1 Street', N'Suburb1', 500000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'C3 Street', N'Suburb1', 560000, N'Type3' );

INSERT  dbo.HouseSold

VALUES  ( N'D4 Street', N'Suburb2', 350000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'A5 Street', N'Suburb2', 440000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'A9 Street', N'Suburb3', 460000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B8 Street', N'Suburb3', 470000, N'Type3' );

INSERT  dbo.HouseSold

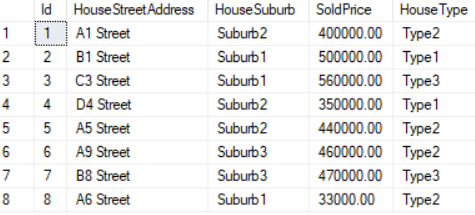
VALUES  ( N'A6 Street', N'Suburb1', 33000, N'Type2' );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.HouseSold;

GO -- Run the previous command and begins new batch



1.2. Group By ... Union All...

--========================================================================

--T032\_01\_02

--Group By ... Union All...

--------------------------------------------------------------------------

--T032\_01\_02\_01

--calculate Sum of SoldPrice by HouseSuburb and HouseType

SELECT  HouseSuburb ,

        HouseType ,

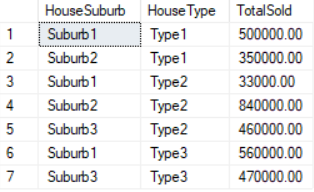
        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseSuburb ,

        HouseType;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_02\_02

--calculate Sum of SoldPrice by HouseSuburb and HouseType

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseSuburb ,

        HouseType

UNION ALL

--calculate Sum of SoldPrice by HouseSuburb

SELECT  HouseSuburb ,

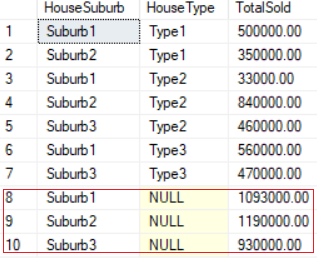
        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseSuburb;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_02\_03

--calculate Sum of SoldPrice by HouseSuburb and HouseType

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseSuburb ,

        HouseType

UNION ALL

--calculate Sum of SoldPrice by HouseSuburb

SELECT  HouseSuburb ,

        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseSuburb

UNION ALL

--calculate Sum of SoldPrice by HouseType

SELECT  NULL ,

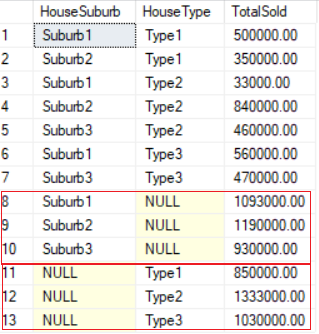
        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseType;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_02\_04

--calculate Sum of SoldPrice by HouseSuburb and HouseType

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseSuburb ,

        HouseType

UNION ALL

--calculate Sum of SoldPrice by HouseSuburb

SELECT  HouseSuburb ,

        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseSuburb

UNION ALL

--calculate Sum of SoldPrice by HouseType

SELECT  NULL ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY HouseType

UNION ALL

--calculate Sum of SoldPrice

SELECT  NULL ,

        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold;

--ORDER BY dbo.HouseSold.HouseSuburb, dbo.HouseSold.HouseType, dbo.HouseSold.SoldPrice

GO -- Run the previous command and begins new batch

/\*

1.

--ORDER BY dbo.HouseSold.HouseSuburb, dbo.HouseSold.HouseType, dbo.HouseSold.SoldPrice

Reference:

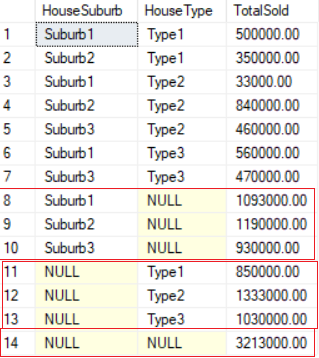
<https://stackoverflow.com/questions/27819047/group-by-and-order-by-with-union-all>

You can not use order by with Group by and Union All in this way.

However, please see the reference

if you really want to know how to use GROUP BY, UNION ALL, and ORDER BY all together.

\*/



1.3. Group BY GROUPING SETS ...

--========================================================================

--T032\_01\_03

--Group BY GROUPING SETS ...

--------------------------------------------------------------------------

--T032\_01\_03\_01

--Group BY GROUPING SETS ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

                       ( HouseType ), -- Sum of SoldPrice by HouseType

                       ( )-- Grand Total Sold

      );

GO -- Run the previous command and begins new batch

/\*

1.

Syntax

--GROUP BY GROUPING SETS (

--     (ColumnA, ColumnB, ...), --GroupingSet1

--     (ColumnB ...), --GroupingSet2

--     ...

--)

1.1.

This is like

--Group By (ColumnA, ColumnB, ...)

--Union ALL

--Group By (ColumnB, ...)

--Union ALL ...

Problem about Union ALL is that

if you Union ALL 4 SELECT ... GROUP BY ...

then the table need to be accessed 4 times.

IF you use GROUP BY GROUPING SETS,

then the table need to be accessed 1 time.

Thus, GROUPING SETS is faster than UNION ALL 4 times of SELECT ... GROUP BY ...

1.2.

We can use Order By with GROUP BY GROUPING SETS.

But we cannot use use Order By with Union ALL.

1.3.

The following clauses are equivalent:

E.g.1.

--GROUP BY GROUPING SETS(

--                       ( HouseSuburb ,

--                         HouseType

--                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

--                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

--                       ( HouseType ), -- Sum of SoldPrice by HouseType

--                       ( )-- Grand Total Sold

--      );

E.g.2.

--Group By (HouseSuburb, HouseType, ...)

--Union ALL

--Group By (HouseSuburb)

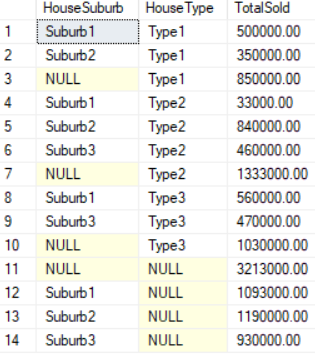
--Union ALL

--Group By (HouseType)

--Union ALL

--Group By ()

\*/



--------------------------------------------------------------------------

--T032\_01\_03\_02

--Group BY GROUPING SETS ...ORDER BY ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

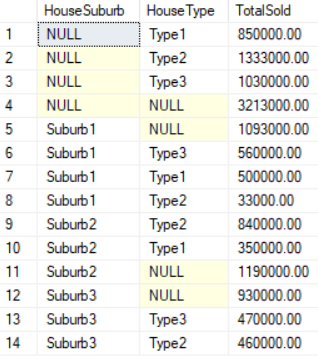
                       ( HouseType ), -- Sum of SoldPrice by HouseType

                       ( )-- Grand Total Sold

      )

ORDER BY HouseSuburb;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_03\_03

--Group BY GROUPING SETS ...ORDER BY ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

                       ( HouseType ), -- Sum of SoldPrice by HouseType

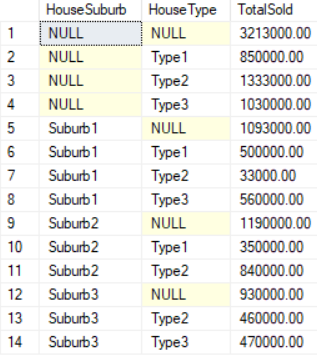
                       ( )-- Grand Total Sold

      )

ORDER BY HouseSuburb ,

        HouseType;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_03\_04

--Group BY GROUPING SETS ...ORDER BY ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

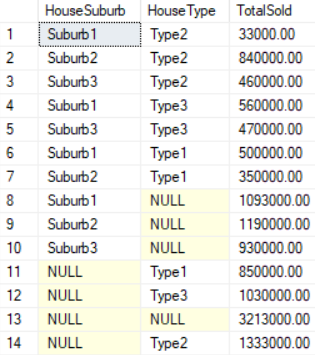
                       ( HouseType ), -- Sum of SoldPrice by HouseType

                       ( )-- Grand Total Sold

      )

ORDER BY GROUPING(HouseSuburb);

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_03\_05

--Group BY GROUPING SETS ...ORDER BY ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

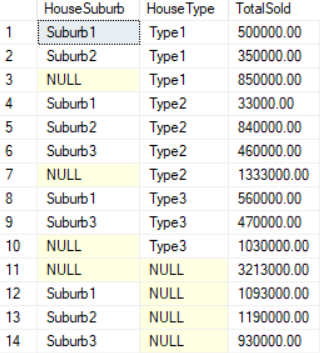
                       ( HouseType ), -- Sum of SoldPrice by HouseType

                       ( )-- Grand Total Sold

      )

ORDER BY GROUPING(HouseType);

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_03\_06

--Group BY GROUPING SETS ...ORDER BY ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

                       ( HouseType ), -- Sum of SoldPrice by HouseType

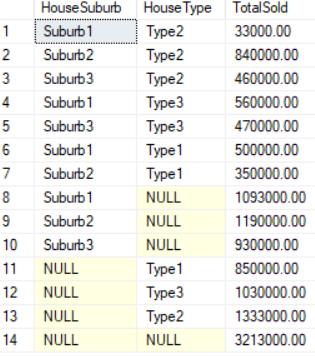
                       ( )-- Grand Total Sold

      )

ORDER BY GROUPING(HouseSuburb) ,

        GROUPING(HouseType);

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_01\_03\_07

--Group BY GROUPING SETS ...ORDER BY ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    dbo.HouseSold

GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), -- Sum of SoldPrice by HouseSuburb and HouseType

                       ( HouseSuburb ), -- Sum of SoldPrice by HouseSuburb

                       ( HouseType ), -- Sum of SoldPrice by HouseType

                       ( )-- Grand Total Sold

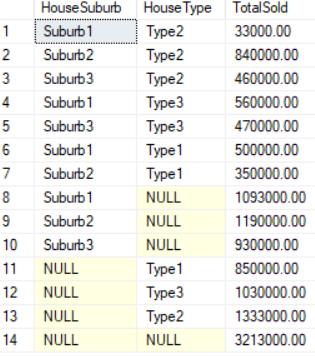
      )

ORDER BY GROUPING(HouseSuburb) ,

        GROUPING(HouseType) ,

        HouseType;

GO -- Run the previous command and begins new batch



1.4. Clean up

--========================================================================

--T032\_01\_04

--Clean up

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

2. Rollup

--========================================================================

--T032\_02\_Rollup

--========================================================================

/\*

1.

--GROUP BY ROLLUP (C1, C2, ..., Cn-1, Cn)

or

--GROUP BY C1, C2, ..., Cn-1, Cn WITH ROLLUP

ROLLUP do aggregate operation on multiple levels in hierarchy.

The following clauses are equivalent:

1.1.

--GROUP BY ROLLUP (C1, C2, C3)

1.2.

--GROUP BY C1, C2, C3 WITH ROLLUP

1.3.

--GROUP BY GROUPING SETS (

--    ,(C1, C2, C3)

--    ,(C1, C2)

--    ,(C1)

--    ,() )

\*/

2.1. Create Sample data

--========================================================================

--T032\_02\_01

--Create Sample data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSold

(

  Id INT IDENTITY(1, 1)

         PRIMARY KEY ,

  HouseStreetAddress NVARCHAR(100) ,

  HouseSuburb NVARCHAR(100) ,

  SoldPrice MONEY ,

  HouseType NVARCHAR(100)

);

GO -- Run the previous command and begins new batch

INSERT  dbo.HouseSold

VALUES  ( N'A1 Street', N'Suburb2', 400000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B1 Street', N'Suburb1', 500000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'C3 Street', N'Suburb1', 560000, N'Type3' );

INSERT  dbo.HouseSold

VALUES  ( N'D4 Street', N'Suburb2', 350000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'A5 Street', N'Suburb2', 440000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'A9 Street', N'Suburb3', 460000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B8 Street', N'Suburb3', 470000, N'Type3' );

INSERT  dbo.HouseSold

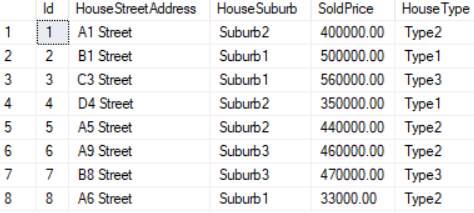
VALUES  ( N'A6 Street', N'Suburb1', 33000, N'Type2' );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.HouseSold;

GO -- Run the previous command and begins new batch



2.2. GROUP BY ROLLUP(C1,C2,...)

--========================================================================

--T032\_02\_02

--GROUP BY ROLLUP(C1,C2,...)

--The following clauses are equivalent.

--------------------------------------------------------------------------

--T032\_02\_02\_01

-- ... UNION ALL ...

SELECT  HouseSuburb ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY HouseSuburb

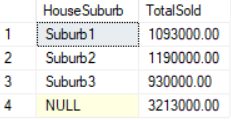
UNION ALL

SELECT  NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_02\_02\_02

-- GROUP BY GROUPING SETS (...)

SELECT  HouseSuburb ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY GROUPING SETS(( HouseSuburb ), ( ));

GO -- Run the previous command and begins new batch

Table

Description automatically generated

--------------------------------------------------------------------------

--T032\_02\_02\_03

-- GROUP BY ROLLUP(C1,C2,...)

SELECT  HouseSuburb ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY ROLLUP(HouseSuburb);

GO -- Run the previous command and begins new batch

Table

Description automatically generated

--------------------------------------------------------------------------

--T032\_02\_02\_04

-- GROUP BY (C1,C2...) WITH ROLLUP

SELECT  HouseSuburb ,

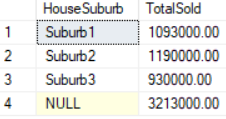
        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY HouseSuburb

        WITH ROLLUP;

GO -- Run the previous command and begins new batch



2.3. GROUP BY ROLLUP(C1,C2,...)

--========================================================================

--T032\_02\_03

--GROUP BY ROLLUP(C1,C2,...)

--The following clauses are equivalent.

--------------------------------------------------------------------------

--T032\_02\_03\_01

-- ... UNION ALL ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY HouseSuburb ,

        HouseType

UNION ALL

SELECT  HouseSuburb ,

        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY HouseSuburb

UNION ALL

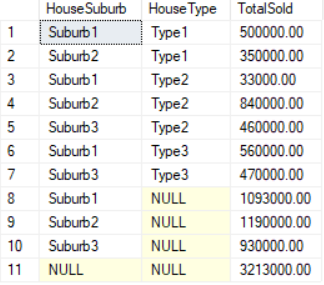
SELECT  NULL ,

        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_02\_03\_02

-- GROUP BY GROUPING SETS (...)

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

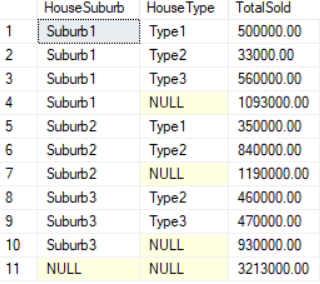
GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), ( HouseSuburb ), ( ));

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_02\_03\_03

-- GROUP BY ROLLUP(C1,C2,...)

SELECT  HouseSuburb ,

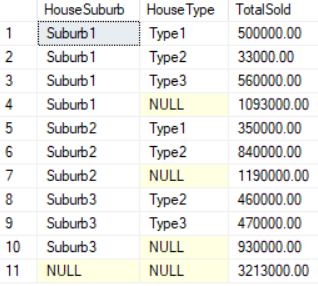
        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY ROLLUP(HouseSuburb, HouseType);

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_02\_03\_04

-- GROUP BY (C1,C2...) WITH ROLLUP

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

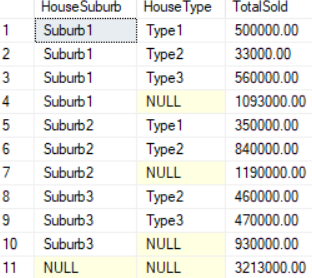
FROM    HouseSold

GROUP BY HouseSuburb ,

        HouseType

        WITH ROLLUP;

GO -- Run the previous command and begins new batch



2.4. Clean up

--========================================================================

--T032\_02\_04

--Clean up

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

=====================================================

3. Cube

--========================================================================

--T032\_03\_Cube

--========================================================================

/\*

1.

--GROUP BY CUBE (C1, C2, ..., Cn-1, Cn)

or

--GROUP BY C1, C2, ..., Cn-1, Cn WITH CUBE

CUBE produces the result set

by generating all combinations of columns

specified in GROUP BY CUBE().

The following clauses are equivalent:

1.1.

--GROUP BY CUBE (C1, C2, C3)

1.2.

--GROUP BY GROUPING SETS (

--    (C1, C2, C3)

--    ,(C1, C2)

--    ,(C1, C3)

--    ,(C2, C3)

--    ,(C1)

--    ,(C2)

--    ,(C3)

--    ,() )

\*/

3.1. Create Sample data

--========================================================================

--T032\_03\_01

--Create Sample data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSold

(

  Id INT IDENTITY(1, 1)

         PRIMARY KEY ,

  HouseStreetAddress NVARCHAR(100) ,

  HouseSuburb NVARCHAR(100) ,

  SoldPrice MONEY ,

  HouseType NVARCHAR(100)

);

GO -- Run the previous command and begins new batch

INSERT  dbo.HouseSold

VALUES  ( N'A1 Street', N'Suburb2', 400000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B1 Street', N'Suburb1', 500000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'C3 Street', N'Suburb1', 560000, N'Type3' );

INSERT  dbo.HouseSold

VALUES  ( N'D4 Street', N'Suburb2', 350000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'A5 Street', N'Suburb2', 440000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'A9 Street', N'Suburb3', 460000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B8 Street', N'Suburb3', 470000, N'Type3' );

INSERT  dbo.HouseSold

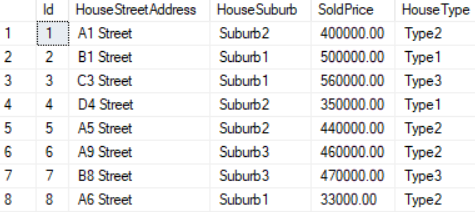
VALUES  ( N'A6 Street', N'Suburb1', 33000, N'Type2' );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.HouseSold;

GO -- Run the previous command and begins new batch



3.2. GROUP BY CUBE (C1, C2, ..., Cn-1, Cn)

--========================================================================

--T032\_03\_02

--GROUP BY CUBE (C1, C2, ..., Cn-1, Cn)

--------------------------------------------------------------------------

--T032\_03\_02\_01

-- ... UNION ALL ...

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY HouseSuburb ,

        HouseType

UNION ALL

SELECT  HouseSuburb ,

        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY HouseSuburb

UNION ALL

SELECT  NULL ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY HouseType

UNION ALL

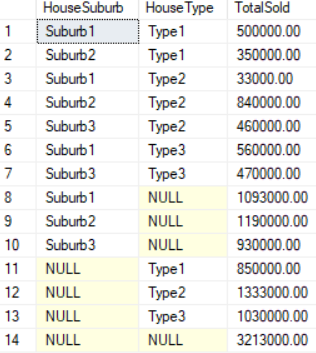
SELECT  NULL ,

        NULL ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold;

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_03\_02\_02

-- GROUP BY GROUPING SETS (...)

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

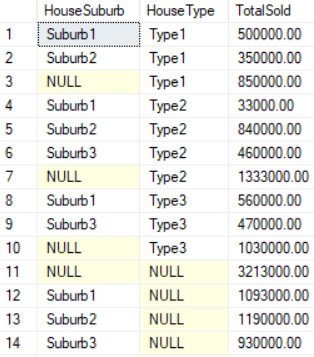
GROUP BY GROUPING SETS(

                       ( HouseSuburb ,

                         HouseType

                       ), ( HouseSuburb ), ( HouseType ), ( ));

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_03\_02\_03

-- GROUP BY CUBE(C1,C2,...)

SELECT  HouseSuburb ,

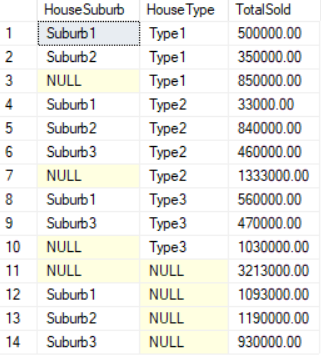
        HouseType ,

        SUM(SoldPrice) AS TotalSold

FROM    HouseSold

GROUP BY CUBE(HouseSuburb, HouseType);

GO -- Run the previous command and begins new batch



--------------------------------------------------------------------------

--T032\_03\_02\_04

-- GROUP BY (C1,C2...) WITH CUBE

SELECT  HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotalSold

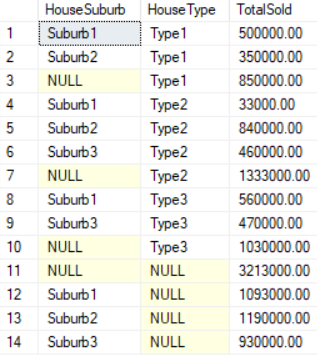
FROM    HouseSold

GROUP BY HouseSuburb ,

        HouseType

        WITH CUBE;

GO -- Run the previous command and begins new batch



3.3. Clean up

--========================================================================

--T032\_03\_03

--Clean up

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

=====================================================

4. Cube V.S. Rollup

--========================================================================

--T032\_04\_Cube V.S. Rollup

--========================================================================

4.1. Create Sample data

--========================================================================

--T032\_04\_01

--Create Sample data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSold

(

  Id INT IDENTITY(1, 1)

         PRIMARY KEY ,

  HouseStreetAddress NVARCHAR(100) ,

  HouseSuburb NVARCHAR(100) ,

  HouseState NVARCHAR(50) ,

  SoldPrice MONEY ,

  HouseType NVARCHAR(100)

);

GO -- Run the previous command and begins new batch

INSERT  dbo.HouseSold

VALUES  ( N'A1 Street', N'Suburb2', N'State01', 400000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B1 Street', N'Suburb1', N'State02', 500000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'C3 Street', N'Suburb1', N'State03', 560000, N'Type3' );

INSERT  dbo.HouseSold

VALUES  ( N'D4 Street', N'Suburb2', N'State03', 350000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'A5 Street', N'Suburb2', N'State02', 440000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'A9 Street', N'Suburb3', N'State02', 460000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B8 Street', N'Suburb3', N'State01', 470000, N'Type3' );

INSERT  dbo.HouseSold

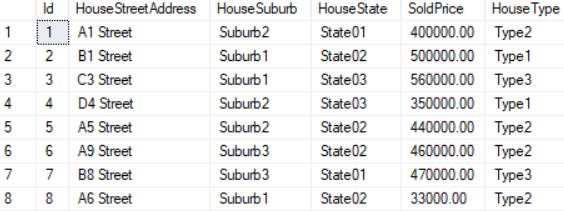
VALUES  ( N'A6 Street', N'Suburb1', N'State02', 33000, N'Type2' );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.HouseSold;

GO -- Run the previous command and begins new batch



4.2. GROUP BY ROLLUP(C1,C2,...)

--========================================================================

--T032\_04\_02

--GROUP BY ROLLUP(C1,C2,...)

SELECT  HouseSuburb ,

        HouseState ,

        HouseType ,

        SUM(SoldPrice) AS TotlSold

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseSuburb, HouseState, HouseType);

--GROUP BY HouseSuburb, HouseState, HouseType  WITH ROLLUP;

GO -- Run the previous command and begins new batch

/\*

Output as the following

--HouseSuburb, HouseState, HouseType

--HouseSuburb, HouseState,

--HouseSuburb

--()

\*/

Table

Description automatically generated

4.3. GROUP BY CUBE(C1,C2,...)

--========================================================================

--T032\_04\_03

--GROUP BY CUBE(C1,C2,...)

SELECT  HouseSuburb ,

        HouseState ,

        HouseType ,

        SUM(SoldPrice) AS TotlSold

FROM    dbo.HouseSold

GROUP BY CUBE(HouseSuburb, HouseState, HouseType);

--GROUP BY HouseSuburb, HouseState, HouseType  WITH CUBE;

GO -- Run the previous command and begins new batch

/\*

Output as the following

--HouseSuburb, HouseState, HouseType

--HouseSuburb, HouseState,

--HouseSuburb, HouseType

--HouseSuburb

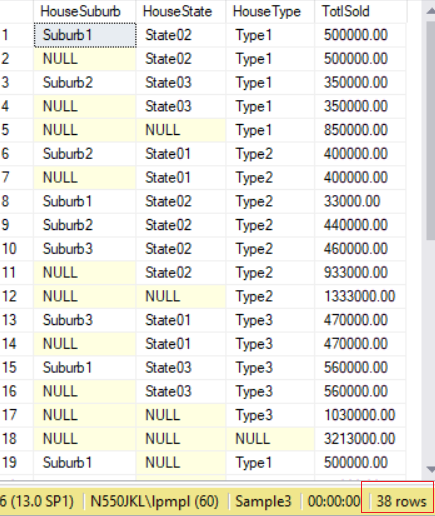
--HouseSuburb, HouseType

--HouseState,

--City

--()

\*/



4.4. ROLLUP and CUBE on a single column is no different.

--========================================================================

--T032\_04\_04

--ROLLUP and CUBE on a single column is no different.

--------------------------------------------------------------------------

--T032\_04\_04\_01

-- GROUP BY ROLLUP(C1,C2,...)

SELECT  HouseSuburb ,

        SUM(SoldPrice) AS TotlSold

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseSuburb);

--GROUP BY HouseSuburb WITH ROLLUP;

GO -- Run the previous command and begins new batch

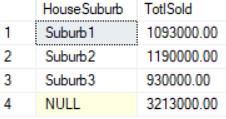
/\*

Output as the following

--HouseSuburb

--()

\*/



--------------------------------------------------------------------------

--T032\_04\_04\_02

-- GROUP BY CUBE(C1,C2,...)

SELECT  HouseSuburb ,

        SUM(SoldPrice) AS TotlSold

FROM    dbo.HouseSold

GROUP BY CUBE(HouseSuburb);

--GROUP BY HouseSuburb WITH CUBE;

GO -- Run the previous command and begins new batch

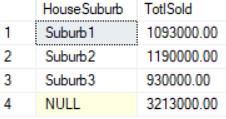
/\*

Output as the following

--HouseSuburb

--()

\*/



4.5. Clean up

--========================================================================

--T032\_04\_05

--Clean up

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

5. GroupingFunction

--========================================================================

--T032\_05\_GroupingFunction

--========================================================================

/\*

1.

--Grouping(columnA)

Syntax

--SELECT  ColumnA ,

--        SUM(ColumnB) AS TotalB ,

--        GROUPING(ColumnA) AS 'GroupingColumnA'

--FROM    dbo.TableName

--GROUP BY ROLLUP(ColumnA);

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/grouping-transact-sql>

<https://docs.microsoft.com/zh-cn/sql/t-sql/functions/grouping-transact-sql>

1.1.

if the columnA in a GROUP BY list is aggregated(Count, Sum, Avg, Min, Max)

then Grouping(columnA) return 1, otherwise return 0.

1.2.

When using ROLLUP, CUBE or GROUPING SETS,

the NULL returned might be normal standard null values,

or the NULL returned might be a column placeholder and means all.

1.3.

If SELECT Grouping(columnA) for that row return 0,

it means columnA in a GROUP BY list for that row is normal standard null values.

1.4.

If SELECT Grouping(columnA) for that row return 1,

it means columnA in a GROUP BY list for that row is a column placeholder

from ROLLUP, CUBE or GROUPING SETS, and it means all.

1.5.

Grouping(columnA) can be used in the

SELECT <select> list,

HAVING, and

ORDER BY clauses

when GROUP BY is specified.

\*/

5.1. Create Sample data

--========================================================================

--T032\_05\_01

--Create Sample data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSold

(

  Id INT IDENTITY(1, 1)

         PRIMARY KEY ,

  HouseStreetAddress NVARCHAR(100) ,

  HouseSuburb NVARCHAR(100) ,

  HouseState NVARCHAR(50) ,

  SoldPrice MONEY ,

  HouseType NVARCHAR(100)

);

GO -- Run the previous command and begins new batch

INSERT  dbo.HouseSold

VALUES  ( N'A1 Street', N'Suburb2', N'State01', 400000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B1 Street', N'Suburb1', N'State02', 500000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'C3 Street', N'Suburb1', N'State02', 560000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'D4 Street', N'Suburb2', N'State01', 350000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'A5 Street', N'Suburb1', N'State02', 440000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'A9 Street', N'Suburb1', N'State02', 460000, NULL );

INSERT  dbo.HouseSold

VALUES  ( N'B8 Street', N'Suburb3', N'State01', 470000, N'Type1' );

INSERT  dbo.HouseSold

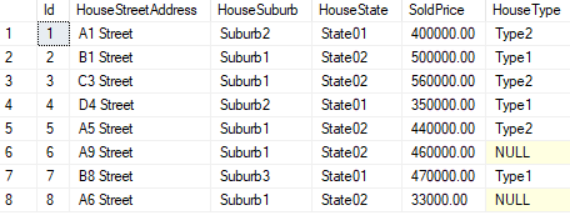
VALUES  ( N'A6 Street', N'Suburb1', N'State02', 33000, NULL );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.HouseSold;

GO -- Run the previous command and begins new batch



5.2. Grouping(columnA)

--========================================================================

--T032\_05\_02

--Grouping(columnA)

SELECT  HouseType ,

        SUM(SoldPrice) AS TotalSold ,

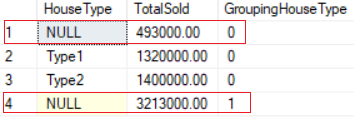
        GROUPING(HouseType) AS 'GroupingHouseType'

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseType);

--GROUP BY HouseType WITH ROLLUP;

GO -- Run the previous command and begins new batch



/\*

1.

--Grouping(columnA)

Syntax

--SELECT  ColumnA ,

--        SUM(ColumnB) AS TotalB ,

--        GROUPING(ColumnA) AS 'GroupingColumnA'

--FROM    dbo.TableName

--GROUP BY ROLLUP(ColumnA);

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/grouping-transact-sql>

<https://docs.microsoft.com/zh-cn/sql/t-sql/functions/grouping-transact-sql>

1.1.

if the columnA in a GROUP BY list is aggregated(Count, Sum, Avg, Min, Max)

then Grouping(columnA) return 1, otherwise return 0.

1.2.

When using ROLLUP, CUBE or GROUPING SETS,

the NULL returned might be normal standard null values,

or the NULL returned might be a column placeholder and means all.

1.3.

If SELECT Grouping(columnA) for that row return 0,

it means columnA in a GROUP BY list for that row is normal standard null values.

1.4.

If SELECT Grouping(columnA) for that row return 1,

it means columnA in a GROUP BY list for that row is a column placeholder

from ROLLUP, CUBE or GROUPING SETS, and it means all.

1.5.

Grouping(columnA) can be used in the

SELECT <select> list,

HAVING, and

ORDER BY clauses

when GROUP BY is specified.

2.

--SELECT  HouseType ,

--        SUM(SoldPrice) AS TotalSold ,

--        GROUPING(HouseType) AS 'GroupingHouseType'

--FROM    dbo.HouseSold

--GROUP BY ROLLUP(HouseType);

Output as following

--HouseType   TotalSold   GroupingHouseType

--NULL       493000.00         0

--Type1            1320000.00    0

--Type2            1400000.00    0

--NULL       3213000.00    1

The result set shows two NULL values under HouseType Column.

2.1.

--NULL    493000.00     0

The 1st NULL value under HouseType Column

means HouseType Column in a GROUP BY list for that row

is normal standard null values.

It represents the group of null values from the HouseType Column.

Thus, SELECT Grouping(HouseType) for that row will return 0,

2.2.

--NULL    3213000.00       1

The 2nd NULL value under HouseType Column

means HouseType Column in a GROUP BY list for that summary row is a column placeholder

from ROLLUP, CUBE or GROUPING SETS, and it means all.

Thus, SELECT Grouping(HouseType) for that row will return 1,

if the columnA in a GROUP BY list is aggregated(Count, Sum, Avg, Min, Max)

then Grouping(columnA) return 1, otherwise return 0.

\*/

5.3. Grouping(columnA)

--========================================================================

--T032\_05\_03

--Grouping(columnA)

SELECT  HouseState ,

        HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotlSold ,

        GROUPING(HouseState) AS GPHSt ,

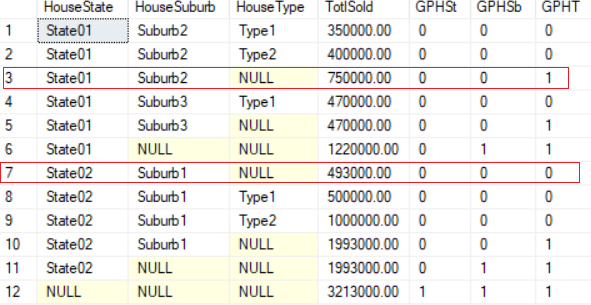
        GROUPING(HouseSuburb) AS GPHSb ,

        GROUPING(HouseType) AS GPHT

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType);

GO -- Run the previous command and begins new batch



/\*

1.

Output as the following

--HouseState   HouseSuburb   HouseType   TotlSold  GPHSt  GPHSb  GPHT

--State01       Suburb2      Type1        350000.00 0         0   0

--State01       Suburb2      Type2        400000.00 0         0   0

--State01       Suburb2      NULL             750000.00    0         0   1

--State01       Suburb3      Type1        470000.00 0         0   0

--State01       Suburb3      NULL             470000.00    0         0   1

--State01       NULL             NULL         1220000.00   0         1   1

--State02       Suburb1      NULL             493000.00    0         0   0

--State02       Suburb1      Type1        500000.00 0         0   0

--State02       Suburb1      Type2        1000000.000         0   0

--State02       Suburb1      NULL             1993000.00   0      0     1

--State02       NULL             NULL         1993000.00   0         1   1

--NULL        NULL         NULL         3213000.00   1         1   1

1.1.

--State01       Suburb2      NULL             750000.00    0         0   1

GPHT=1 here means GROUPING(HouseType) for that row is aggregated

from ROLLUP, CUBE or GROUPING SETS, and it means "ALL".

1.2.

--State02       Suburb1      NULL             493000.00    0         0   0

GPHT=0 here means GROUPING(HouseType) for that row is NOT aggregated

from ROLLUP, CUBE or GROUPING SETS.

It is normally standard group of NULL value, means "Unknow"

\*/

5.4. Grouping(columnA)

--========================================================================

--T032\_05\_04

--Grouping(columnA)

--------------------------------------------------------------------------

--T032\_05\_04\_01

--Replace Null by "ALL" if Grouping(columnA) return 1, otherwise by "Unknow"

SELECT  CASE WHEN GROUPING(HouseState) = 1 THEN 'All'

             ELSE ISNULL(HouseState, 'Unknown')

        END AS HouseState ,

        CASE WHEN GROUPING(HouseSuburb) = 1 THEN 'All'

             ELSE ISNULL(HouseSuburb, 'Unknown')

        END AS HouseSuburb ,

        CASE WHEN GROUPING(HouseType) = 1 THEN 'All'

             ELSE ISNULL(HouseType, 'Unknown')

        END AS HouseType ,

        SUM(SoldPrice) AS TotlSold

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType);

GO -- Run the previous command and begins new batch

Table

Description automatically generated

/\*

1.

In previous example.

--SELECT  HouseState ,

--        HouseSuburb ,

--        HouseType ,

--        SUM(SoldPrice) AS TotlSold ,

--        GROUPING(HouseState) AS GPHSt ,

--        GROUPING(HouseSuburb) AS GPHSb ,

--        GROUPING(HouseType) AS GPHT

--FROM    dbo.HouseSold

--GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType);

Output as the following

--HouseState   HouseSuburb   HouseType   TotlSold  GPHSt  GPHSb  GPHT

--State01       Suburb2      Type1        350000.00 0         0   0

--State01       Suburb2      Type2        400000.00 0         0   0

--State01       Suburb2      NULL             750000.00    0         0   1

--State01       Suburb3      Type1        470000.00 0         0   0

--State01       Suburb3      NULL             470000.00    0         0   1

--State01       NULL             NULL         1220000.00   0         1   1

--State02       Suburb1      NULL             493000.00    0         0   0

--State02       Suburb1      Type1        500000.00 0         0   0

--State02       Suburb1      Type2        1000000.000         0   0

--State02       Suburb1      NULL             1993000.00   0      0     1

--State02       NULL             NULL         1993000.00   0         1   1

--NULL        NULL         NULL         3213000.00   1         1   1

2.

In Current example.

--SELECT

--    CASE WHEN

--         GROUPING(HouseState) = 1 THEN 'All' ELSE ISNULL(HouseState, 'Unknown')

--    END AS HouseState,

--    CASE WHEN

--         GROUPING(HouseSuburb) = 1 THEN 'All' ELSE ISNULL(HouseSuburb, 'Unknown')

--    END AS HouseSuburb,

--    CASE

--         WHEN GROUPING(HouseType) = 1 THEN 'All' ELSE ISNULL(HouseType, 'Unknown')

--    END AS HouseType,

--    SUM(SoldPrice) AS TotlSold

--FROM dbo.HouseSold

--GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType)

Output as the following

--HouseState   HouseSuburb   HouseType   TotlSold

--State01       Suburb2        Type1        350000.00

--State01       Suburb2        Type2        400000.00

--State01       Suburb2        All          750000.00

--State01      Suburb3          Type1        470000.00

--State01       Suburb3        All          470000.00

--State01       All          All      1220000.00

--State02       Suburb1        Unknown  493000.00

--State02       Suburb1        Type1        500000.00

--State02       Suburb1        Type2        1000000.00

--State02       Suburb1        All          1993000.00

--State02       All          All      1993000.00

--All         All            All      3213000.00

2.1.

--State01       Suburb2        All          750000.00

ALL here means GROUPING(HouseType)=1 for that row is aggregated

from ROLLUP, CUBE or GROUPING SETS, and it means "ALL".

2.2.

--State02       Suburb1      NULL             493000.00

NULL here means GROUPING(HouseType)=0 for that row is NOT aggregated

from ROLLUP, CUBE or GROUPING SETS.

It is normally standard group of NULL value, means "Unknow"

\*/

--------------------------------------------------------------------------

--T032\_05\_04\_02

/\*

Replace Null by "ALL" if Grouping(columnA) return 1, otherwise by "Unknow"

If only using ISNULL, it will cause logic error,

The actuall NULL value in the raw data is also replaced with the word 'All',

which is incorrect. Therefore the need for Grouping function.

\*/

SELECT  ISNULL(HouseState, 'All') AS HouseState ,

        ISNULL(HouseSuburb, 'All') AS HouseSuburb ,

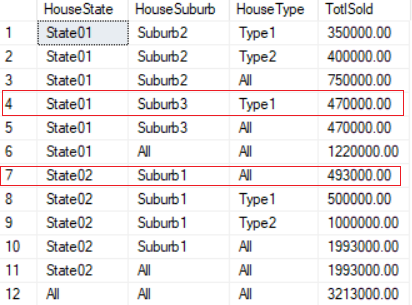
        ISNULL(HouseType, 'All') AS HouseType ,

        SUM(SoldPrice) AS TotlSold

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType);

GO -- Run the previous command and begins new batch



5.5. Clean up

--========================================================================

--T032\_05\_05

--Clean up

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

6. Grouping\_IDFunction

--========================================================================

--T032\_06\_Grouping\_IDFunction

--========================================================================

/\*

1.

--GROUPING\_ID(C1,C2,...Cn)

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/grouping-id-transact-sql>

<https://docs.microsoft.com/zh-cn/sql/t-sql/functions/grouping-id-transact-sql>

1.1.

Syntax

--SELECT  C1,C2,...Cn ,

--        SUM(ColumnB) AS TotalB ,

--        GROUPING\_ID(C1,C2,...Cn) AS 'GPID'

--FROM    dbo.TableName

--GROUP BY ROLLUP(C1,C2,...Cn);

1.1.1.

GROUPING\_ID(C1,C2,...Cn) function concatenates

all the GOUPING(C1), GOUPING(C2),...GOUPING(Cn) functions,

and then perform the binary string to decimal conversion.

1.1.2.

The column list of GROUPING\_ID(C1,C2,...Cn) must match

the column list of GROUP BY ROLLUP(C1,C2,...Cn).

1.1.3.

GROUPING\_ID(C1,C2,...Cn) function computes the level of grouping.

We normally use GROUPING\_ID(C1,C2,...Cn) in ORDER BY and HAVING clause to

order the ROLLUP or CUBE.

1.1.4.

GROUPING\_ID(C1,C2,...Cn) can be used in the

SELECT <select> list,

HAVING, and

ORDER BY clauses

when GROUP BY is specified.

This usage is same as Grouping(C1) function

1.2.

E.g.

--SELECT  C1,C2,C3,

--        SUM(ColumnB) AS TotalB ,

--        GROUPING\_ID(C1,C2,C3) AS 'GPID'

--FROM    dbo.TableName

--GROUP BY ROLLUP(C1,C2,C3);

GROUPING\_ID(C1,C2,C3) binary string =

CAST(GROUPING(C1) AS NVARCHAR(1)) +

CAST(GROUPING(C2) AS NVARCHAR(1)) +

CAST(GROUPING(C3) AS NVARCHAR(1));

GROUPING\_ID(C1,C2,C3) = convert GROUPING\_ID(C1,C2,C3)BinaryString to decimal.

Grouping(C1), Grouping(C2), or Grouping(C3) will return 1 or 0.

GROUPING\_ID(C1,C2,C3) function concatenates

all the GOUPING(C1), GOUPING(C2),GOUPING(C3) functions,

and then perform the binary string to decimal conversion.

------------------------------------------------------------------------

2.

----If function exists then DROP it

--IF ( EXISTS ( SELECT    \*

--              FROM      INFORMATION\_SCHEMA.ROUTINES

--              WHERE     ROUTINE\_TYPE = 'FUNCTION'

--                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

--                        AND SPECIFIC\_NAME = 'fnBinaryStrToDecimal' ) )

--    BEGIN

--        DROP FUNCTION fnBinaryStrToDecimal;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE FUNCTION [dbo].[fnBinaryStrToDecimal] ( @Input VARCHAR(255) )

--RETURNS BIGINT

--AS

--    BEGIN

--        DECLARE @Cnt TINYINT = 1;

--        DECLARE @Len TINYINT = LEN(@Input);

--        DECLARE @Output BIGINT = CAST(SUBSTRING(@Input, @Len, 1) AS BIGINT);

--        WHILE ( @Cnt < @Len )

--            BEGIN

--                SET @Output = @Output

--                    + POWER(CAST(SUBSTRING(@Input, @Len - @Cnt, 1) \* 2 AS BIGINT),

--                            @Cnt);

--                SET @Cnt = @Cnt + 1;

--            END;

--        RETURN @Output;

--    END;

--GO -- Run the previous command and begins new batch

--PRINT dbo.fnBinaryStrToDecimal('111')

\*/

--========================================================================

--T032\_06\_01

--Create Sample data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSold

(

  Id INT IDENTITY(1, 1)

         PRIMARY KEY ,

  HouseStreetAddress NVARCHAR(100) ,

  HouseSuburb NVARCHAR(100) ,

  HouseState NVARCHAR(50) ,

  SoldPrice MONEY ,

  HouseType NVARCHAR(100)

);

GO -- Run the previous command and begins new batch

INSERT  dbo.HouseSold

VALUES  ( N'A1 Street', N'Suburb2', N'State01', 400000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'B1 Street', N'Suburb1', N'State02', 500000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'C3 Street', N'Suburb1', N'State02', 560000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'D4 Street', N'Suburb2', N'State01', 350000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'A5 Street', N'Suburb1', N'State02', 440000, N'Type2' );

INSERT  dbo.HouseSold

VALUES  ( N'A9 Street', N'Suburb1', N'State02', 460000, NULL );

INSERT  dbo.HouseSold

VALUES  ( N'B8 Street', N'Suburb3', N'State01', 470000, N'Type1' );

INSERT  dbo.HouseSold

VALUES  ( N'A6 Street', N'Suburb1', N'State02', 33000, NULL );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.HouseSold;

GO -- Run the previous command and begins new batch

Table

Description automatically generated with medium confidence

--========================================================================

--T032\_06\_02

--fnBinaryStrToDecimal

--Reference:

--<http://improve.dk/converting-between-base-2-10-and-16-in-t-sql/>

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnBinaryStrToDecimal' ) )

    BEGIN

        DROP FUNCTION fnBinaryStrToDecimal;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION [dbo].[fnBinaryStrToDecimal] ( @Input VARCHAR(255) )

RETURNS BIGINT

AS

    BEGIN

        DECLARE @Cnt TINYINT = 1;

        DECLARE @Len TINYINT = LEN(@Input);

        DECLARE @Output BIGINT = CAST(SUBSTRING(@Input, @Len, 1) AS BIGINT);

             --Get the most right hand side binary digit as initial.

        WHILE ( @Cnt < @Len )

            BEGIN

                SET @Output = @Output

                    + POWER(CAST(SUBSTRING(@Input, @Len - @Cnt, 1) \* 2 AS BIGINT),

                            @Cnt);

                           --Keep Getting the most right hand side binary digit then convert it to decimal.

                           --1st loop, get the most right hand side binary digit then convert it to decimal.

                           --2nd loop, get the second last binary digit then convert it to decimal.

                           --...

                SET @Cnt = @Cnt + 1;

            END;

        RETURN @Output;

    END;

GO -- Run the previous command and begins new batch

PRINT dbo.fnBinaryStrToDecimal('111');



--========================================================================

--T032\_06\_03

--Grouping(columnA)

SELECT  HouseState ,

        HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotlSold ,

        GROUPING(HouseState) AS GPHSt ,

        GROUPING(HouseSuburb) AS GPHSb ,

        GROUPING(HouseType) AS GPHT ,

        CAST(GROUPING(HouseState) AS NVARCHAR(1))

        + CAST(GROUPING(HouseSuburb) AS NVARCHAR(1))

        + CAST(GROUPING(HouseType) AS NVARCHAR(1)) AS Gps ,

        dbo.fnBinaryStrToDecimal(CAST(GROUPING(HouseState) AS NVARCHAR(1))

                                 + CAST(GROUPING(HouseSuburb) AS NVARCHAR(1))

                                 + CAST(GROUPING(HouseType) AS NVARCHAR(1))) AS [fnBinaryStrToDecimal(GPs)] ,

        GROUPING\_ID(HouseState, HouseSuburb, HouseType) AS [GROUPING\_ID]

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType);

GO -- Run the previous command and begins new batch

/\*

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/grouping-id-transact-sql>

<https://docs.microsoft.com/zh-cn/sql/t-sql/functions/grouping-id-transact-sql>

1.

Output as the following

--HouseState   HouseSuburb   HouseType   TotlSold  GPHSt  GPHSb  GPHT Gps  fnBinaryStrToDecimal(GPs)  GROUPING\_ID

--State01       Suburb2      Type1        350000.00 0         0    0    000  0                          0

--State01       Suburb2      Type2        400000.00 0         0    0    000  0                          0

--State01       Suburb2      NULL             750000.00    0         0    1    001  1                          1

--State01       Suburb3      Type1        470000.00 0         0    0    000  0                          0

--State01       Suburb3      NULL             470000.00    0         0    1    001  1                          1

--State01       NULL             NULL         1220000.00   0         1    1    011  3                          3

--State02       Suburb1      NULL             493000.00    0         0    0    000  0                          0

--State02       Suburb1      Type1        500000.00 0         0    0    000  0                          0

--State02       Suburb1      Type2        1000000.000         0    0    000  0                          0

--State02       Suburb1      NULL             1993000.00   0      0      1    001  1                          1

--State02       NULL             NULL         1993000.00   0         1    1    011  3                          3

--NULL        NULL         NULL         3213000.00   1         1    1    111  7                          7

1.1.

--State01       Suburb2      NULL             750000.00    0         0   1  001  1   1

1.1.1.

GPHT=1 here means GROUPING(HouseType) for that row is aggregated

from ROLLUP, CUBE or GROUPING SETS, and it means "ALL".

1.1.2.

GROUPING\_ID(C1,C2,C3) binary string =

CAST(GROUPING(C1) AS NVARCHAR(1)) +

CAST(GROUPING(C2) AS NVARCHAR(1)) +

CAST(GROUPING(C3) AS NVARCHAR(1));

GROUPING\_ID(C1,C2,C3) = convert GROUPING\_ID(C1,C2,C3)BinaryString to decimal.

Grouping(C1), Grouping(C2), or Grouping(C3) will return 1 or 0.

GROUPING\_ID(C1,C2,C3) function concatenates

all the GOUPING(C1), GOUPING(C2),GOUPING(C3) functions,

and then perform the binary string to decimal conversion.

1.1.3.

In this case,

GROUPING\_ID(HouseState, HouseSuburb, HouseType)

= convert GROUPING\_ID(HouseState, HouseSuburb, HouseType)BinaryString to decimal

= fnBinaryStrToDecimal(GPs) = fnBinaryStrToDecimal(001)

= 1

-----------------

1.2.

--State02       Suburb1      NULL             493000.00    0         0   0  000  0   0

1.2.1.

GPHT=0 here means GROUPING(HouseType) for that row is NOT aggregated

from ROLLUP, CUBE or GROUPING SETS.

It is normally standard group of NULL value, means "Unknow"

1.2.2.

In this case,

GROUPING\_ID(HouseState, HouseSuburb, HouseType)

= convert GROUPING\_ID(HouseState, HouseSuburb, HouseType)BinaryString to decimal

= fnBinaryStrToDecimal(GPs) = fnBinaryStrToDecimal(000)

= 0

---------------

1.3.

--State02       NULL             NULL         1993000.00   0         1    1    011  3  3

1.3.1.

(GPHSb=1 and GPHT=1) here means

(GROUPING(HouseSuburb), and GROUPING(HouseType))

for that row is aggregated

from ROLLUP, CUBE or GROUPING SETS, and it means "ALL".

1.3.2.

In this case,

GROUPING\_ID(HouseState, HouseSuburb, HouseType)

= convert GROUPING\_ID(HouseState, HouseSuburb, HouseType)BinaryString to decimal

= fnBinaryStrToDecimal(GPs) = fnBinaryStrToDecimal(011)

= 3

---------------

1.4.

--NULL        NULL         NULL         3213000.00   1         1    1    111  7   7

GPHSt  GPHSb  GPHT

1.4.1.

(GPHSt=1, GPHSb=1 and GPHT=1) here means

(GROUPING(HouseState),GROUPING(HouseSuburb), and GROUPING(HouseType))

for that row is aggregated

from ROLLUP, CUBE or GROUPING SETS, and it means "ALL".

1.3.2.

In this case,

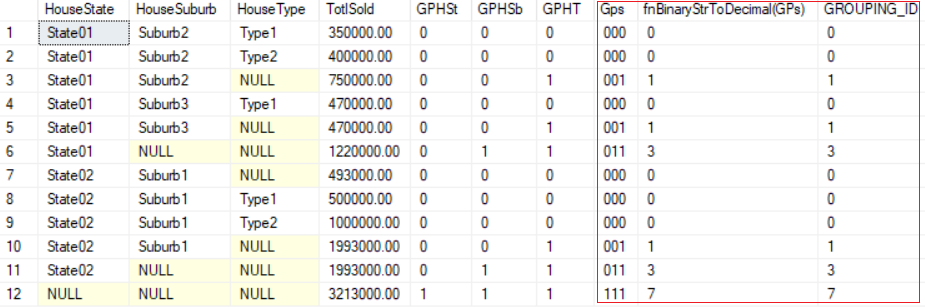
GROUPING\_ID(HouseState, HouseSuburb, HouseType)

= convert GROUPING\_ID(HouseState, HouseSuburb, HouseType)BinaryString to decimal

= fnBinaryStrToDecimal(GPs) = fnBinaryStrToDecimal(111)

= 7

\*/



--========================================================================

--T032\_06\_04

--Grouping(columnA)

SELECT  HouseState ,

        HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotlSold ,

        GROUPING(HouseState) AS GPHSt ,

        GROUPING(HouseSuburb) AS GPHSb ,

        GROUPING(HouseType) AS GPHT ,

        CAST(GROUPING(HouseState) AS NVARCHAR(1))

        + CAST(GROUPING(HouseSuburb) AS NVARCHAR(1))

        + CAST(GROUPING(HouseType) AS NVARCHAR(1)) AS Gps ,

        dbo.fnBinaryStrToDecimal(CAST(GROUPING(HouseState) AS NVARCHAR(1))

                                 + CAST(GROUPING(HouseSuburb) AS NVARCHAR(1))

                                 + CAST(GROUPING(HouseType) AS NVARCHAR(1))) AS [fnBinaryStrToDecimal(GPs)] ,

        GROUPING\_ID(HouseState, HouseSuburb, HouseType) AS [GROUPING\_ID]

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType)

ORDER BY [GROUPING\_ID];

GO -- Run the previous command and begins new batch

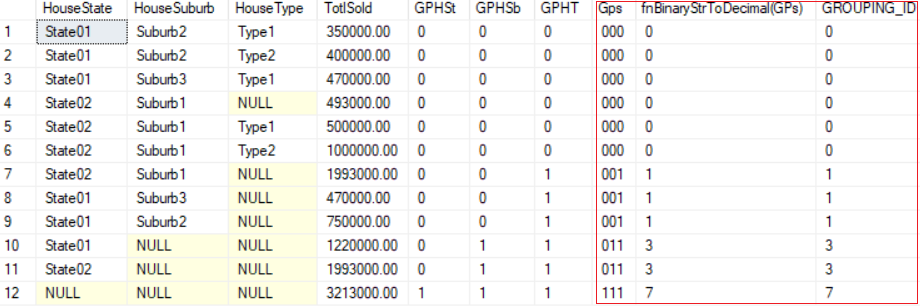
/\*

GROUPING\_ID(C1,C2,...Cn) function computes the level of grouping.

We normally use GROUPING\_ID(C1,C2,...Cn) in ORDER BY and HAVING clause to

order the ROLLUP or CUBE.

\*/



--========================================================================

--T032\_06\_05

--Grouping(columnA)

SELECT  HouseState ,

        HouseSuburb ,

        HouseType ,

        SUM(SoldPrice) AS TotlSold ,

        GROUPING(HouseState) AS GPHSt ,

        GROUPING(HouseSuburb) AS GPHSb ,

        GROUPING(HouseType) AS GPHT ,

        CAST(GROUPING(HouseState) AS NVARCHAR(1))

        + CAST(GROUPING(HouseSuburb) AS NVARCHAR(1))

        + CAST(GROUPING(HouseType) AS NVARCHAR(1)) AS Gps ,

        dbo.fnBinaryStrToDecimal(CAST(GROUPING(HouseState) AS NVARCHAR(1))

                                 + CAST(GROUPING(HouseSuburb) AS NVARCHAR(1))

                                 + CAST(GROUPING(HouseType) AS NVARCHAR(1))) AS [fnBinaryStrToDecimal(GPs)] ,

        GROUPING\_ID(HouseState, HouseSuburb, HouseType) AS [GROUPING\_ID]

FROM    dbo.HouseSold

GROUP BY ROLLUP(HouseState, HouseSuburb, HouseType)

HAVING  GROUPING\_ID(HouseState, HouseSuburb, HouseType) > 2

ORDER BY [GROUPING\_ID];

GO -- Run the previous command and begins new batch

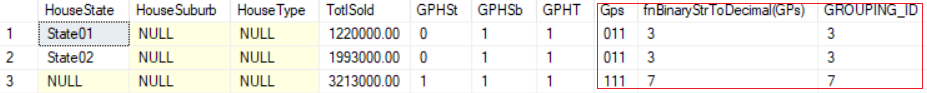
/\*

GROUPING\_ID(C1,C2,...Cn) function computes the level of grouping.

We normally use GROUPING\_ID(C1,C2,...Cn) in ORDER BY and HAVING clause to

order the ROLLUP or CUBE.

\*/



--========================================================================

--T032\_06\_06

--Clean up

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSold' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSold;

        DROP TABLE HouseSold;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnBinaryStrToDecimal' ) )

    BEGIN

        DROP FUNCTION fnBinaryStrToDecimal;

    END;

GO -- Run the previous command and begins new batch