Who are the top five athlete's from any country to have won more than one Gold Medal?”

DB2 Universal Database’s Business Intelligence Functions assist in the Sydney 2000 Olympic Games

Dan Gibson
IBM Canada Lab
DGIBSON@CA.IBM.COM

Introduction

The Olympic Cauldron is lit on September 15th and the Closing Ceremonies take place on October 1st. During this brief 17 day period, the Sydney 2000 Olympic Games will be the focus of the world as each country cheers on their representative athletes to award winning performances.

The IBM data management family of products are a major part of making the Sydney 2000 Olympic Games a success:

DB2 on an OS/390 is the major repository for the Games. Here the Central Results System stores the results received from all sports venues. In addition, all of the World New Press Agencies receive the data needed to deliver information worldwide to their various clients from this same repository.

DB2 Universal Database is the database management product behind the INFO kiosks. This system is responsible for providing members of the Olympic Family comprehensive information such as results, events schedules, athletes results, and athlete biographies, and news.

The Official Sydney 2000 Games Web site uses DB2 Universal Database to collect and manage live results data during the games. In addition to live results, information such as event ticket sales and games merchandise is also available.

DB2 Universal Database supplies a variety of built-in Business Intelligence functions that perform functions such as ranking, statistical analysis, as well as multi-dimensional analysis - all using just SQL. These Business Intelligence functions allow for users to answer questions like, “How many suits have we sold in California and Arizona during the last two quarters?” Or, “What are the top 25 bestsellers being ordered in the northeastern part of the United States?” And of course, “Who are the top five athlete’s from any country to have won more than one gold medal?”

This paper presents examples of how the built in Business Intelligence Functions of DB2 Universal Database can assist in the development and implementation of the IBM Olympic Games System at the Sydney 2000 Olympic Games.

Athlete Ranking (or Who's number one?)

The ranking of athletes is used in almost every sport to provide a means of presenting the order of completion that an athlete finished a competition in. In DB2 Universal Database, two ranking functions are provided. These are the RANK and the DENSERANK functions.

When the RANK function is specified, 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, there will be one or more gaps in the sequential rank numbering.

If the DENSERANK function is specified, the rank of a row is defined as "1 plus the number of rows preceding the row that are distinct with respect to the ordering." Therefore, there will be no gaps in the sequential rank numbering.

Let's start with a simple example. Suppose for a single event (diving) you wanted to rank the athletes from first place onward. Assume the following table is used:

Event	Athlete	City	Score
Diving	Dan	Sydney	10.0
Diving	Dave	Sydney	9.9
Diving	Rob	Sydney	9.8
Diving	Grant	Melbourne	9.7
Diving	Gene	Melbourne	9.9
Diving	Leon	Melbourne	7.0
Diving	Ed	Perth	9.0
Diving	Bob	Perth	9.5

The following query ranks the above athletes by their score:

```
SELECT event, firstname,  score, city,
    rank over (order by score desc) as Rank              -- Rank by score
FROM medals
ORDER BY Rank;
```

Below are the results:

Event	Firstname	Score	City	Rank
Diving	Dan	10.0	Sydney	1
Diving	Dave	9.9	Melbourne	2
Diving	Gene	9.9	Sydney	2
Diving	Rob	9.8	Sydney	4
Diving	Grant	9.7	Melbourne	5
Diving	Bob	9.5	Perth	6
Diving	Ed	9.0	Perth	7
Diving	Leon	7.0	Melbourne	8

--Dave and Gene are
--correctly ranked 2nd
-- Rob should be 3rd

In this example notice that both Dave and Gene are tied for 2nd. However, Rob is ranked 4th as opposed to 3rd. This is because when using the ***rank*** function, the rank of a row is defined as 1 plus the number of rows that strictly precede the row. Therefore if two or more rows are not distinct with respect to the ordering, there will be one or more gaps in the sequential rank numbering. To achieve proper ranking in the above example, the ***denserank*** function must be used.

Here is the same query using denserank instead:

```
SELECT event, firstname,  score, city,
    denserank over (order by score desc) as Rank                --Rank by score
 FROM medals
ORDER BY Rank;
```

Here is the output using denserank:

Event	Firstname	Score	City	Rank
Diving	Dan	10.0	Sydney	1
Diving	Dave	9.9	Melbourne	2
Diving	Gene	9.9	Sydney	2
Diving	Rob	9.8	Sydney	3
Diving	Grant	9.7	Melbourne	4
Diving	Bob	9.5	Perth	5
Diving	Ed	9.0	Perth	6
Diving	Leon	7.0	Melbourne	7

Notice that using the denserank function Rob is ranked correctly (3rd) as well as the other athletes.

For all examples that involve ranking, the function denserank will be used.

Which athletes won the Gold, Silver, and Bronze?

While the above query provides the correct ranking of every athlete, the query does not tell you who won a medal. Displaying which athlete won which medal can be easily accomplished using the CASE expression:

```
SELECT event, firstname,  score, city,
       denserank() over (order by score desc) as Rank,
       CASE denserank() over ( order by score desc)           --Obtain the rank of the Athlete
          WHEN 1 then 'Gold'                                   --and based the value
          WHEN 2 then 'Silver'                                 --return Gold, Silver or Bronze
          WHEN 3 then 'Bronze'
       END as Medal
 FROM medals
ORDER BY Rank;
```

This returns:

Event	Firstname	Score	City	Rank	Medal
Diving	Dan	10.0	Sydney	1	Gold
Diving	Gene	9.9	Melbourne	2	Silver
Diving	Dave	9.9	Sydney	2	Silver
Diving	Rob	9.8	Sydney	3	Bronze
Diving	Grant	9.7	Melbourne	4	---------
Diving	Bob	9.5	Perth	5	---------
Diving	Ed	9.0	Perth	6	---------
Diving	Leon	7.0	Melbourne	7	---------

Which athlete from each city has the highest score?

So far we have been returning all rows of the table. But suppose we wanted to know "Which person from each city has the highest score?". To do this we would need to order by score within each city. That is we would **partition by city, order by score**:

```
Select firstname, score, city, rank from
(
SELECT event, firstname,  score, city,
       denserank() over (partition by city                     --Within each city
                   order by score desc) as Rank                -- Order by Score
  FROM medals
) as nested_medals
```

```
Where rank = 1                                         --But only return those ranked
ORDER BY Rank;                                         --Number one in each city
```

Here is the result of the above query:

Firstname	Score	City	Rank
Bob	9.5	Perth	1
Dan	10.0	Sydney	1
Gene	9.9	Melbourne	1

Notice that in addition to the use of the denserank() function, the clause **partition by** is also used. The expression specified defines the partition within which the ranking or numbering is applied.

The above query uses what is called a **nested table expression**. A nested table expression is a temporary view where the definition is nested (defined directly) in the FROM clause of the main query. Table expressions are temporary and are only valid for the life of the SQL statement; they cannot be shared, but they allow more flexibility than views.

DB2 Universal Database also supports another type of table expressions called **Common Table Expressions**. A common table expression is a named result table that is defined using the WITH keyword prior to the beginning of a fullselect. Repeated references to a common table expression use the same result set. By comparison, if you used nested table expressions or views, the result set would be regenerated each time, with possibly different results.

Here is the above query rewritten using a common table expression:

```
WITH Country_First (firstname, score, city, event, rank)

AS

(
SELECT    firstname, score, city, event,
denserank() over (partition by city                   --Within each city
                  order by score desc) as Rank          -- Order by Score
FROM medals
)

Select firstname, score, city
from Country_First
Where rank = 1                                         --But only return those ranked
ORDER BY Rank;                                         --Number one in each city
```

By city, by event, by day

The queries posed so far have only involved a single sport. But suppose you wanted to view multiple sports by various categories? For example, suppose you wanted to know the total number of medals that each city has won each day, and the total number of medals awarded so far? Or even better - how many Gold, Silver, and Bronze medals were awarded each day, to each city, as well as the total number of medals awarded each day to each city? Notice that these questions are asking information from various categories (medals, city, and day). These categories are commonly referred to as ***dimensions***.

Dimensions allow for the organizing of data in such a manner that the above questions Can easily be answered. For example, using the above dimensions this "cube" depicts the intersection of three axes: Product (medals), Geography (city), and Time (day), enabling users to answer questions like, “How many medals has each city won by day?”

The phrase Online Analytical Processing (OLAP) is often used to describe the above type of queries. DB2 Universal Database easily supports answering these types of complex queries thru the use of it’s built-in OLAP capability. This capability has been implemented in DB2 Universal Database by extending the GROUP BY clause. Below is a very brief explanation of those extensions:

The GROUP BY clause has been extended to support "super groups". One type of super group is a " ROLLUP group"; a result set that contains "sub-total" and "overall total" rows in addition to the regular grouped rows. Another type of super group is a "CUBE group"; a result set that contains "cross-tabulation" rows in addition to all the rows that would be in a ROLLUP group for the same columns.

The table below contains the two sporting events diving and gymnastics:

Event	Athlete	City	Score
Diving	Dan	Sydney	10.0
Diving	Dave	Sydney	9.8
Diving	Rob	Sydney	9.8
Diving	Grant	Melbourne	9.7
Diving	Gene	Melbourne	9.9
Diving	Leon	Melbourne	7.0
Diving	Ed	Perth	9.0
Diving	Bob	Perth	9.5
Gymnastics	Jack	Sydney	8.0
Gymnastics	Chris	Sydney	9.8
Gymnastics	Tim	Sydney	9.0
Gymnastics	Bill	Melbourne	10.0

Gymnastics	George	Melbourne	9.9
Gymnastics	Phil	Melbourne	7.0
Gymnastics	Terry	Perth	9.8
Gymnastics	Gary	Perth	9.5

Suppose you wanted to answer the question “What are the total number of medals each city has won?” The query would be:

```
WITH MEDAL_INFO (day, score, city, event, medal)
AS
(
SELECT day, score, city, event,
     CASE denserank() over (partition by event order by score desc) -- By event, order by

       WHEN 1 then 'Gold'                                              --score, assign medals
       WHEN 2 then 'Silver'
       WHEN 3 then 'Bronze'
      END as Medal
  FROM medals)

Select day, city, count(medal)                                       -- and count medals by
from  medal_info where medal is not null                             -- city
group by (day, city)
order by day, city;
```

Below is the result:

DAY	CITY	Count
Mon	Sydney	3
Mon	Melbourne	1
Tue	Perth	1
Tue	Sydney	1
Tue	Melbourne	2

The above query shows the total number of medals won by each city. But suppose you wanted to answer the question “What are the total number of medals each city has won each day as well as the total number of medals awarded so far ?” In order to include these sub-totals (by day) as well as the final total, the ROLLUP function Can be used:

```
 WITH MEDAL_INFO (day, score, city, event, medal)
AS
(
SELECT day, score, city, event,
       CASE denserank() over (partition by event order by score desc) -- By event, order by

          WHEN 1 then 'Gold'                                        -- score, assign medals
          WHEN 2 then 'Silver'
          WHEN 3 then 'Bronze'
        END as Medal
  FROM medals)

Select day, city, count(medal) as COUNT                             -- count medals by
from  medal_info where medal is not null                           -- city
group by  ROLLUP (day, city)                                       -- and include sub-totals
order by day, city;
```

In addition to the rows returned in the first query, notice the additional rows returned:

DAY	CITY	COUNT	
Mon	Sydney	3	
Mon	Melbourne	1	
Mon	--	4	--- Subtotal of medals for Monday
Tue	Perth	1	
Tue	Sydney	1	
Tue	Melbourne	2	
Tue	--	4	--- Subtotal of medals for Tuesday
--	---	8	--- Total number of medals awarded

Observe what happens if the same query is executed but instead of grouping by ROLLUP(day, city), ROLLUP(city,day) was used:

```
WITH MEDAL_INFO (day, score, city, event, medal)
AS
(
SELECT day, score, city, event,
       CASE denserank() over (partition by event order by score desc) -- By event, order by

          WHEN 1 then 'Gold'                                        -- score, assign medals
          WHEN 2 then 'Silver'
          WHEN 3 then 'Bronze'
        END as Medal
   FROM medals)

Select day, city, count(medal) as COUNT                             -- count medals by
from medal_info where medal is not null                            -- city
```

```
group by  ROLLUP (city, day)                                     -- and include sub-totals
order by day, city;
```

Notice how now totals by city are returned versus totals by day:

DAY	CITY	COUNT	
Mon	Sydney	3	
Mon	Melbourne	1	
Tue	Perth	1	
Tue	Sydney	1	
Tue	Melbourne	2	
--	Perth	1	--- Total medals for Perth
--	Sydney	4	--- Total medals for Sydney
--	Melbourne	3	--- Total medals for Melbourne
--	--	8	--- Grand Total medal count

Below are a few more examples. The underlined portion of each query shows what has been added to the query from the previous one.

Query:

Show by day how many Gold, Silver, and Bronze medals each city has won
and the total number of medals awarded each day.

```
WITH MEDAL_INFO (day, score, city, event, medal)
AS
(
SELECT    day,  score, city, event,
       CASE denserank() over (partition by event order by score desc)
          WHEN 1 then 'Gold'
          WHEN 2 then 'Silver'
          WHEN 3 then 'Bronze'
      END as Medal
 FROM medals)

Select day, city, medal , count(medal) as Count
from medal_info where medal is not null
group by rollup(day,(city,medal))
order by day,city;
```

DAY	CITY	MEDAL	COUNT	

Mon	Sydney	Bronze	2	
Mon	Sydney	Gold	1	
Mon	Melbourne	Silver	1	
Mon	--	--	4	-- Total awarded Monday
Tue	Perth	Bronze	1	
Tue	Sydney	Bronze	1	
Tue	Melbourne	Gold	1	
Tue	Melbourne	Silver	1	
Tue	--	--	4	-- Total awarded Tuesday
--	--	--	8	

Query:

Show by day how many Gold, Silver, and Bronze medals each city has won, the total number of medals awarded each day and the total number of medals each city has won overall.

```
WITH MEDAL_INFO (day, score, city, event, medal)
AS
(
SELECT    day, score, city, event,
  CASE denserank() over (partition by event order by score desc)
         WHEN 1 then 'Gold'
         WHEN 2 then 'Silver'
         WHEN 3 then 'Bronze'
  END as Medal
 FROM medals)

Select day, city, medal, count(medal) as Count
from  medal_info where medal is not null
group by cube(day,(city,medal))
order by day,city;
```

DAY	CITY	MEDAL	COUNT
Mon	Sydney	Bronze	2
Mon	Sydney	Gold	1
Mon	Melbourne	Silver	1
Mon	--	--	4
Tue	Perth	Bronze	1

Tue	Sydney	Bronze	1
Tue	Melbourne	Gold	1
Tue	Melbourne	Silver	1
Tue	--	--	4
--	Perth	Bronze	1
--	Sydney	Bronze	3
--	Sydney	Gold	1
--	Melbourne	Gold	1
--	Melbourne	Silver	2
--	--	--	8

Conclusion

Business Intelligence means using your data assets to make better business decisions. Using the built-in Business Intelligence functionality of DB2 Universal Database allows the user to have easier access to these data assets allowing for faster analysis of one’s data to gain a competitive advantage.