**Data warehousing and OLAP in MYSQL – Practice 1/2**

**1. Problem specifications**

A telephone company is interested in analysing its own data to improve customer services. At present, the company has a database with call logs. For each call, the caller and receiver phone numbers, the duration, the type of charge (e.g., peak, off-peak rates), the start time (date, hour, minute, second) are known. The managers want to obtain very fast the information about the telephone traffic on the company lines and the daily income based on the caller location, the day and the phone rate.

In particular, the managers want to analyze the following situations:

- Monthly net income and number of calls for each caller city.

- Monthly net income and number of calls for each receiver city.

- Monthly net income and number of calls for each caller province and region.

- Monthly net income and number of calls for each receiver province and region.

- Daily net income and number of calls for each caller province.

- Yearly net income and number of calls for each caller province and region.

- Monthly net income and number of calls for each phone rate (type of charge).

- Net income and number of calls for each day of the week and phone rate.

- Daily number of calls for each caller region.

- Daily number of calls for each receiver region.

The OLTP database of the telephone company is reported in **Table 1.**

**2. Querying the data warehouse**

In **Table 2** the proposed solution for the data warehouse is shown.

* Create the corresponding tables in MYSQL/Oracle and upload the sample data xls data from BB.
  + **First Create the Dimensions tables and then the Business Process or Fact Table(s)#**
  + **#First Insert Data in the Dimensions tables and then in the Business Process or Fact Table#**
* Use these tables as source to answer the following OLAP queries.

*1. Select the yearly income for each phone rate, the total income for each phone rate, the total yearly income and the total income.*

*2. Select the number of calls and the yearly income. Associate the RANK() to each year according to its income (1 for the year with the highest income, 2 for the second, etc., the last year is the one with the least income).*

*3. For the year 2004, select the total number of calls.*

**3. GROUP WITH ROLLUP in MYSQL**

If you have more than one column specified in the GROUP BY clause, the ROLLUP clause assumes a hierarchy among the input columns.

**For example:**

GROUP BY c1, c2, c3 **WITH** **ROLLUP**

The ROLLUP assumes that there is the following hierarchy: c1 > c2 > c3 And it generates the following grouping sets: (c1, c2, c3) (c1, c2) (c1) ()

And in case you have two columns specified in the GROUP BY clause: GROUP BY c1, c2 **WITH** **ROLLUP**

then the ROLLUP generates the following grouping sets: (c1, c2) (c1) ()

# Assigning Ranks In MYSQL Before Version 8.0

## Problem

You want to assign ranks to a set of values.

## Solution

Decide on a ranking method, then put the values in the desired order and apply the method to them.

## Discussion

Some kinds of statistical tests require assignment of ranks. I’ll describe three ranking methods and show how each can be implemented using SQL variables. The examples assume that a table t contains the following scores, which are to be ranked with the values in descending order:

mysql> **SELECT score FROM t ORDER BY score DESC;**

+-------+

| score |

+-------+

| 5 |

| 4 |

| 4 |

| 3 |

| 2 |

| 2 |

| 2 |

| 1 |

+-------+

One type of ranking simply assigns each value its row number within the ordered set of values. To produce such rankings, keep track of the row number and use it for the current rank:

mysql> **SET @rownum := 0;**

mysql> **SELECT @rownum := @rownum + 1 AS rank, score**

-> **FROM t ORDER BY score DESC;**

+------+-------+

| rank | score |

+------+-------+

| 1 | 5 |

| 2 | 4 |

| 3 | 4 |

| 4 | 3 |

| 5 | 2 |

| 6 | 2 |

| 7 | 2 |

| 8 | 1 |

+------+-------+

That kind of ranking doesn’t take into account the possibility of ties (instances of values that are the same). A second ranking method does so by advancing the rank only when values change:

mysql> **SET @rank = 0, @prev\_val = NULL;**

mysql> **SELECT @rank := IF(@prev\_val=score,@rank,@rank+1) AS rank,**

-> **@prev\_val := score AS score**

-> **FROM t ORDER BY score DESC;** +------+-------+ | rank | score | +------+-------+ | 1 | 5 | | 2 | 4 | | 2 | 4 | | 3 | 3 | | 4 | 2

|  |  |
| --- | --- |
| **Tables** | **Description** |
| **DWABD.PHONERATES**  (  phoneRateType INT NOT NULL,  phoneRateName VARCHAR(20) NOT NULL, phoneRate\_CostPerSecond FLOAT NOT NULL, PRIMARY KEY(phoneRateType)  ); | Different phone rates 7 rows |
| **DWABD.PLACES**  (  Places\_ID INT NOT NULL,  City VARCHAR(20) NOT NULL,  Province VARCHAR(20) NOT NULL,  Region VARCHAR(20) NOT NULL, PRIMARY KEY(Places\_ID)  ); | Places 1500 rows |
| **DWABD.CALLS**  (  CallerPhoneNumber VARCHAR(20) NOT NULL, ReceiverPhoneNumber VARCHAR(20) NOT NULL, CallerLocation INT NOT NULL,  ReceiverLocation INT NOT NULL,  FullDate DATE NOT NULL,  StartTimeHour INT NOT NULL,  StartTimeMinute INT NOT NULL,  StartTimeSecond INT NOT NULL,  CallDuration FLOAT NOT NULL,  phoneRateType INT NOT NULL, PRIMARY  KEY(CallerPhoneNumber,ReceiverPhoneNumber,FullDate,StartTimeHour  ,StartTimeMinute,StartTimeSecond), FOREIGN KEY(phoneRateType) REFERENCES  PhoneRates(phoneRateType) ON DELETE CASCADE,  FOREIGN KEY(CallerLocation)REFERENCES Places(Places\_ID) ON DELETE CASCADE,  FOREIGN KEY(ReceiverLocation) REFERENCES Places(Places\_ID) ON DELETE CASCADE  ); | Calls in 2003 and 2004  ~ 1300000 rows |

Table 1 – OLTP/ Source data base with single call information

|  |  |
| --- | --- |
| **Tables** | **Description** |
| **TIMEDIM**  (  ID\_time INT NOT NULL,  DayOfWeek CHAR(15) NOT NULL,  DateMonth CHAR(15) NOT NULL,  DateYear INT NOT NULL,  PRIMARY KEY(ID\_time)); | Time dimension 10 rows |
| **PHONERATE**  (  ID\_phoneRate INTEGER NOT NULL, phoneRateType VARCHAR(20) NOT NULL,  PRIMARY KEY(ID\_phoneRate)  ); | Phone rate dimension  7 rows |
| **LOCATION**  (  ID\_location INTEGER NOT NULL,  City VARCHAR(20) NOT NULL,  Province CHAR(20) NOT NULL,  Region CHAR(20) NOT NULL,  PRIMARY KEY(ID\_location)  ); | Place dimension  1500 rows |
| **BUSINESS Process**  (  ID\_time INTEGER NOT NULL,  ID\_phoneRate INTEGER NOT NULL, ID\_location\_Caller INTEGER NOT NULL, ID\_location\_Receiver INTEGER NOT NULL, Price FLOAT NOT NULL,  NumberOfCalls INTEGER NOT NULL,  PRIMARYKEY(ID\_time,ID\_phoneRate,ID\_location\_Caller,ID\_locaton\_Receiver),  FOREIGN KEY(ID\_time) REFERENCES timeDim(ID\_time),  FOREIGN KEY(ID\_phoneRate) REFERENCES phoneRate(ID\_phoneRate),  FOREIGN KEY(ID\_location\_Caller) REFERENCES location(ID\_location),  FOREIGN KEY(ID\_location\_Receiver) REFERENCES location(ID\_location)  ); | Fact table 7809 rows |

Table 2 – Proposed solution - Data warehouse tables