Data Warehouse exercise

European Union: evaluation of scientific activity

Research activity evaluation

To evaluate the results of publicly funded research activities, European Union would like to analyse scientific publications of researchers working in European universities.

Researchers and professors at European universities present their research results by writing scientific papers, generally named "publications".

Each publication has a specific type (e.g., conference paper, journal paper, book chapter, etc.), and it is characterized by a specific date of publication, one or more authors, and a publication venue (e.g., conference, journal, workshop, book, etc.), which determines the specific publication type. The publication venue (conference, journal, etc.) has an editor (e.g., Elsevier) and can have one or more editions, whose year is of interest for the European Union analysis (e.g., International Data Base Conference 2015 edition, International Data Base Conference 2016 edition, etc.).

Research activity evaluation

One of the publication authors is identified as main author, and she belongs to a specific university department. Each department is part of a campus, and each university consists of one or more campuses. Furthermore, universities are divided by size (small, medium, large, depending on the number of their researchers). Each specific department is also characterized by a scientific sector of interest. For instance, the Department of Computer and Control Engineering is characterized by the ING-INF/05 scientific sector, belongs to the Cittadella Politecnica campus, which is part of the Politecnico di Torino university.

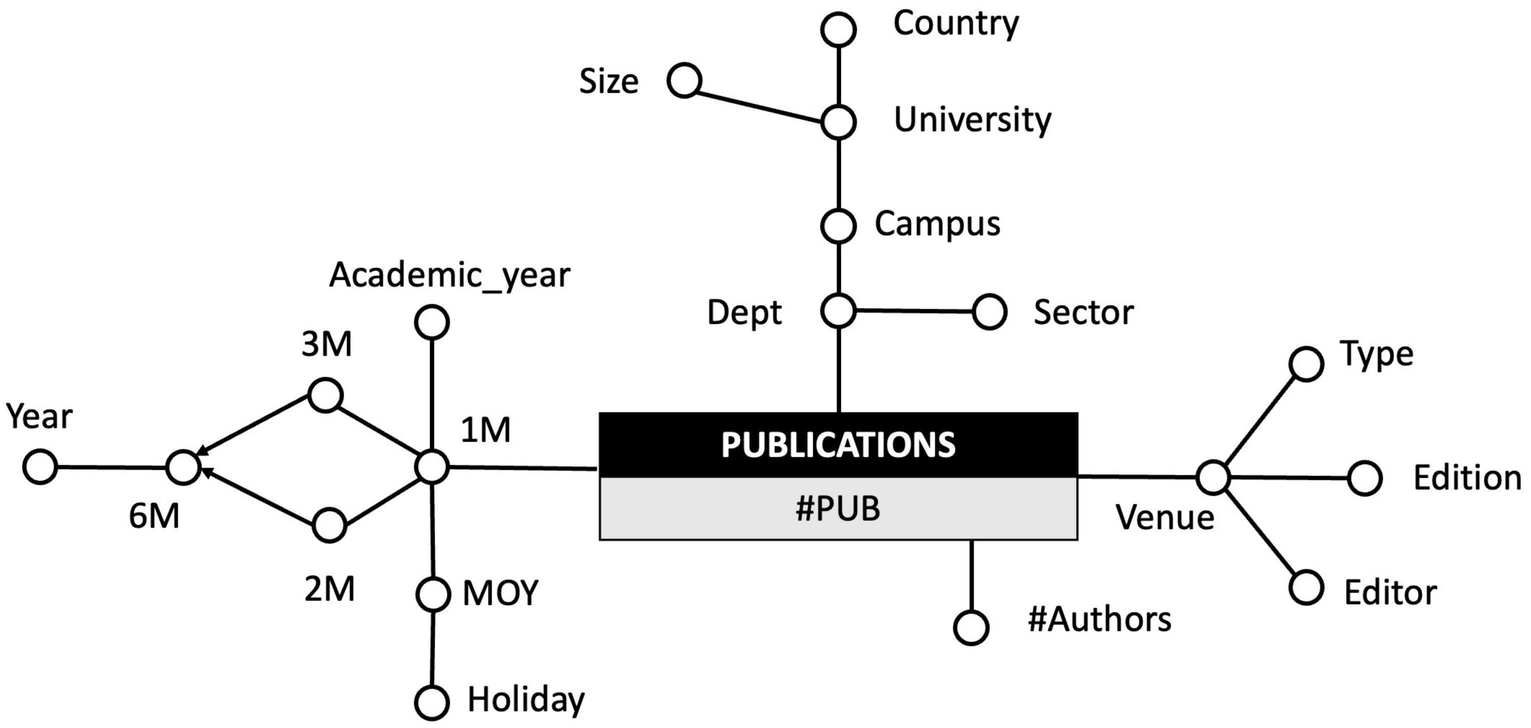
European Union: research activity

The European Union is interested in analyzing the number of publications according to the following dimensions:

- month, 2-month period, 3-month period, semester, year of publication;
- academic year (from September to August);
- holiday months (i.e., July and August of all years, when no teaching activities are provided);
- month of the year;
- main author's department, campus, and university;
- University's size and European country;
- Department's scientific sector;
- number of authors (from 1 to 10 all integer values, then a single value for more than 10 authors), publication venue (conference, journal, etc.), edition (year of the conference or year of the journal publication), editor, publication type (conference paper, journal article, book chapter, etc.).

Design

1. Design the data warehouse, including both the conceptual model and the fact and dimension tables, to address the given specifications. The data warehouse must also allow efficient execution of the following queries.
2. Write the following two queries using the extended SQL language.
   a. For each European university, each publication type, and each year, compute the monthly average number of publications, and the yearly cumulative total of publications. Consider only months when there is at least a publication.
   b. For each year, compute the percentage of publications of each department with respect to the total of its university. Globally rank all departments of all universities, separately for each year, and by number of total publications (the first in rank is the department with the highest number).



TIME (<u>TimeID</u>, Month, 2M, 3M, 6M, Year, AY, MoY, Holiday)
VENUE (<u>VenueID</u>, VenueName, Editor, Type, Edition)
DEPARTMENT (<u>DeptID</u>, Dept, Campus, University, Country, Size, Sector)
PUBBLICATIONS (<u>VenueID</u>, <u>DeptID</u>, <u>TimeID</u>, <u>#Authors</u>, #Pub)

Query a

1. Design the data warehouse, including both the conceptual model and the fact and dimension tables, to address the given specifications. The data warehouse must also allow efficient execution of the following queries.
2. Write the following two queries using the extended SQL language.
   a. For each European university, each publication type, and each year, compute the monthly average number of publications, and the yearly cumulative total of publications. Consider only months when there is at least a publication.
   b. For each year, compute the percentage of publications of each department with respect to the total of its university. Globally rank all departments of all universities, separately for each year, and by number of total publications (the first in rank is the department with the highest number).

Query a

```
SELECT University, Type, Year,
SUM(#Pub) / COUNT ( DISTINCT (Month)) AS Monthly_AVG,
SUM(SUM(#Pub)) OVER (PARTITION BY University, Type
                                         ORDER BY Year,
                                          ROW UNBOUNDED PRECEDING)
FROM TIME T, DEPARTMENT D, PUBLICATIONS P
WHERE T.TimeID = P. TimeID AND D.DeptID = P.DeptID
GROUP BY University, Type, Year
```

Query b

1. Design the data warehouse, including both the conceptual model and the fact and dimension tables, to address the given specifications. The data warehouse must also allow efficient execution of the following queries.
2. Write the following two queries using the extended SQL language.
   a. For each European university, each publication type, and each year, compute the monthly average number of publications, and the yearly cumulative total of publications. Consider only months when there is at least a publication.
   b. For each year, compute the percentage of publications of each department with respect to the total of its university. Globally rank all departments of all universities, separately for each year, and by number of total publications (the first in rank is the department with the highest number).

Query b

```
SELECT Year, Dept, University
100*SUM(#Pub) / SUM(SUM(#Pub)) OVER (PARTITION BY University, Year)
RANK () OVER (PARTITION BY Year
                    ORDER BY SUM (#Pub) DESC)
FROM TIME T, DEPARTMENT D, PUBLICATIONS P
WHERE T.TimeID = P. TimeID AND D.DeptID = P.DeptID
GROUP BY Year, DeptID, Dept, University
```

Materialized View

3. Create and maintain a materialized view with the ORACLE’s commands CREATE MATERIALIZED VIEW and CREATE MATERIALIZED VIEW LOG
   a. Consider the following queries of interest:
      I. Considering only the universities located in Italy and France, and the publication period 2013-2015, show the total number of publications for each university campus and semester.
      II. Considering only the "Information Systems Processing" scientific disciplinary sector, show the total number of publications for each pair (year, University).
      III. Show for each quarter (i.e., 3-months) and State the total number of publications, separately by scientific disciplinary sector.

Queries of interest

Query a

```
SELECT SUM(#Pub)
FROM PUBLICATIONS P, TIME T, DEPARTMENT D
WHERE (Country = «Italy» OR Country = «France») AND Year >= 2013
AND Year <= 2015 AND P.TimeID = T.TimeID AND P.DeptID = D.DeptID
GROUP BY Campus, 6M
```

Query b

```
SELECT SUM(#Pub)
FROM PUBLICATIONS P, TIME T, DEPARTMENT D
WHERE Sector = "Information Systems Processing”
AND P.TimeID = T.TimeID AND P.DeptID = D.DeptID
GROUP BY University, Year
```

Queries of interest

Query c

```
SELECT SUM(#Pub)
FROM PUBLICATIONS P, TIME T, DEPARTMENT D
WHERE P.TimeID = T.TimeID AND P.DeptID = D.DeptID
GROUP BY 3M, Country, Sector
```

Materialized View – exercise 3.b

b. Define the SQL query for a materialized view defined with the CREATE MATERIALIZED VIEW command, in order to reduce the response time of the queries listed in point 3a. Use the following syntax:

```
CREATE MATERIALIZED VIEW VM1
BUILD IMMEDIATE
RERESH FAST ON COMMIT
ENABLE QUERY REWRITE
AS  Query
```

Materialized view - exercise 3.b

```
CREATE MATERIALIZED VIEW VM1
BUILD IMMEDIATE
REFRESH FAST ON COMMIT
---enable query rewite
AS
SELECT Campus, 6M, Country, Year, University, 3M, Sector, SUM(#Pub) AS
TOTPub
FROM PUBLICATIONS P, TIME T, DEPARTMENT D
WHERE P.TimeID = T.TimeID AND P.DeptID = D.DeptID
GROUP BY Campus, 6M, Country, Year, University, 3M, Sector
```

GROUP BY **Campus**, 6M, Country, Year, University, **3M**, **Sector**

In bold: attributes whose combination determines the cardinality of the view

Materialized View – exercise 3.c

c. Define the logs of the materialized view using the CREATE MATERIALIZED VIEW LOG command (see example below), for each table where it is deemed necessary. For which tables is it useful to keep track of logs? Identify all and only the necessary tables. Furthermore, for each table identify all and only the attributes for which it is necessary to keep track of changes.

```
CREATE MATERIALIZED VIEW LOG on TableName
WITH SEQUENCE, ROWID
(List of Attributes)
INCLUDING NEW VALUES;
```

Materialized view – exercise 3.c

```
CREATE MATERIALIZED VIEW LOG ON
PUBLICATIONS
WITH SEQUENCE, ROWID
(TimeID, DeptID, #Pub)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON
TIME
WITH SEQUENCE, ROWID
(TimeID, 3M, 6M, Year)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON
DEPARTMENT
WITH SEQUENCE, ROWID
(DeptID, Campus, Sector, University, Country)
INCLUDING NEW VALUES;
```

Materialized View- exercise 4.a

4. View update and management using Triggers (assuming that the CREATE MATERIALIZED VIEW command is not available). Create the materialized view defined in point 3 and define the update procedure starting from changes on the fact table using a trigger.

   More specifically,

   a. Write the SQL statement to properly populate the table created with the statement

      CREATE TABLE VM1 (...) in the statement

      INSERT INTO VM1 (...) (**SELECT ... ...**)

Materialized view – exercise 4.a

```
CREATE TABLE VM1
Campus varchar(20) (check Campus IS NOT NULL),
6M_D DATE (check 6M_D IS NOT NULL),
Country varchar(20) (check Country IS NOT NULL),
Year_D DATE (check Year_D IS NOT NULL),
University varchar(20) (check University IS NOT NULL),
3M DATE (checl 3M_D IS NOT NULL),
Sector varchar(20) (check Sector IS NOT NULL),
TOTPub INTEGER);


INSERT INTO VM1(Campus, 6M_D, Country, Year_D, University, 3M_D, Sector,
TOTPub)
(SELECT---query defined in instruction of CREATE MATERIALIZED VIEW)
```

Materialized View- exercise 4.b

4. View update and management using Triggers (assuming that the CREATE MATERIALIZED VIEW command is not available). Create the materialized view defined in point 3 and define the update procedure starting from changes on the fact table using a trigger.

   More specifically,

   a. Write the SQL statement to properly populate the table created with the statement

      CREATE TABLE VM1 (...) in the statement

      INSERT INTO VM1 (...) (**SELECT ... ...**)

   b. Write the trigger to propagate the modifications (inserting a new record) made in the FACTS table to the materialized view VM1.

Trigger – exercise 4.b

```
CREATE OR REPLACE TRIGGER TriggerOfVM1
AFTER INSERT ON PUBLICATIONS
FOR EACH ROW
DECLARE
Var3M DATE; Var 6M DATE;  VarYear DATE;
VarCampus varchar(20); VarUniversity varchar(20);  VarCountry  varchar(20);
VarSector  varchar(20);
BEGIN
SELECT 3M, 6M, Year INTO Var3M, Var 6M, VarYear
FROM TIME
WHERE TimeID = :NEW.TimeID;


SELECT Campus, University, Country, Sector, INTO VarCampus, VarUniversity,
                                                VarCountry, VarSector
FROM DEPARTMENT
WHERE DeptID = :NEW.DeptID;
```

Trigger – exercise 4.b

```
SELECT COUNT(*) INTO N
FROM VM1
WHERE Campus = varCampus AND Sector = varSector AND AND 3M_D =
var3M [AND **Country = varCountry** AND **6M_D = Var6M** AND **University =
varUniversity** AND **Year_D = varYear ------ in bold redundant conditions
that can be omitted]**


IF (N>0) THEN
        UPDATE VM1
        SET TOTPub = TOTPub + :NEW.#Pub
        WHERE Campus = varCampus AND Sector = VarSector AND
        3M_D = var3M **[**AND **University = varUniversity** AND **Country =
        varCountry** AND **6M_D = Var6M** AND **Year_D = varYear ----- in
        bold redundant conditions that can be omitted]**
ELSE
```

Trigger – exercise 4.b

```
ELSE
            INSERT INTO VM1(….)
            VALUES (varCampus, var6M, varCountry, varYear,
            varUniversity, var3M, varSSD, NEW.#Pub);
END IF;
END;
```

[Specify an insert operation into the PUBLICATION table]