Politecnico di Torino
Database Management Systems

July 8^{th} 2011

1. (6 Points) The following relations are given (primary keys are underlined):

   USER(UId, Name, Surname, City, State, BirthDate)
   PHOTO(PId, UId, Resolution, KBs)
   UPLOAD(PId, Date, Time, Description)
   TAG(PId, Tag)

   Assume the following cardinalities:

   - card(USER)= 10^6 tuples,
     MIN(BirthDate) = 1-1-1941, MAX(BirthDate) = 31-12-1990,
     number of City $\simeq 10^2$,
     number of State $\simeq 10$,
   - card(PHOTO)= 10^8 tuples,
     number of Resolution $\simeq 10$,
     MIN(KBs) = 10^2, MAX(KBs) = $2 \cdot 10^3$,
   - card(UPLOAD)= $5 \cdot 10^8$ tuples,
     MIN(Date) = 01-01-2010, MAX(Date) = 31-12-2010,
   - card(TAG)= 10^9 tuples

   Furthermore, assume the following reduction factor for the group by condition:

   - having count(distinct tag) $\leq$ 10 $\simeq \frac{99}{100}$.

   Consider the following SQL query:

```
select UId, Name, Surname
from USER U, PHOTO P
where U.UId=P.PId and BirthDate > 1-1-1981
      and City ='Rome'
      and PId NOT IN (select UP.PId
                      from UPLOAD UP, TAG T, PHOTO P1
                      where UP.PId=T.PId and UP.PId=P1.PId
                      and Date ≥ 01-06-2010 and Date ≤ 30/06/2010
                      and Resolution <> '1280x720'
                      group by UP.PId
                      having count(distinct Tag) ≤10);
```

   For the SQL query:

   (a) Report the corresponding algebraic expression and specify the cardinality of each node (representing an intermediate result or a leaf). If necessary, assume a data distribution. Also analyze the group by anticipation.

   (b) Select one or more secondary physical structures to increase query performance. Justify your choice and report the corresponding execution plan (join orders, access methods, etc.).

2. (7 Points) The following relations are given (primary keys are underlined, optional attributes are denoted with *):

```
PERSON (RegNumber, Job)
SHIFT (RegNumber, Date, TCode)
SHIFT-TYPE (TCode, StartTime, Duration)
LEAVE-REQUEST(RCode, RegNumber, Date)
NOTIFICATION(RegNumber, Date, RequestOutcome)
```

Write a trigger to manage one-day leave requests by people working in a hospital (insert into the `LEAVE-REQUEST` table). A leave request is accepted if the person requesting it is not on duty in the requested date (`SHIFT` table). If instead the person is on duty, the request is accepted only if another person is available to fill the requester's shift. Otherwise, the request is declined. A person may substitute another person on a shift if they both have the same job and the substitute is not on duty in the same date.

The outcome of the request (accepted or declined) must be notified by means of an insert into the `NOTIFICATION` table.

3. Data Warehouse design

Problem specifications

An international software development company computes statistics about the code developed in its offices for different software applications.

Each office has its own management system for the developed code (i.e., repository), in which the developers share the projects.

Developers can check out the latest available version of the code from the repository and, after applying the required changes, they update the repository with the revised code version (commit operation).

When a commit is performed, the repository keeps track of:

- the developer who performed commit,
- the number of code rows which have been modified, added and removed,
- the changed files,
- the type of change that has been made (it can be a combination of *bugfix*, *new feature*, *refactor* and *improvement*, each one being either true or false),
- the date and the time of commit operation.

Moreover, the repository assigns, after each commit, a unique and incremental revision identifier which allows developers to know when the code has been updated. Possible conflicts due to concurrent updates are locally solved by the developers before the commit operation, so that the last revision always includes all the previous changes.

Developers are grouped in development teams. Each developer belongs to only one team, and each team refers to a particular company office. Furthermore, each developer has a particular role, e.g., *tester*, *performance*, etc.

The projects are composed of different distinct modules. The files are grouped in packages, and each module comprises different packages. Each application in the company catalog is composed of a set of projects. Each project is included in only one application. The developers work on different projects at the same time (possibly on all the projects).

Code revisions are grouped in milestones. The milestones can be categorized, according to the maturity level reached by the code, as stable version, release candidate, beta release, or build. Moreover, each milestone belongs to a major release.

The company wants to build a data warehouse to analyze the activities of software development, collecting available information stored in the databases of its offices. It should be possible to analyze the total number of commit operations, the total number of added code rows, the total number of removed code rows, the total number of modified code rows, and the average number of changed code rows (added, removed or modified) for each commit according to:

- the day of the week (Monday-Sunday), the day of the month (1-31), the working days or the holidays,
- the date, the month, the two-month period, the quarter, the year,
- the developer, the role of the developer, the development team, the office and the nation in which the office is located,

- the application, the project, the module, the package, the file,
- the milestone, the maturity level of the code, the major release,
- the commit type (*bugfix*, *new feature*, *refactor*, *improvement*).

The data warehouse will store information about years 2006-2010. Moreover, the following statistics are known (the candidate may estimate missing information that she deems relevant):

- the company has approximately 2000 developers;
- a team is composed on average of 10 developers;
- the developers are divided according to 5 different roles;
- the company has 20 offices in 10 nations;
- the repositories of the offices include around 500 milestones, 10,000 files, 1000 packages, 500 modules, 100 projects and 20 applications.

The following are some of the frequent analyses the company is interested in:

(a) For each project and year, select the daily average of the number of code rows which have been added by developers, separately for each developer role.

(b) For each team and month, select the average number of code rows which have been added by each commit operation, separately for each combination of commit types (*bugfix*, etc.).

(c) For each month and team, select the percentage of commits made by each developer with respect to the total number of commits made by her team. Consider for the analysis only commits related to a *bugfix*.

Design

(a) (7 Points) Design the data warehouse to address the described issues. In particular, the designed data warehouse must allow efficient execution of all the queries described in the specifications.

(b) (4 Points) Write frequent query (a) of the "problem specifications" using the extended SQL language.

(c) (*Optional:* 5 Points) Write frequent query (c) of the "problem specifications" using the extended SQL language.