Technologies for Information Systems

Part I (10 points)

prof. L. Tanca – January 19th, 2018

Available time: 25 minutes

Last Name _____

First Name _____

Student ID _____ **Signature**_____

1. Briefly define pervasive data management and the main problems that must be solved.

2. Describe the main differences between materialized and virtual data integration, explaining which of these two approaches is used in the case of Data Warehousing and why.

- **During this part of the exam, students are not allowed to use books or notes.**
- **Students should answer the theoretical questions using their own words, in order for the teachers to be able to assess their real level of understanding.**

Technologies for Information Systems

Part II (23 points)

prof. L. Tanca – January 19th, 2018

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliCourses is an online platform that offers courses and related exercises. Exercises are of different types, and each has a maximum score. The score obtained when solving an exercise is computed as the fraction of the maximum score corresponding to the level of completion, plus a possible bonus assigned by the system (e.g., because the student solved the exercise quickly). A student can try each specific exercise only once.
The management of PoliCourses has now hired you to design a data warehouse to analyze the results of the exercises performed by the students.

The following is the schema of the operational database used by PoliCourses:

UNIVERSITY (UniversityName, Country, NumOfStudents, Rector)

STUDENT (StudentId, Name, Surname, BirthDate, Nationality, UniversityName*) *// This table contains the students registered to the system. Registered students may be affiliated to a university (UniversityName is null if this is not the case).*

COURSE (CourseId, CourseTitle, Area) *// Sample areas: computer science, mathematics, physics, …*

EXERCISE (ExerciseId, Title, Text, Type, Topic, MaximumScore, CourseId) *// Sample types: multiple choice, check all that apply, fill in the blank, … Sample topics: database queries, derivatives, quantum mechanics, …*

EXERCISERESULT (ExerciseId, StudentId, Date, PercentageOfCompletion, Bonus)
// PercentageOfCompletion is a number between 0 and 100. The score obtained by the student for the performed exercise is computed applying the percentage of completion to the maximum score of that exercise, and then summing the bonus.

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

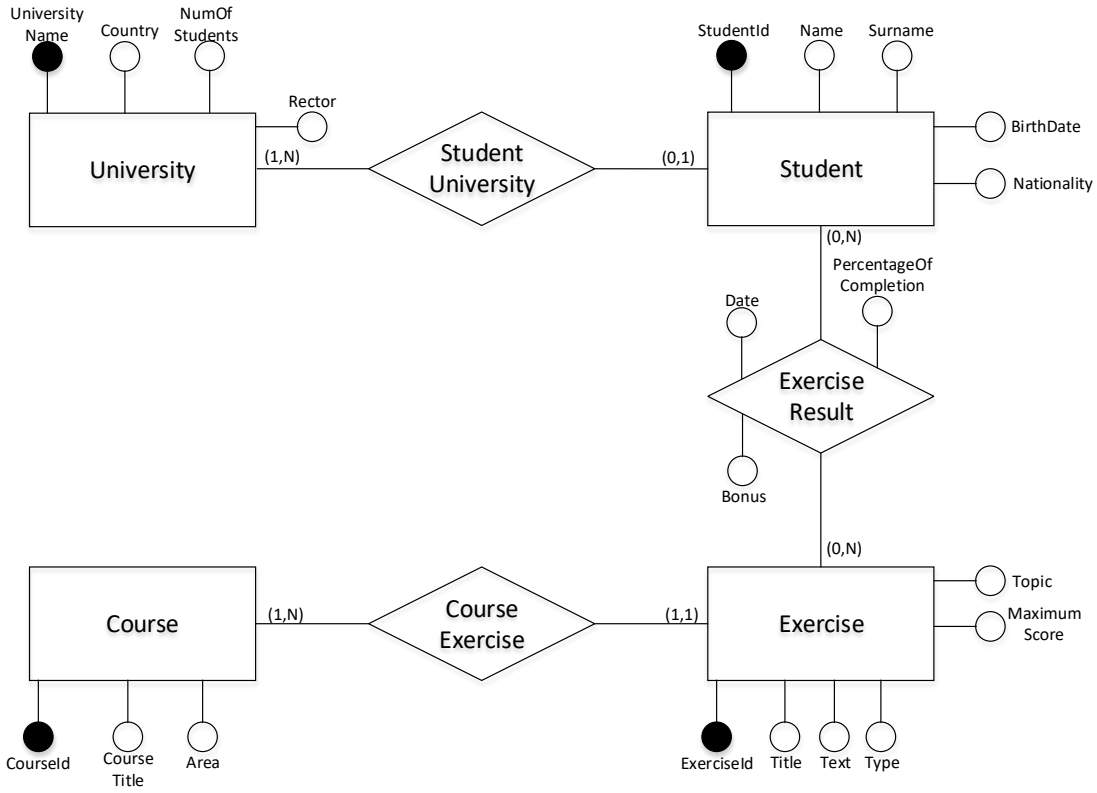
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2.5 points) Produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (2 points) Compute the total number of solved exercises for each university, type of exercise and area of the course. Include in the answer also the aggregations computed using only one and two of the three attributes.
    b. (2.5 points) Compute the average bonus assigned on Monday to students born in 1994 for each course (specify id and title), topic and country of the university.
    c. (2 points) Aggregate the total score by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).
    d. (2 points) Identify the Italian student(s) who have obtained the greatest total score (specify id, name and surname).

SOLUTION

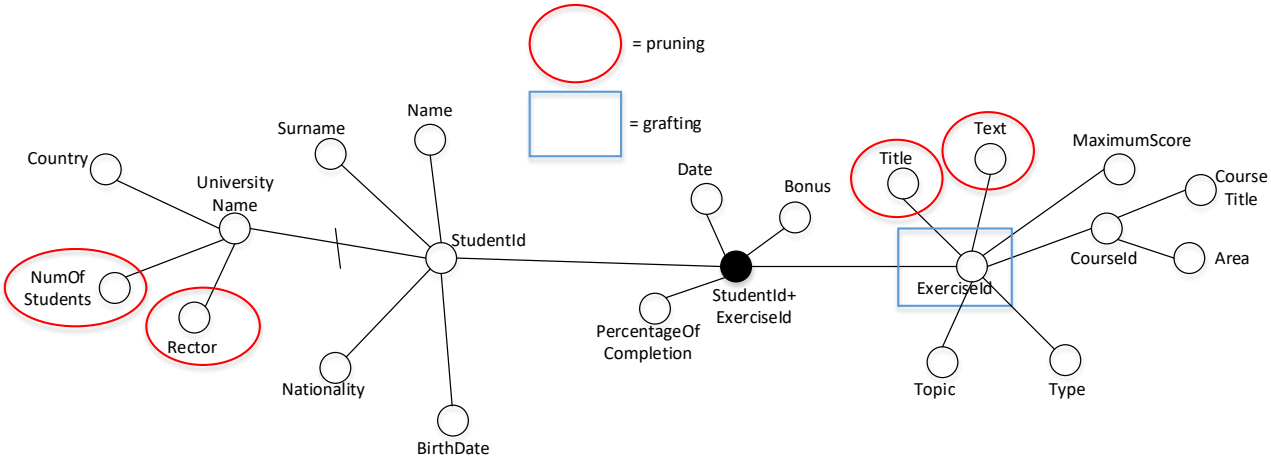
1. Reverse engineering



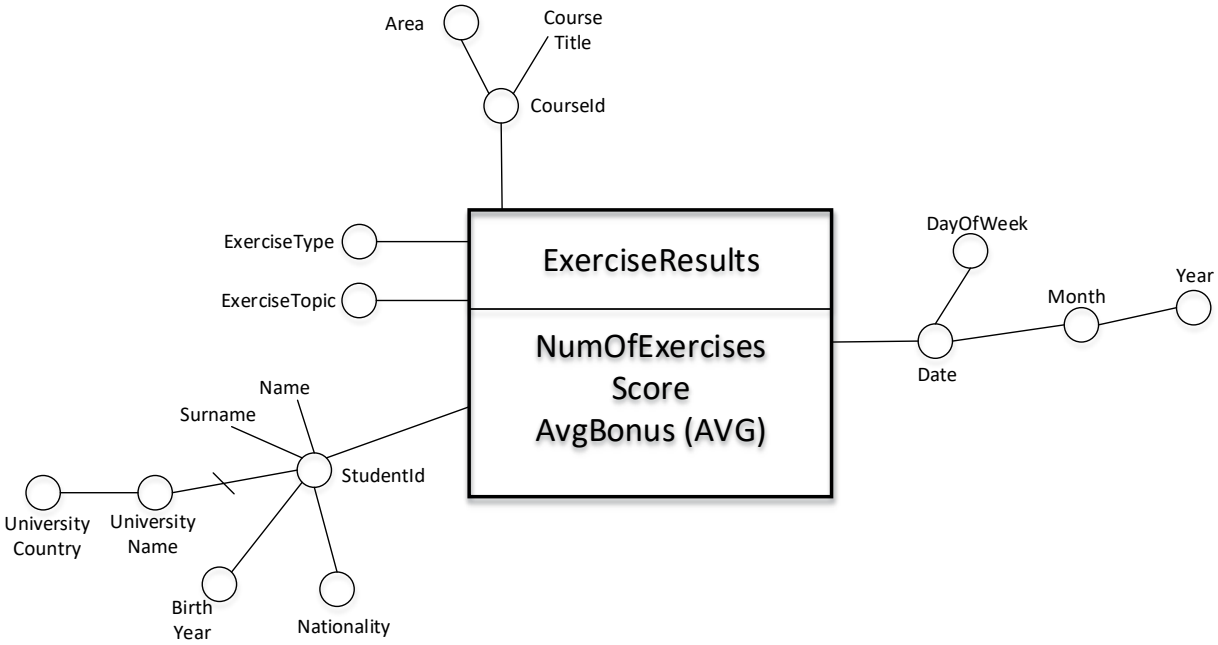
2. Conceptual design

Fact: ExerciseResults (ExerciseResult relationship)

2a) Attribute tree



2b) Fact schema



2c) Glossary

NumOfExercises

SELECT ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId, **COUNT**(*)
FROM ExerciseResult **AS** ER, Exercise **AS** E
WHERE ER.ExerciseId=E.ExerciseId
GROUP BY ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId

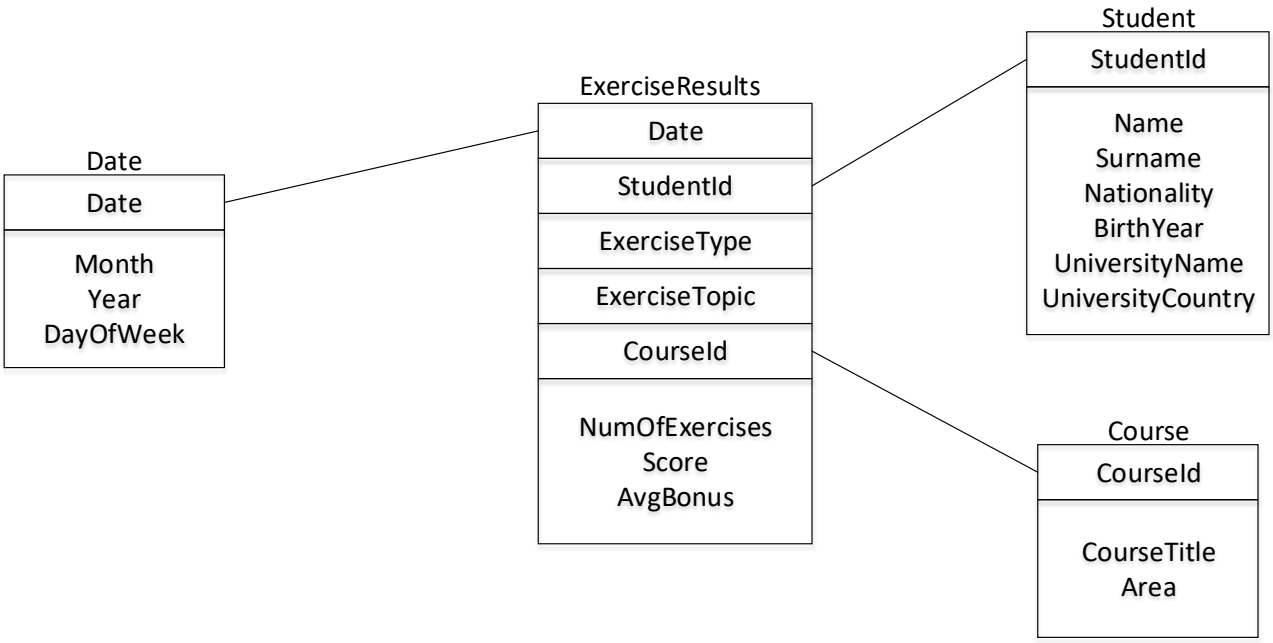
Score

SELECT ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId,
SUM(ER.Percentage*0.01*E.MaximumScore+ER.Bonus)
FROM ExerciseResult **AS** ER, Exercise **AS** E
WHERE ER.ExerciseId=E.ExerciseId
GROUP BY ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId

AvgBonus

SELECT ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId, **AVG**(ER.Bonus)
FROM ExerciseResult **AS** ER, Exercise **AS** E
WHERE ER.ExerciseId=E.ExerciseId
GROUP BY ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId

3. Logical design



4. Query answering

4a) Compute the total number of solved exercises for each university, type of exercise and area of the course. Include in the answer also the aggregations computed using only one and two of the three attributes.

```
SELECT S.UniversityName, E.ExerciseType, C.Area, SUM(E.NumOfExercises)
FROM ExerciseResults AS E, Student AS S, Course AS C
WHERE E.StudentId=S.StudentId AND E.CourseId=C.CourseId
GROUP BY S.UniversityName, E.ExerciseType, C.Area WITH CUBE
```

4b) Compute the average bonus assigned on Monday to students born in 1994 for each course (specify id and title), topic and country of the university.

```
SELECT C.CourseId, C.CourseTitle, E.Topic, S.UniversityCountry,
        SUM(E.NumOfExercises*E.AvgBonus)/SUM(E.NumOfExercises)
FROM ExerciseResults AS E, Student AS S, Course AS C, Date AS D
WHERE E.StudentId=S.StudentId AND E.CourseId=C.CourseId AND E.Date=D.Date AND
        D.DayOfWeek='Monday' AND S.BirthYear=1994
GROUP BY C.CourseId, C.CourseTitle, E.Topic, S.UniversityCountry
```

4c) Aggregate the total score by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).

```
SELECT D.Year, D.Month, D.Date, SUM(E.Score)
FROM ExerciseResults AS E, Date AS D
WHERE E.Date=D.Date
GROUP BY D.Year, D.Month, D.Date WITH ROLLUP
```

4d) Identify the Italian student(s) who have obtained the greatest total score (specify id, name and surname).

```
CREATE VIEW StudentTotalScore (StudentId, Name, Surname, Score) AS (
        SELECT S.StudentId, S.Name, S.Surname, SUM(E.Score)
        FROM ExerciseResults AS E, Student AS S
        WHERE E.StudentId=S.StudentId AND S.Nationality='Italy'
        GROUP BY S.StudentId, S.Name, S.Surname
)
SELECT StudentId, Name, Surname
FROM StudentTotalScore
WHERE Score = (         SELECT MAX(Score)
                        FROM StudentTotalScore
                )
```