Data Management for Big Data

September 6, 2019

2018–2019, 1st fall session
Teachers: Angelo Montanari, Dario Della Monica, and Paolo Gallo

Surname and name: ____

Student ID (matricola): __________ email: ____________________________

Each part of the exam will contribute equally (one third) to determine the final score. Thus, total score of each part might be normalized so that you get at most 10 points from each part. Teachers can assign extra bonus points at their own discretion.

Part I: Fundamentals of database systems

Data Management for Big Data - Exam 6 September 2019

Exercise 1:

Let us consider the following relational schema about actors and films:

$FILM(FilmCode, Title, Filmmaker, Year)$;

$ACTOR(ActorCode, Surname, Name, Sex, BirthDate, Nationality)$;

$INTERPREATATION(Film, Actor, Role)$

Let every film be univocally identified by a code and characterized by a title, a filmmaker, and the year when it has been released. For the sake of simplicity, let as assume each film to be directed by a single filmmaker and each filmmaker to be univocally identified by his/her surname. Let us consider the possibility that distinct films have the same title (this is the case, for instance, with remakes). Let every actor be univocally identified by a code and characterized by a name, a surname, a sex, a birth date, and a nationality. Let us assume that more than one actor may act in a film and that the same actor may act in more than one film. For the sake of simplicity, let us assume that, in a given film, each actor may play one role only.

Define preliminarily primary keys, other candidate keys (if any), and foreign keys (if any). Then, formulate an SQL query to compute the following data (exploiting aggregate functions only if they are strictly necessary):

- the film (the films if more than one) with the highest number of actors.

Exercise 2:

Let us sinthesize the ER conceptual schema of a database for the management of information about a set of cars on the basis of the following set of requirements.

- Every car is univocally identified by its licence plate and it is characterized by its model, its year of manufacture, its color, its market value, and its owner (there can be more than one owner). Among cars, we would like to distinguish the subset of vintage cars (we say that a car is a vintage one if it has been manufactored 25 years, or more, ago), the subset of sport cars, which are characterized by their maximum speed, and the subset of vintage sport cars.
- Each car model is characterized by a name, a manufacturer, and a cylinder capacity. The name univocally identifies the model among the models offered by the manufacturer (we do not exclude the possibility that different manufacturers offer models–obviously different–with the same name).
- Each manufacturer is univocally identified by its name and it is characterized by its foundation year and its set of plants. The same person can be the president of more than one manufacturer. Each plant is characterized by a name, that univocally identifies it among the plants of the manufacturer, a home city, and a number of employees.

Build an ER schema that describes the above requirements, clearly explaining any assumption you made. In particular, for each entity, identify its possible keys, and carefully specify the constraints associated with each relation.

Part II: Advanced database models, languages, and systems

Instructions for multiple-choice questions.

- A right answer is worth +1.
- A wrong answer is worth -0.33.
- Short explanations should be **very** short (no more than 2 lines) and should be written in a neat handwriting. They are optional and used **at teacher's discretion**, mainly to increase the score of a wrong answer; seldom, to lower the score of a right one.

Instructions for open questions.

- The score assigned to an open question can range from 0 to 3 points.
- No negative score is assigned to open questions.
- Be careful: use a neat handwriting.
- Try to be succinct also for open questions.

1. Let R_1 and R_2 be relations and τ be a predicate (e.g., "$age = 20$") over attributes of R_1. Consider the following equivalent relational algebra expressions:

   (i) $\sigma_\tau(R_1) \bowtie R_2$,

   (ii) $\sigma_\tau(R_1 \bowtie R_2)$.

   Which is the correct statement among the following:

   ☐ "pushing" the selection inside a join is always more efficient, i.e., (i) is more efficient than (ii)

   ☐ "pushing" the selection inside a join is always less efficient, i.e., (ii) is more efficient than (i)

   ☐ there is no general rule, i.e., the relative efficiency of (i) and (ii) depends on concrete instances of τ, R_1, and R_2

   Short explanation (optional): ______________________________

   ____

2. Let t_S = "time for one seek", t_T = "time for one-block transfer", and h be the height of a secondary index (a tree) over attribute A of relation R. Which is the estimated cost for accessing all tuples where $A = X$? (Assume that A is not a key and there are 5 tuples satisfying $A = X$.)

   ☐ $h * (t_T + t_S) + t_S + t_T$

   ☐ $h * (t_T + t_S) + t_S + 5 * t_T$

   ☐ $h * (t_T + t_S) + 5 * t_S + 5 * t_T$

   Short explanation (optional): ______________________________

   ____

   Hint: recall that
   - *a primary index is defined over the attribute(s) used to physically order the file in the filesystem;*
   - *a secondary index is defined over any (subset) of the other attributes.*

3. Select the correct statement.

   ☐ The *natural join* is not commutative

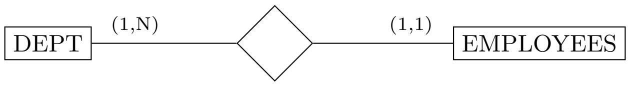
   ☐ The *natural join* is commutative but the order of operands affects efficiency

   ☐ The *natural join* is commutative and the order of operands is completely irrelevant

   Short explanation (optional): ____________________________________

4. Distributed DB Systems are a necessity for several reasons. They come with benefits and drawbacks. Name three such benefits.

5. Let DEPT(<u>*dept_code*</u>, *name*) and EMPLOYEES(<u>*cf*</u>, *name*, *dept_code*) be relations, where *dept_code* is a foreign key referring to attribute <u>*code*</u> of relation DEPT. Graphically:



   In other words, there is a link L with *owner(L)* = DEPT and *member(L)* = EMPLOYEES.

   Moreover, assume that DEPT is fragmented, according to primary horizontal fragmentation, into $\{\text{DEPT}_1, \ldots, \text{DEPT}_n\}$.

   Which is the induced derived horizontal fragmentation?

   ☐ $\{\text{DEPT}_i \ltimes \text{EMPLOYEES} \mid 1 \le i \le n\}$

   ☐ $\{\text{EMPLOYEES} \ltimes \text{DEPT}_i \mid 1 \le i \le n\}$

   ☐ $\{\text{DEPT}_i \bowtie \text{EMPLOYEES} \mid 1 \le i \le n\}$

   Short explanation (optional): ____________________________________

6. Consider the 2 transactions T_1 (over operations $R_1(y), R_1(x), W_1(x)$) and T_2 (over operations $W_2(x), W_2(y)$) formalized through the 2 following partial orders, respectively:

$$T_1 = \{R_1(x) \prec W_1(x), R_1(y) \prec W_1(x)\}$$
$$T_2 = \{W_2(y) \prec W_2(x)\}.$$

Is there a history over $\{T_1, T_2\}$ that is serializable but not serial? If yes, write down one such history. If not, write down a history that is both serializable and serial.

Is there a history over $\{T_1, T_2\}$ that is serial but not serializable? If yes, write down one such history. If not, write down a history that is both serializable and serial.

Hint: recall that serial histories are always concurrent transaction executions, that is, they are formalized through linear orders.

7. Explain the difference between checking that an XML document is well-formed and validating it.

Hint: which is the difference between an XML parser and an XML validator?

Part III: Data analysis and big data

1. With respect to the Big Data context, briefly describe the classification by source and level of structuring.

   Hint: "source" is the one that generates data

2. Briefly describe the available indexes in MongoDB.

3. Give some examples of the most suitable context for the usage of graph databases.

4. Briefly explain the difference between representing a graph using a *native* graph database versus a relational database.

> *Hint: Despite the graph can be represented using both solutions, performances in case of ...*

5. Consider a time series from a single sensor which is transmitting a speed value every second. Using a document database you can store a single JSON document for each reading:

   ```
   {
     pressID: "S8008",
     speed: 85,
     ts: ISODate("2018-11-10T22:56:25.00-0500")
    }
   ```

   alternatively, you can store a JSON document containing a nested object ready to store one hour of observations and update it as soon as a new measure comes (every second):

   ```
    {
       pressID: "S8008",
         speed:{
            0: {0:63, ..., 59:45},
            ...,
            59:{0:65, ..., 59:65}
           },
       ts: ISODate("2018-11-10T22:00:00.00-0500")
    }
   ```

   For each of the two models above write down:

   - How many document writes will typically occur in one hour?
   - How many documents updates will typically occur in one hour?
   - Suppose that each document will include a 100 bytes long field for indexing purposes, what will be the space used only by those fields for storing an hour of readings?

6. Point out some of the advantages of using Hadoop instead of HPC (High Performances Computing) approach.

> *Hint: HPC paradigm means that a single powerful node needs to access ALL the data to process.*

7. List some schemas usually employed into data-warehouse systems.

Hint: Just list the main schemas, no description needed.