Introduction to databases

Iniziato	giovedì, 24 giugno 2021, 09:59
Stato	Completato
Terminato	giovedì, 24 giugno 2021, 11:59
Tempo impiegato	2 ore
Valutazione	Non ancora valutato

Domanda 1

Risposta corretta

Punteggio ottenuto 1,00 su 1,00

A transaction is atomic if

- it takes the system from a valid state to another valid state
- none of the answers are correct
- it is executed on the system at the same time as other transactions as if it were the only one being executed
- it makes modifications permanent immediately after the transaction has ended
- all of the operations composing it are either completed, or they are undone, as if they had never been executed ✓

Risposta corretta.

La risposta corretta è:
all of the operations composing it are either completed, or they are undone,
as if they had never been executed

Domanda 2

Risposta corretta

Punteggio ottenuto 1,00su 1,00

A foreign key in a table

- may not be composed of a single element
- should, in turn, be declared as the primary key of another table
- none of the answers are correct ✗
- must belong to the primary key of the table in which it is defined
- may not be a composite key

Risposta errata.

La risposta corretta è:
should, in turn, be declared as the primary key of another table

Domanda 3

Risposta corretta

Punteggio ottenuto 1,00 su 1,00

The SQL command

```
CREATE TABLE T1
(A1 CHAR(5) NOT NULL,
A2 INTEGER,
A3 CHAR(5) NOT NULL,
PRIMARY KEY (A1, A3),
FOREIGN KEY (A2) REFERENCES T2
ON UPDATE SET NULL);
```

- (●) none of the answers are correct ✓
- ( ) creates a table T1 where, upon each UPDATE operation on T1, the foreign key values are reset to NULL
- ( ) creates a table T1 whose primary key is (A1, A3, A2)
- ( ) is incorrect because attribute A2 should be declared NOT NULL
- ( ) is incorrect because the attributes composing the primary key should be declared contiguously

Risposta corretta.

La risposta corretta è:
none of the answers are correct

Domanda 4

Completo

Punteggio max.: 1,00

The following relations are given (primary keys are underlined):

ATHLETE(ACode, AName, DateOfBirth, CityOfResidence)

TOURNAMENT(TCode, TName, Level, OrganizingAssociation)

RACE(RCode, RName, TCode, Date, CityOfRace)

ATHLETE-PARTICIPATES-IN-RACE(ACode, RCode, Ranking)

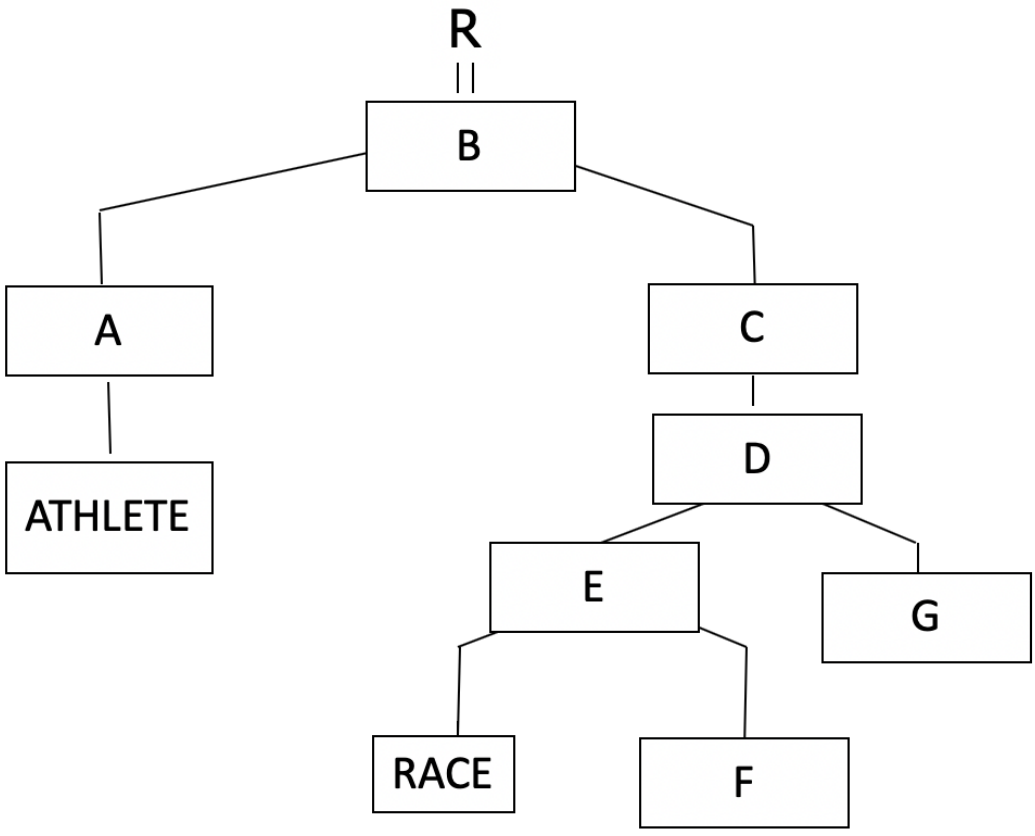
Write the following query in relational algebra:

Show the code and the name of each athlete who has never participated in races held in the city where she/he resides.

Instructions needed to complete the exercise:

The following query tree graphically represents the requested query. You are asked to indicate, for each of the boxes from A to G in the query tree, the relational table or the corresponding algebraic operator. Use the text box below to report the solution.

Note: each box in the query tree is associated with a single relational table or a single algebraic operator.



A PROJECTION(ACode,AName)

B, LEFT ANTI JOIN

C SELECTION (A1.CityOfResidence=RACE.CityOfRace)

D NATURAL JOIN

E NATURAL JOIN

F ATHLETE-PARTICIPATES-IN-RACE APR

G ATHLETE A1

Domanda 5

Completo

Punteggio max.: 1,00

The following relations are given (primary keys are underlined):

DANCER(SSN, Name, DateOfBirth, CityOfResidence, MainDanceType)

DANCE SCHOOL(DSCode, Name, City, ArtisticDirector)

BALLET(BCode, DSCode, Title, #Scene, Type, ChoreographerName)

DANCER-PARTICIPATES-IN-BALLET(SSN, BCode, DSCode, Role)

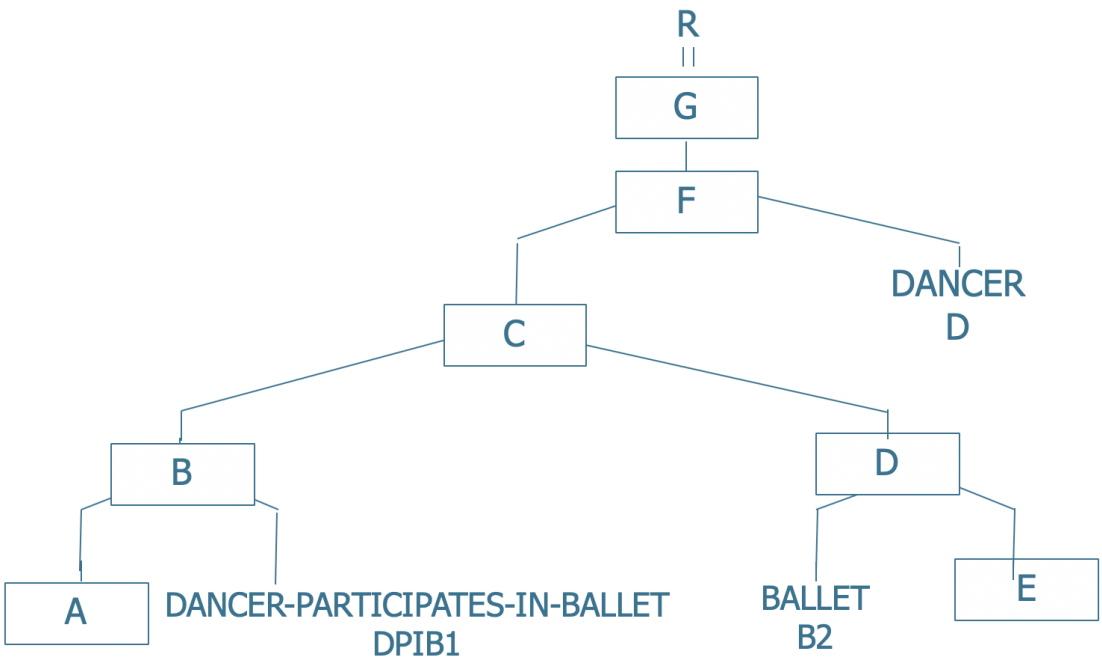
Write the following query in relational algebra:

Show the name and the city of residence of each dancer who has participated in at least two ballets of the same type.

Instructions needed to complete the exercise:

The following query tree graphically represents the requested query. You are asked to indicate, for each of the boxes from A to G in the query tree, the relational table or the corresponding algebraic operator. Use the text box below to report the solution.

Note: each box in the query tree is associated with a single relational table or a single algebraic operator.



A, BALLET B1

B, NATURAL JOIN

C, THETA JOIN

(DPIB1.SSN=DPIB2.SSN

AND B1.BCode<>B2.BCode

AND B1.Type=B2.Type)

D,NATURAL JOIN

E, DANCER-PARTICIPATES-IN-BALLET DPIB2

F, THETA JOIN (DPIB1.SSN=D.SSN)

G, PROJECTION(Name,CityOfResidence)

Domanda 6

Completo

Punteggio max.: 1,00

Given the following relational schema

E-BOOK (ISSN, AuthorId, Title, Language, Editor)

AUTHOR (AuthorId, Name, Surname, Nationality)

DOWNLOAD (Date, Hour, Minutes, UserId, ISSN)

USER (UserId, Name, Gender, DateOfBirth)

For each author of Italian nationality, show name and surname of the author and each editor with which the author has published at least 5 e-books.

```
SELECT Name,Surname, Editor
FROM AUTHOR A, E-BOOK E
WHERE Nationality="Italian"
   AND A.AuthorID=E.AuthorID
GROUP BY A.AuthorID, Name, Surname, Editor
HAVING COUNT(*)>=5
```

Domanda 7

Completo

Punteggio max.: 1,00

Given the following relational schema

E-BOOK (ISSN, AuthorId, Title, Language, Editor)

AUTHOR (AuthorId, Name, Surname, Nationality)

DOWNLOAD (Date, Hour, Minutes, UserId, ISSN)

USER (UserId, Name, Gender, DateOfBirth)

For each editor that has never published e-books written by German authors, show the editor, the total number of published e-books, and the total number of downloads.

```
SELECT E.Editor, COUNT(*) as PublishedEbook#, Total.TotalDownload#
FROM E-BOOK E ,
   (
   SELECT Editor, COUNT(*) as TotalDownload#
   FROM E-BOOK E2, DOWNLOAD D
   WHERE D.ISSN=E2.ISSN
   GROUP BY Editor
   ) AS Total
WHERE E.Editor NOT IN
   (SELECT Editor
    FROM E-BOOK E1, AUTHOR A
    WHERE E1.AuthorID=A.AuthorID AND
      A.Nationality="German"
   )
   AND E.Editor=Total.Editor
GROUP BY E.Editor, Total.TotalDownload#
```

Domanda 8

Completo

Punteggio max.: 1,00

Given the following relational schema

E-BOOK (ISSN, AuthorId, Title, Language, Editor)

AUTHOR (AuthorId, Name, Surname, Nationality)

DOWNLOAD (Date, Hour, Minutes, UserId, ISSN)

USER (UserId, Name, Gender, DateOfBirth)

For each user who has downloaded at least 3 different e-books in English, show the name, date of birth and number of (different) editors that published the books he/she downloaded

---

```
SELECT Name, DateOfBirth, COUNT(DISTINCT Editor) as Editors#
FROM USER U, DOWNLOAD D, E-BOOK E
WHERE U.UserId=D.UserId AND E.ISSN=D.ISSN
  AND U.UserID IN
   (SELECT UserId
    FROM DOWNLOAD D1, USER U1, E-BOOK E1
    WHERE U1.UserId=D1.UserID AND E1.ISSN=D1.ISSN
      AND E1.Language="English"
   GROUP BY U1.UserId
   HAVING COUNT(DISTINCT D1.ISSN)>=3
   )
GROUP BY U.UserId, Name, DateOfBirth
```

Domanda 9

Completo

Punteggio max.: 1,00

A gym would like to design a database to manage some of its activities.

Several trainers work at the gym. Each trainer is identified by her/his Social Security Number (SSN) and is characterized by name and surname, date of birth, education, e-mail, phone number (if available) and the list of sports activities the trainer is qualified to teach.

Several sport activities can be carried out at the gym. Activities are identified by a unique code and classified as musical activities, body building, or cardio-fitness. For each musical activity, the name, level (basic, intermediate, advanced) and the list of equipments used (e.g., weights, rope, step) are known. Keep track of the days of the week and of the room where each musical activity takes place, the starting and ending times, and the corresponding trainer. Please note that a trainer can not teach two different musical activities on the same day and time. However, the same musical activity may be carried out at the same time and day in different rooms. Furthermore, the same musical activity can be carried out on different days and many times on the same days in different time periods.

List the main characteristics (Entity Names, their Identifiers, attributes, relationships) of an E-R conceptual schema of the database for the above application.

ENTITY: TRAINER

Int ID: SSN

Attrib: name,surname, dateOfBirth, education,email. phoneNumber*

ENTITY: SPORT ACTIVITY

Int ID: CodeSport

RELATION: CAN_DO_ACTIVITIES

TRAINER(1,N)- SPORTACTIVITY(0,N)

SPORT ACTIVITY is a FATHER ENTITY with (t,e)

entity-child:BODY-BUILDING

entity-child:CARDIO-FITNESS

entity-child: MUSICAL ACTIVITIES

->attr: name, level, listOfEquipment(0,N)

ENTITY: DO-MUSICAL-ACTIVITIES

Int ID: Day, StartTime

Ext ID: SSN, CodeSport

Attrib: Place, EndTime

BINARY RELATION:

DO-MUSICAL-ACTIVITIES(1,1) - TRAINER(0,N)

DO-MUSICAL-ACTIVITES(1,1) - MUSICAL ACTIVITIES(0,N)

Domanda 10

Completo

Punteggio max.: 1,00

A gym would like to design a database to manage some of its activities.

Gym members are identified by the number of their membership card. Each member is characterized by a name, surname, date of birth, category (e.g., students, under 16, over 60), and validity time period. Members access the gym. For each access keep track of the corresponding member, date, entrance, and leaving times (e.g., from 10 am to 2 pm). Assume that each member may access the gym only once per date.

The gym offers different packages of activities. Packages are characterized by a code and a brief description. Packages can be purchased by gym members as a subscription, individual entrance, and special mix. For all types of packages, the price and the time slots at which members are allowed to access the gym are known. For subscriptions and special mix, the package duration (in months) is also known. Individual entrances are characterized by the type of entrance (e.g., simple, power).

The database stores all the activity packages purchased by gym members during the year. Members purchase packages and then associate a time period of usage (starting and ending date) in which the activities of the packages can be carried out at the gym. Assume that a member can associate different packages to the same time period, and the same package can be associated with different members to the same period.

List the main characteristics (Entity Names, their Identifiers, attributes, relationships) of an E-R conceptual schema of the database for the above application.

ENTITY: MEMBER

Int ID: MemberCard

Attr: name,surname,dateofbirth,category, validityTimePeriod

ENTITY:ACCESS

Int ID: Date

Ext ID: MemberCard REFERENCES MEMBER(Membercard)

Attribute: entraceTime, leavingTime

RELATIONSHIP: DO

ACCESS(1,1) - MEMBER(0,N)

ENTITY: TIME

Int id: Date

RELATION: BUY

PACKAGE(0,N)- MEMBER(1,N)- TIME(1,N)

Attrib_Enddate,

ENTITY father: PACKAGE

Int ID: Code

Attr: Description, price, timeSlots

HIERARCLY(t,e):

entity-child: individual

attr, typeOfEntrance

entity-child: subscription

attri, monthDuration

entity-child: specialMix

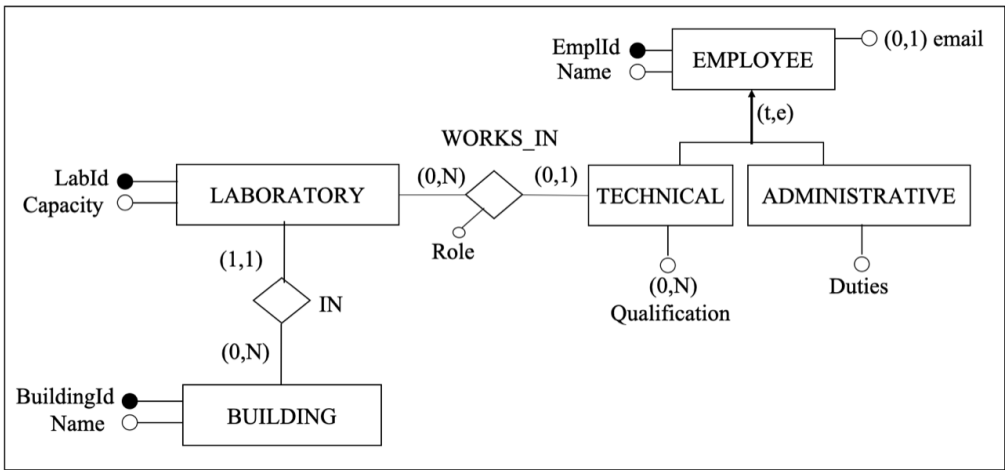
attr, monthDuration

Domanda 11

Completo

Punteggio max.: 1,00

Given the Entity-Relationship schema shown in the following picture



It is asked to:

1. Derive a normalized relational logical schema for the same database.
2. Define the referential integrity constraints for the 2 relationships defined in the Entity Relationship schema.

BUILDING(BuildingID, Name)
LABORATORY(LabId, Capacity, BuildingId)

EMPLOYEE(EmplId, Name, email*)

ADMINISTRATIVE(EmplId, Duties)

TECHNICAL(EmplId)

HAS-QUALIFICATION(EmplId, Qualification)

WORKS_IN(EmplId, LabId, Role)

1) FOREIGN KEY LABORATORY(BuildingId) REFERENCES BUILDING(BuildingId)

2) FOREIGN KEY WORKS_IN(EmplId) REFERENCES TECHNICAL(EmplId)
   FOREIGN KEY WORKS_IN(LabId) REFERENCES LABORATORY(LabId)

Domanda 12

Completo

Punteggio max.: 1,00

The following relations are given (primary keys are underlined).

SUPERMARKET(SupermarketCode, City, NumberOfPlacesInQueue, OpeningTime, ClosingTime)

ENTRANCE_QUEUE(SupermarketCode, CustomerSSN)

ENTRANCE_REQUEST(RequestCode, SupermarketCode, CustomerSSN, Date, Time)

We would like to automatically manage the entrance queues in a chain of supermarkets.

Write the trigger needed to manage the following activity.

When a customer wants to enter in the queue for a specific supermarket, a new record is inserted into the ENTRANCE_REQUEST table.

Write the trigger to check the availability of a place in the queue for the requested supermarket.

The ENTRANCE_QUEUE table contains the customers already queued in each supermarket. Customers are queued until the maximum number of places is reached.

The maximum number of places available in the queue for a supermarket is equal to the value of the NumberOfPlacesInQueue attribute of the SUPERMARKET table. If there is no place available in the queue for the requested supermarket, the trigger ends with an error. Otherwise the customer is put in the queue.

Tip: Use the raise_application_error (....) function to report an error. It is not required to specify the parameters passed to the function.

```
create trigger CHECK_INSERT_ENTRANCE_REQUEST
after insert ON ENTRANCE_REQUEST
FOR EACH ROW

DECLARE
NumberQueue Number;
Counter Number;

BEGIN

SELECT NumberOfPlacesInQueue INTO NumberQueue
FROM SUPERMARKET
WHERE SupermarketCode= :NEW.SupermarketCode;

SELECT COUNT(*) INTO Counter
FROM ENTRANCE_QUEUE
WHERE SupermarketCode= :NEW.SupermarketCode;

IF(Counter=NumberQueue)
   THEN raise_application_error();
```

```
END IF;

INSERT INTO ENTRANCE_QUEUE
VALUES (:NEW.SupermarketCode, :NEW.CustomerSSN);

END;
```

Domanda 13

Completo

Punteggio max.: 1,00

The following relations are given (primary keys are underlined).

SUPERMARKET(SupermarketCode, City, NumberOfPlacesInQueue, OpeningTime, ClosingTime)

ENTRANCE_QUEUE(SupermarketCode, CustomerSSN)

ENTRANCE_REQUEST(RequestCode, SupermarketCode, CustomerSSN, Date, Time)

We would like to automatically manage the entrance queues in a chain of supermarkets.

Write the trigger needed to manage the following activity.

Integrity constraint on the duration of the opening

There can be at most 10 supermarkets in Turin with a duration of the opening (difference between ClosingTime and OpeningTime attributes in the SUPERMARKET table) greater than 18 hours.

Any modification of the SUPERMARKET table that causes the constraint violation must not be executed.

Carefully evaluate all the triggering events on table SUPERMARKET.

Solution guidelines: Given the following template of the trigger solution, you are asked to complete the bold parts (i.e. parts A and B)

```
create trigger CheckOpeningTime
A
declare
X number;
begin
        B
        if (X > 10) then
                raise_application_error(...);
        end if;
end;
```

A, BEFORE INSERT OR UPDATE ON SUPERMARKET

B,

```
SELECT COUNT(*) INTO X
FROM SUPERMARKET
WHERE City="Turin" AND ClosingTime-OpeningTime>18;

IF( (OLD.SupermarketCode IS NOT NULL AND OLD.ClosingTime-OLD.OpeningTime<=18
AND NEW.ClosingTime-NEW.OpeningTime>18)
    OR (OLD.SupermarketCode IS NULL AND NEW.ClosingTime-NEW.OpeningTime>18) )
  X++;
END IF;
```