Introduction to databases

Iniziato	domenica, 4 luglio 2021, 13:21
Stato	Completato
Terminato	domenica, 4 luglio 2021, 13:27
Tempo impiegato	5 min. 30 secondi
Valutazione	Non ancora valutato

Domanda 1

Risposta corretta

Punteggio ottenuto 1,00 su 1,00

A transaction is atomic if

- [✓ all of the operations composing it are either completed, or they are undone, as if they had never been executed] ✓
- it makes modifications permanent immediately after the transaction has ended
- none of the answers are correct
- it takes the system from a valid state to another valid state
- it is executed on the system at the same time as other transactions as if it were the only one being 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 1,00 su 1,00

A foreign key in a table

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

Risposta corretta.

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);
```

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

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(<u>ACode</u>, AName, DateOfBirth, CityOfResidence)

TOURNAMENT(<u>TCode</u>, TName, Level, OrganizingAssociation)

RACE(<u>RCode</u>, RName, TCode, Date, CityOfRace)

ATHLETE-PARTICIPATES-IN-RACE(<u>ACode</u>, <u>RCode</u>, Ranking)

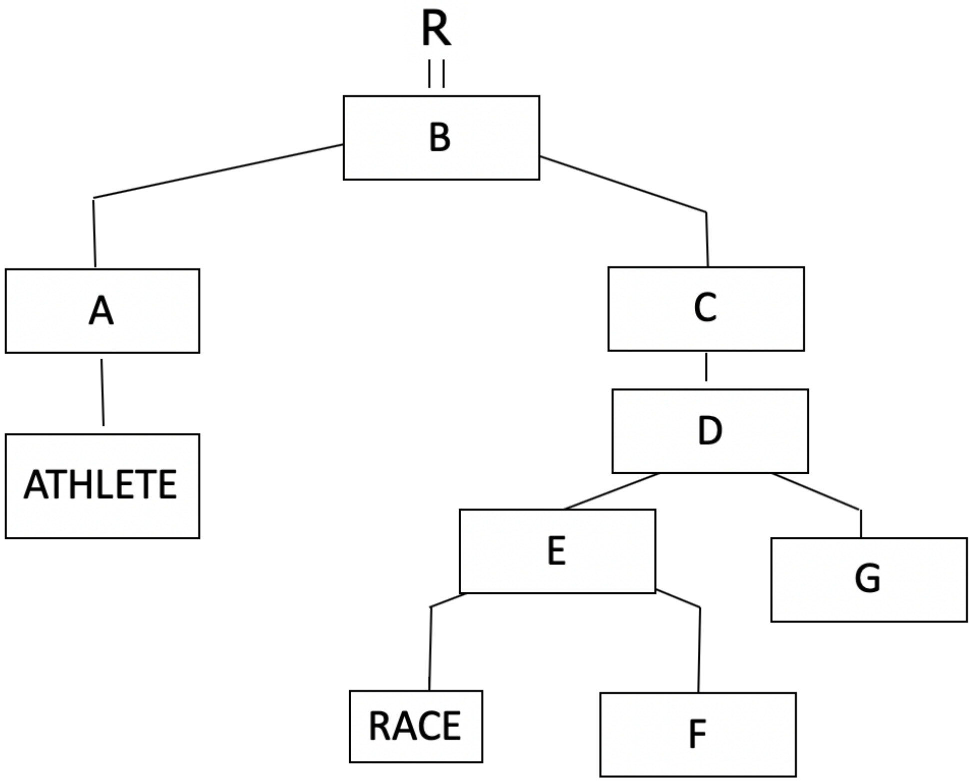
Write the following query in relational algebra:

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: Athlete.ACode, Athlete.Aname

B. projection: Athlete.ACode, Athlete.Aname

C. projection: A.ACode, A.Aname

D. Anti-join: A.CityofResidence != Race.CityofRace

E. Natural join

F. ATHLETE-PARTICIPATES-IN-RACE APR

G. Athlete A

Domanda 5

Completo

Punteggio max.: 1,00

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

DANCER(<u>SSN</u>, Name, DateOfBirth, CityOfResidence, MainDanceType)

DANCE SCHOOL(<u>DSCode</u>, Name, City, ArtisticDirector)

BALLET(<u>BCode</u>, <u>DSCode</u>, Title, #Scene, Type, ChoreographerName)

DANCER-PARTICIPATES-IN-BALLET(<u>SSN</u>, <u>BCode</u>, <u>DSCode</u>, Role)

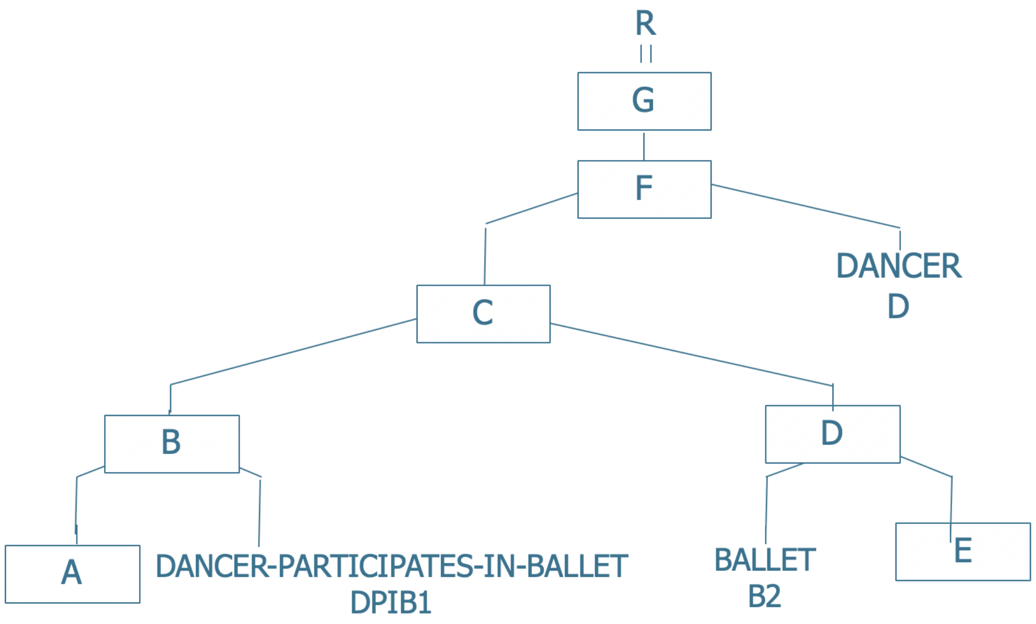
Write the following query in relational algebra:

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. Theta join: B1.DSCode= DP1B2.DSCode AND B1.BCode = DPIB1.BCode

C. Theta join: B1.Type = B2.Type AND B1.BCode != B2.BCode AND DPIB1.SSN = DPIB2.SSN

D. Theta join: B2.DSCode= DP1B2.DSCode AND B1.BCode = DPIB1.BCode

E. Dancer-partecipants-in-ballet DPIB2

F. Theta join: DP1B1.SSN = D.SSN

G. projection D.Name, D.CityofResidence

Domanda 6

Completo

Punteggio max.: 1,00

Given the following relational schema

E-BOOK (<u>ISSN</u>, AuthorId, Title, Language, Editor)

AUTHOR (<u>AuthorId</u>, Name, Surname, Nationality)

DOWNLOAD (<u>Date</u>, <u>Hour</u>, <u>Minutes</u>, <u>UserId</u>, ISSN)

USER (<u>UserId</u>, 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 A.Nationality= 'Italien' AND E.AuthorId = A.AuthorId
GROUPING BY A.AuthorId, A.Name, A.Surname, E.Editor
HAVING COUNT (*)>= 5
```

Domanda 7

Completo

Punteggio max.: 1,00

Given the following relational schema

E-BOOK (<u>ISSN</u>, AuthorId, Title, Language, Editor)

AUTHOR (<u>AuthorId</u>, Name, Surname, Nationality)

DOWNLOAD (<u>Date</u>, <u>Hour</u>, <u>Minutes</u>, <u>UserId</u>, ISSN)

USER (<u>UserId</u>, 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 Editor, COUNT (DISTINCT ISSN), COUNT(*)
FROM E-Book E, DOWNLOAD D
WHERE Editor NOT IN (BLOCK A) AND E.ISSN = D.ISSN
GROUP BY Editor
Block A
SELECT Editor
FROM Author A, E-Book E1
WHERE A.AuthorID = E1.AuthorID AND Nationality = 'German'
```

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 (*)

FROM User U, Download D, E-Book E

WHERE U.UserID = D.UserId AND E.ISSN = D.ISSN AND U.UserId IN (BLOCK A)

BLOCK A

SELECT D1.UserID

FROM Donwload D1, E-Book E1

WHERE D1.ISSN = E1.ISSN AND E1.Language = 'English'

GROUP BY D1.UserId

HAVING COUNT (DISTINCT E1.ISSN)>= 3

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

Internal Identifier: SSN

Attributes: Name, Surname, DateofBirth, education, e-mail, phone*

Generalization (t,e)

Parent entity: Activity

Internal Identifier: ID

Children entity: Musical activity

Attribute: name, level, equipment(1,N)

Children entity: body building

Children entity: cardio-fitness

Binary-Relationship: qualified

TRAINER(1,N) ACTIVITY(1,N)

Ternary-Relationship: takes

TRAINER(1,N) TIME(1,1) MUSICAL-ACTIVITY (1,N)

Attribute: EndTIme, room

Entity: Time

Internal Identifier: Date, StartTime

Foreign Key: SSN (Trainer)

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: GMember

Internal Identifier: Mid

Attributes: Name, Surname, DateOfBirth, Category, ValidityTimePeriod

Binary-Relationship: want to

GMember(1,N) Access(1,1)

Entity: Access

Internal Id: Date

Foreign Key: Mid (GMember)

Attribute: entrance, leaving

GENERALIZATION (t,e)

Parent Entity: Package

Internal Identifier: pid

Attribute: briefDescription, price, time_slot (1,N)

Children entity: Subscription

Attribute: packageduration

Children entity: individual entrance

Attribute: typeofentrance

Children entity: special mix

Attribute: packageDuration

Ternary-relationship: Purchase

Gmember (1,N) Package (1,N)

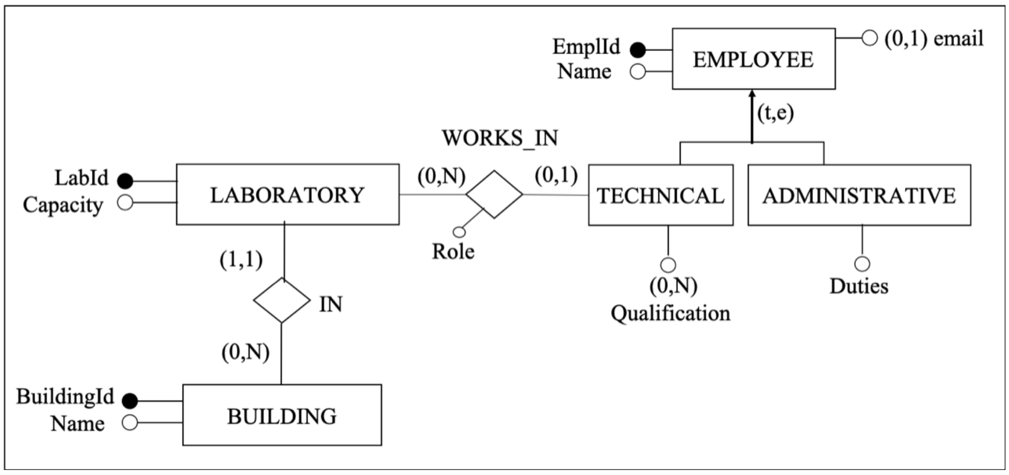
Attribute: Starting, ending

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_ ,Time)

Laboratory (LabId_, BuildingId_, Capacity)

Employee(EmpId_, name, email*)

Techincial(EmpIdm_ Labid, Role)

HAS (Emplid_, Qid_)

Admnistrative(Empid_, Duties)

Specialization(Qid_)

Domanda **12**

Completo

Punteggio max.: 1,00

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

SUPERMARKET(<u>SupermarketCode</u>, City, NumberOfPlacesInQueue, OpeningTime, ClosingTime)

ENTRANCE_QUEUE(<u>SupermarketCode</u>, <u>CustomerSSN</u>)

ENTRANCE_REQUEST(<u>RequestCode</u>, 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 OR REPLACE TRIGGER CHECK_AV
AFTER INSERT ON ENTRANCE-REQUEST
FOR EACH ROW
DECLARE
N Number;
NMax Number;
BEGIN
    SELECT COUNT(*) INTO N
    FROM ENTRANCE-QUEUE EQ
    WHERE SupermarketCode = :New.SupermarketCode

    SELECT NumberofPlacesInQueue AS NMAX
    FROM Supermarket S
    WHERE SupermarketCode =: New.SupermarketCode

    If( N = NMAX) then
        raise_application_error()
        else
                INSERT INTO ENTRANCE_QUEUE()
                        VALUES (:New.SupermartCode, :New.CustomerSSN);
        end if;
end;
```

Domanda **13**

Completo

Punteggio max.: 1,00

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

SUPERMARKET(<u>SupermarketCode</u>, City, NumberOfPlacesInQueue, OpeningTime, ClosingTime)
ENTRANCE_QUEUE(<u>SupermarketCode</u>, <u>CustomerSSN</u>)
ENTRANCE_REQUEST(<u>RequestCode</u>, 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. AFTER INSERT ON UPDATE ON Supermarket
  FOR EACH ROW

B. SELECT COUNT(*) INTO X
   FROM Supermarket
   WHERE (ClosingTIme- OpenTime) >18