anubhavjindal Pushing existing repo Latest commit 79d0526 on 18 Dec 2019 History

1 contributor

241 KB Download

CSCI 585: Database Systems

Spring 2016 - Midterm Exam

3/4/16, 6:00-7:00 PM

Name: _____

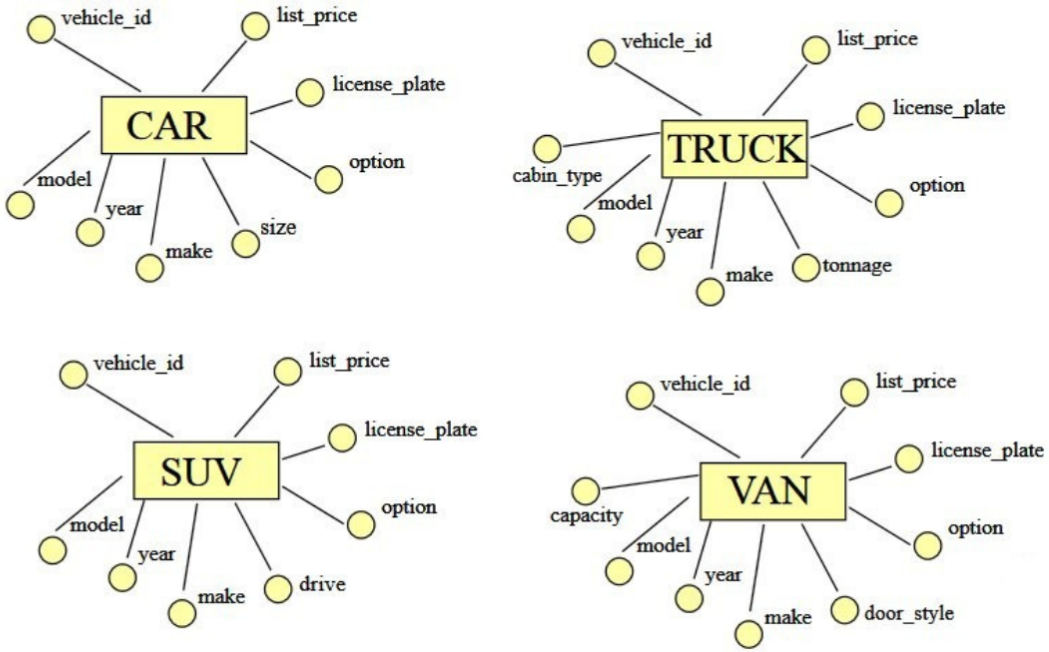
Student ID: _____

Question	Your score	Max score
1		3
2		6
3		4
4		3
5		4
6		2
7		5+1
8		3
Bonus		1
Total		32

HAVE FUN!!

Question 1 [3 points]

Consider the four entities shown below:



Draw a single E-R diagram that depicts the info contained, as a subclass/superclass hierarchy - use whatever notation you are familiar with. Be sure to indicate a suitable primary key.

The expected answer is this: a 'vehicle' (or transportation device etc) superclass, with all the common properties (including vehicle_id, list_price etc), with CAR, TRUCK, SUV, VAN being subclasses (each with its specific attributes, eg. CAR would have 'size'). vehicle_id is the PK.

Question 2 [6 points]

The following table is in 1NF:

```
CREATE TABLE Classes
(course CHAR(7) NOT NULL,
 section CHAR(1) NOT NULL,
 time INTEGER NOT NULL,
```

```
time INTEGER NOT NULL,
room INTEGER NOT NULL,
roomsize INTEGER NOT NULL,
professor CHAR(25) NOT NULL,
student CHAR(25) NOT NULL,
major CHAR(10) NOT NULL,
grade CHAR(1) NOT NULL);
```

Knowing the student and course is enough to determine section and grade. Also, a student can have just a single major. Knowing these facts, convert the above table to 2NF (where there are no partial dependencies); be sure to include primary keys.

2NF:

```
CREATE TABLE Classes (course, section, room, roomsize, time, professor, PRI

CREATE TABLE Enrollment (student, course, section, grade, PRIMARY KEY(stude

CREATE TABLE Students (student, major), PRIMARY KEY(student));
```

Further, 'roomsize' needs to depend (just) on 'room'. Knowing this, create a 3NF version (where there are no transitive dependencies); again, include suitable PKs.

3NF:

```
CREATE TABLE Classes (course, section, room, time, professor, PRIMARY KEY(c

CREATE TABLE Rooms (room, roomsize, PRIMARY KEY(room));

CREATE TABLE Enrollment (student, course, section, grade, PRIMARY KEY(stude

CREATE TABLE Students (student, major), PRIMARY KEY(student));
```

```
CREATE TABLE Students (student, major), PRIMARY KEY(student));
```

Question 3 [4 points]

Here is a SQL query:

```
SELECT sno, sname
FROM Suppliers
WHERE 100 > (SELECT SUM(quantity)
             FROM Shipments
             WHERE Shipments.sno = Suppliers.sno);
```

In the above, sno stands for 'supplier number', and sname, for 'supplier name'. Describe using a sentence or two, what the query does.

Lists suppliers who have shipped less than 100 shipments.

Here is another query that has to do with product managers (a product manager is responsible for selling a

Here is another query that has to do with product managers (a product manager is responsible for selling a product - product managers get products out of warehouses and into stores, where they sell them). The Personnel table lists product managers and their products, while the Warehouses and Stores tables list products and their quantities. What does the query produce?

```
SELECT manager, product
FROM Personnel as P1
WHERE (SELECT SUM(qty)
       FROM Warehouses AS W1
       WHERE P1.product = W1.Product)
    < (SELECT SUM(qty)
       FROM Stores AS S1
       WHERE P1.product = S1.Product);
```

Lists product managers who have more product in stores than in warehouses.

Question 4 [3 points]

Imagine there are two items X and Y in a database, and two transactions T1 and T2 that operate (read, write) on them.

T1 reads X and Y, and modifies X: T1:R(X), T1:R(Y), T1:W(X), c1 [c1 is 'commit 1', ie. T1's commit].

T2 reads X and Y, and modifies both X and Y: T2:R(X), T2:R(Y), T2:W(X), T2:W(Y), c2.

When T1 and T2 interleave their operations shown above, the following problems can happen (unless we carefully avoid them, eg. using two-phase locking):

- write-write conflict, aka "lost update": two transactions overwrite an object - the second (latter) transaction's update makes the first transaction's update to become 'lost'
- write-read conflict, aka "dirty read": one transaction reads a value, after it has written over by another transaction which not yet committed
- read-write conflict, aka "unrepeatable read": one transaction reads the value of an object twice, and another transaction overwrites that value in-between the two reads of the first transaction

Examine the following transaction histories, and indicate which of the if any of the above three problems could occur, and if so, indicate which problem:

- T2:R(X), T2:R(Y), T2:W(X), T1:R(X) ... : write-read conflict

- T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X), T2:R(X) ... : read-write conflict
- T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X), T2:W(X) ... : write-write conflict

Question 5 [4 points]

Rewrite the following queries (eg. for optimization purposes):

```
SELECT product_id, product_name
FROM product
WHERE unit_price BETWEEN MAX(unit_price) and MIN(unit_price)
```

Answer:

```
SELECT product_id, product_name
FROM product
WHERE unit_price >= MAX(unit_price)
and unit_price <= MIN(unit_price)

<=MAX and >=MIN is also acceptable, because of the way I stated the query
```

```
SELECT *
FROM product p
WHERE product_id IN
(SELECT product_id
 FROM order_items)
```

Answer:

```
SELECT *
from product p
where EXISTS (SELECT * from order_items o
              where o.product_id = p.product_id)
```

Question 6 [2 points]

How would you categorize the following transaction scheme (T is a transaction, X and Y are distributed objects)?

- T starts execution
- T reads X at initiating site
- T writes Y at initiating site
- updated value of Y is sent to all nodes that have duplicates of Y, along with a timestamp of when the update occured
- if the other nodes can all update their Y values without conflict, the transaction goes through, ie is committed; otherwise the transaction is discarded (and restarted)

Optimistic locking (or optimistic concurrency control).

Question 7 [5+1 points]

USC has been around since 1880. There have been hundreds of thousands of undergrads who have been graduating since then, and we have pretty detailed info on their academic performance (via transcripts). Imagine we would like to do multidimensional data analysis, specifically on the cumulative GPAs of all the (undergrad) students who have graduated out of 'SC. The 'fact table' might consist of entries that include GPA, student name, major, etc.

We need to build a data warehouse to perform the analysis, and choose to use ROLAP to do so.
For such a scenario, draw a star schema that depicts the fact table along with several dimension tables. Indicate relevant attributes inside each table. In other words, what factors can we consider, to mine this rich data?

The fact table would consist of entries that include:

Cumulative_GPA Gender School_Within_University Major Minor City State Country Year_Graduated

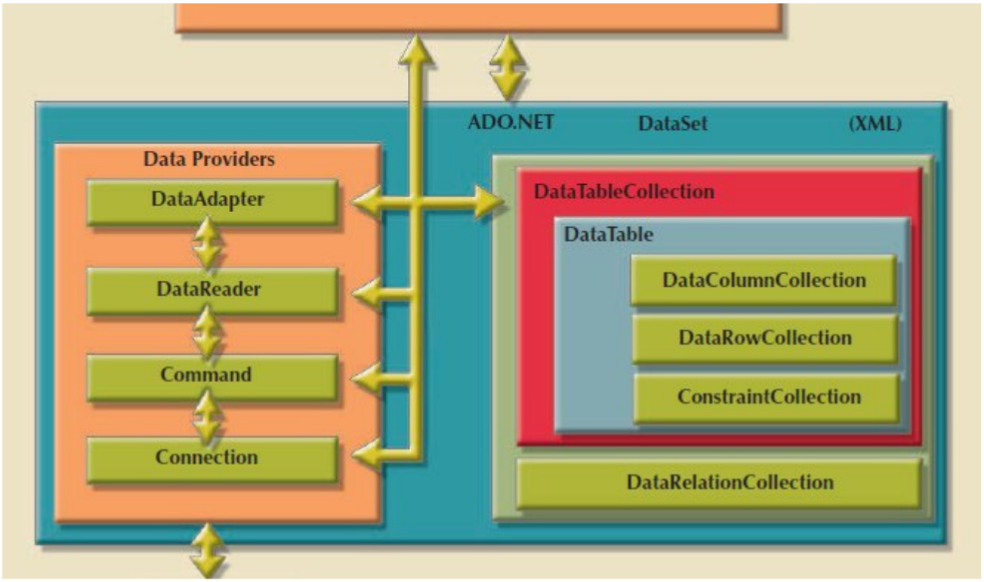
The corresponding dimension tables would be School, Major, Minor, Gender, Location, Year, Age.

Bonus (1 point): create a 'snowflake' schema instead.

A dimension table such as Location could further be split into City, State, Country; Major could be subdivided into Social_Science, Physical_Science, Engineering, etc.

Question 8 [3 points]

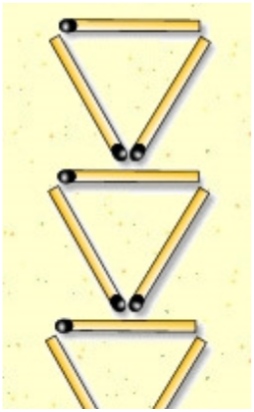
Shown below is the centerpiece of Microsoft's ADO.NET architecture. What does it enable (what capability does it provide)?



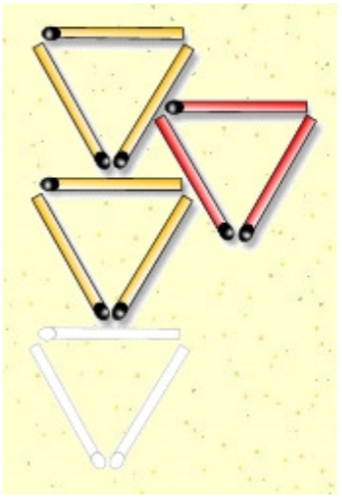
ADO.NET is used to create an XML-based, in-memory view of an underlying database.

Bonus (non-DB-related) [1 point]

Create four equilateral triangles by moving just three matches - no overlapping or breaking allowed.



Answer:



Other solutions are OK too - moving just two matches, or creating a tetrahedron.