**Total points: 102**

**Question 1**. Below is one sample document from the collection tours in MongoDB. Other documents in the tours collection have the same structure. (15 points)

--------------------------------------------------------------------------

{

"\_id": {

"$oid": "572bb8222b288919b68abf5a"

},

"tripduration": 379,

"start station id": 476,

"start station name": "E 31 St & 3 Ave",

"end station id": 498,

"end station name": "Broadway & W 32 St",

"bikeid": 17827,

"usertype": "Subscriber",

"birth year": 1969,

"gender": 1,

"start station location": {

"type": "Point",

"coordinates": [

-73.97966069,

40.74394314

]

},

"end station location": {

"type": "Point",

"coordinates": [

-73.98808416,

40.74854862

]

},

"start time": {

"$date": {

"$numberLong": "1451606445000"

}

},

"stop time": {

"$date": {

"$numberLong": "1451606824000"

}

}

}

--------------------------------------------------------------------------

Write the following queries using MongoDB shell:

1. List the tripduration, birth year and gender (exclude \_id) of documents of which the birth year (Int32) is after 1980 (> 1980). (3 points)

db.tours.find({"birth year": {$gt: 1980}, {tripduration: 1, "birth year": 1, \_id: 0})

1. List all the tour documents that have overlapped with station ids from 399 to 501 (both 399 and 501 are included). Let a/b denote the start/end station id. Notice that a tour could have a b or a > b. Overlap means . (6 points)

db.tours.find({"start station id": {$lte 399}, "end station id": {$gte: 501}})

1. (3.1) Create an index on tripduration and birth year, first by tripduration in descending order and then by birth year in ascending order. (3 points) (3.2) Dispaly all documents using the index created in (3.1). (3 points)

3.1) db.tours.ensureIndex({tripduration: -1, "birth year": 1})

3.2) db.tours.find()

**Question 2**. From what we have learned so far, list all methods or ways to replace joins or that are equivalent to joins in RDBMS, and briefly explain. (5 points)

In MongoDB: Embeddred structure or link via DBRef

In object-relational database system, use reference type

In Spark, use MapReduce

**Question 3**. (12 points)

1. Is C in CAP the same as C in ACID? Why? (For definition of C in ACID, refer to <https://en.wikipedia.org/wiki/ACID>) (4 points)

The two consistency is different. In CAP theorm, consistency means that all nodes see the same data at any time, or reads return lates written data. It ensures all clients get the same result. As in ACID, consistency means the inserted data must be consistent defined schema and constraints.

1. Can A in ACID avoids ‘read uncommited’? Why? (4 points)

Yes. The A (Atomicity) in ACID means each transaction is treated as a single unit. It is either fully commited or never happens. From user level, the data a user read is either unchanged or fully commited.

1. If ‘read uncommited’ is allowed, is it a CP or an AP system? Why? (4 points)

AP. It means the write/read is fast, while different nodes may see different data.

**Question 4**. (10 points) We have three documents as follows:

Document 1 (d1): Structural joins: A primitive for efficient XML query pattern matching

Document 2 (d2): Holistic twig joins: optimal XML pattern matching

Document 3 (d3): Keyword proximity search in XML trees

After preprocessing, we have:

d1: structural join primitive efficient xml query pattern matching

d2: holistic twig join optimal xml pattern matching

d3: keyword proximity search xml tree

1. Calculate the tf and idf for each term. (Use the definitions of tf(di, t) and idf(t) in Lecture 8, where t is a term in di.) (5 points)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Terms | idf(t) | tf(d1,t) | tf(d2,t) | tf(d3,t) |
| structural | 1 | 0.051 | 0 | 0 |
| join | 0.5 | 0.051 | 0.058 | 0 |
| primitive | 1 | 0.051 | 0 | 0 |
| efficient | 1 | 0.051 | 0 | 0 |
| xml | 0.333 | 0.051 | 0.058 | 0.079 |
| query | 1 | 0.051 | 0 | 0 |
| pattern | 0.5 | 0.051 | 0.058 | 0 |
| matching | 0.5 | 0.051 | 0.058 | 0 |
| holistic | 1 | 0 | 0.058 | 0 |
| twig | 1 | 0 | 0.058 | 0 |
| optimal | 1 | 0 | 0.058 | 0 |
| keyword | 1 | 0 | 0 | 0.079 |
| proximity | 1 | 0 | 0 | 0.079 |
| search | 1 | 0 | 0 | 0.079 |
| tree | 1 | 0 | 0 | 0.079 |

1. (5 points) Given a keyword query Q = {xml, join, query}, calculate the relevance of documents to Q as follows:

\* The number of citations of each document are:

c\_d1: 1303

c\_d2: 1442

c\_d3: 267

Let c denote the sum of number of citations of d1, d2 and d3.

\* Assign a weight to each document as w\_i = c\_di/c

\* Relevance of a document di to Q is defined as:

r(di, Q) = w\_i ∗∑t∈Q tf(di, t) ∗idf(t)

Xml: 0.333\*(0.051+0.058+0.079) = 0.063

Join : 0.5\*(0.051+0.058) = 0.055

Query: 0.051

r(d1, Q) = (0.051+0.055+0.063) \* (1303/3012) = 0.073

r(d2, Q) = (0.051+0.055+0.063) \* (1442/3012) = 0.081

r(d3, Q) = (0.051+0.055+0.063) \* (267/3012) = 0.015

**Question 5**. (12 points) Consider relation *R* with 4 attributes VXYZ. For FDs = {VX → Y, VX → Z, Y → V, Z → X}, assuming these are the only dependencies given for R, do the following:

1. Identify the candidate key(s) for R. (4 points)

VX

1. Identify if R satisfies BCNF and why. (4 points)

No. Y -> VY is not a trivial FD and Y is not a superkey.

1. If R is not in BCNF, decompose it into a set of BCNF relations that *preserve the dependencies*. If dependency preserving BCNF decomposition is impossible, explain why. (4 points)

It is impossible to preserver both VX -> Y and VX -> Z.

**Directions for Questions 6 - 15.** (36 points)

For questions 6-14, choose **one** answer from (a), (b), (c), and (d). Each question is 4 points. You will get -1 points for each incorrect answer, and 0 points for no answer.

* Relations may be empty or have NULL's.
* Notice that it is possible to have duplicate tuples in SQL, but relations are sets in relational algebra unless stated otherwise.

**Question 6**. Consider relation Apple whose schema is Apple(a,b,c), i.e., Apple has three attributes a, b and c.

Q1:

  SELECT b FROM Apple;

Q2:

 (SELECT b FROM Apple)

     INTERSECT

 (SELECT b FROM Apple);

INTERSECT is an set operation. Which of the following is correct and explain why:

1. The results of Q1 and Q2 are the same.
2. The result of Q1 is always contained in that of Q2.
3. The result of Q2 is always contained in that of Q1.
4. The results of Q1 and Q2 are different.

Answer is c. Intersect will remove all duplicates.

For the next 3 questions, consider XML documents conforming to the following DTD about students taking courses:

<!DOCTYPE Courses [

<!ELEMENT Courses (Course\*)>

<!ELEMENT Course (Subject, Students)>

<!ATTLIST Course CID ID #REQUIRED Credits PCDATA #REQUIRED>

<!ELEMENT Subject (#PCDATA)>

<!ELEMENT Students (Student+)>

<!ELEMENT Student EMPTY>

<!ATTLIST Student Name PCDATA #REQUIRED> ]>

**Question 7**.

Q1: XPath:

//Course[@CID="1234"]//Student

Q2: XQuery:

for $c in /Courses/Course

where $c/@CID="1234"

return $c/Students/Student

Which of the following is correct (Ignore the order of items in the returned result):

1. The results of Q1 and Q2 are the same.
2. The result of Q1 is always contained in that of Q2.
3. The result of Q2 is always contained in that of Q1.
4. The results of Q1 and Q2 are different.

Answer is a.

**Question 8.**

Q1: XPath:

/Courses/Course

        [Students/Student/@Name != Students/Student/@Name]/Subject

Q2: XQuery:

for $c in /Courses/Course

for $s1 in $c/Students/Student

for $s2 in $c/Students/Student

where $s1/@Name != $s2/@Name

return $c/Subject

Which of the following is correct (Ignore the order of items in the returned result) and explain why:

1. The results of Q1 and Q2 are the same.
2. The result of Q1 is always contained in that of Q2.
3. The result of Q2 is always contained in that of Q1.
4. The results of Q1 and Q2 are different.

**Question 9.**

Q1: XPath:

/Courses/Course[Students/Student/@Name="Smith"]/Subject

Q2: XQuery:

for $c in /Courses/Course

where every $s in $c/Students/Student satisfies $s/@Name="Smith"

return $c/Subject

Which of the following is correct (Ignore the order of items in the returned result):

1. The results of Q1 and Q2 are the same.
2. The result of Q1 is always contained in that of Q2.
3. The result of Q2 is always contained in that of Q1.
4. The results of Q1 and Q2 are different.

**Question 10**. Consider relations Appleand Banana whose schemas are Apple(a,b) and Grape(g,h).

Q1:

SELECT a

FROM Apple

WHERE Apple.b > ALL(SELECT h FROM Grape);

Q2:

SELECT a

FROM Apple

WHERE Apple.b > ANY(SELECT h FROM Grape);

Which of the following is correct and explain why:

1. The results of Q1 and Q2 are the same.
2. The result of Q1 is always contained in that of Q2.
3. The result of Q2 is always contained in that of Q1.
4. The results of Q1 and Q2 are different.

Answer is d.

For Q1, it selected apple whose b is larger than biggest h in grape.

For Q2, it selects apply whose b is larger than smallest h in grape.

**Question 11**. Consider relations Apple and Grape, each having attributes a and b.

Q1:

Q2:

Which of the following is correct and explain why:

1. The results of Q1 and Q2 are the same.
2. The result of Q1 is always contained in that of Q2.
3. The result of Q2 is always contained in that of Q1.
4. The results of Q1 and Q2 are different.

Answer is a.

They both project the intersect of apple and grape.

**Question 12**. Consider relation Apple whose schema is Apple(a,b).

Q1:

SELECT a1.a, a3.a

FROM Apple a1, Apple a2, Apple a3

WHERE a1.b = a2.b AND a2.a = a3.a AND a2.b <> a3.b;

Q2:

SELECT a1.a, a3.a

FROM Apple a1, Apple a2, Apple a3

WHERE a1.a = a2.a AND a2.b = a3.b AND a1.b <> a2.b;

Which of the following is correct:

1. The results of Q1 and Q2 are the same.
2. The result of Q1 is always contained in that of Q2.
3. The result of Q2 is always contained in that of Q1.
4. The results of Q1 and Q2 are different.

Answer is d

**Question 13**. Consider the relation scores and the query given below:

|  |  |  |
| --- | --- | --- |
| **student** | **DSA5014\_score** | **seminar\_score** |
| A | 75 | NULL |
| B | NULL | 91 |
| C | 100 | 85 |

SELECT student  
FROM scores  
WHERE (DSA5104\_score > seminar\_score AND seminar\_score > 75 AND DSA5104\_score > 90) OR (DSA5104\_score < 80)

Which students are returned?  
  
a) B and C only.  
b) A and C only.

c) A only.  
d) A, B, and C.

Answer is b

For the next two questions, consider the following two relations:

*staffs*(id, name, dept, salary)

*managers*(dept, mID)

The staffs relation stores the ID, name, department (dept), and salary of an staff. The manager relation stores the manager ID (mID) of each department (dept).

**Question 14**. To constrain the relations so that in the mID attribute of a tuple in *managers*, there must appear the ID of a staff in *staffs*. Which of the following changes by itself enforces this constraint?

1. In the declaration of *managers*, add for attribute mID the attribute-based check CHECK(EXISTS(SELECT \* FROM staffs WHERE id = mID)).
2. In the declaration of staffs, add the constraint FOREIGN KEY id REFERENCES managers(mID).
3. In the declaration of managers, add the constraint FOREIGN KEY mID REFERENCES staffs(id).
4. More than one of the above.

Answer is d

**Question 15**. To constrain the relations so that the total salary of all staffs in one department could not exceed $1,000,000. Below is a SQL query framework to create an assertion to enforce this constraint:

CREATE ASSERTION total\_salary CHECK(

NOT EXISTS(Q));

Which query Q best enforces this constraint?

a)

SELECT \* FROM staffs WHERE SUM(salary) > 1000000

b)

SELECT dept, SUM(salary) FROM staffs

GROUP BY dept

c)

SELECT SUM(salary)

FROM staffs, managers

WHERE id = mID

GROUP BY staffs.dept

HAVING SUM(salary) > 1000000

d)

SELECT dept FROM staffs

GROUP BY dept HAVING SUM(salary) > 1000000

Answer is d

**Question 16**. Consider relation R(A,B,C,D,E) with functional dependencies FDs = {AB → C, BC → D, CD → E, DE → A, AE → B}. (4 points)

What is the total number of superkeys (including keys) of R? List them below or briefly explain.

Total number of superkeys of R is \_\_21\_\_\_\_\_.

Why:

AB, BC, CD, DE, AE are candidate keys. So any key with three or more atrributes are superkeys.

So is 5 + (10C5) + (10C4) + (10C3) = 21

**Question 17**. Consider the following instance of relation R(X,Y,Z,H) (4 points)

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **H** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

R does not satisfy the functional dependency X → Y. What is the minimum number of tuples to delete from R (with no other insertion or update) such that X → Y holds? Give an sample instance of R below after removing the minimum number of tuples to make X → Y hold.

The minimum number is \_\_\_\_4\_\_\_\_\_.

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **H** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |