**CS585**

Database Systems

Spring 2010

Exam II

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Maximum | Received |
| Problem 1 | 24 |  |
| Problem 2 | 12 |  |
| Problem 3 | 12 |  |
| Problem 4 | 15 |  |
| Problem 5 | 7 |  |
| Problem 6 | 20 |  |
| Problem 7 | 10 |  |

**DURATION:** 2 Hours

Please look over the entire exam first

and start with easy questions.   
Good luck!

**Problem 1: (24 points)**

Consider the following xml files storing the account numbers and transactions between accounts:

Accounts.xml:

<?xml version="1.0"?>

<Accounts>

<Account >

<OwnerName>Shahriar Shamsian </OwnerName>

<AccountNumber>123-456-789</AccountNumber>

</Account>

…

</Accounts>

Transactions.xml:

<?xml version="1.0"?>

<Transactions>

<Transaction id =0>

<PayerAccount>234-456-789</PayerAccount>

<Amount Unit=”$”>80</Amount>

<PayeeAccount>123-456-789</ PayeeAccount>

</Transaction>

…

</Transactions>

Write the corresponding XQueries for the following queries assuming the given XML files.

**a)** Display the owner names and the account numbers which have received more than $50 from other accounts in average. The query output should be in the following format. (8 points)

<Qa>

<Account>

<OwnerName> Shahriar Shamsian </OwnerName>

< AccountNumber >123-456-789</ AccountNumber >

<AvgMoneyReceived>56</AvgMoneyReceived>

</Account>

…

</Qa>

<Qa>

{

for $ac in document("accounts.xml")//Account

let $t := document("transactions.xml")//transaction[PayeeAccount = $ac/AccountNumber]

where avg($t/Amount) > "50"

return

<Account>

{$ac/OwnerName}

{$ac/AccountNumber}

< AvgMoneyReceived >

{avg($t/ Amount)}

</ AvgMoneyReceived >

</Account>

}

</Qa>

**b)** Display all the people who sent money to “Shahriar Shamsian”. Sort the output alphabetically. The output should be in the following format. (8 points)

<Qb>

<Name> Shahin Shayandeh </Name>

<Name> Krati Ahuja</Name>

<Name> Ronak Sankhla</Name>

</Qb>

<Qb>

{

let $p2 := document("accounts.xml")//Account[OwnerName = " Shahriar Shamsian"]

let $t := document("transactions.xml")//Transaction[PayeeAccount = $p2/AccountNumber]

for $p1 in distinct-values(document("accounts.xml")//Account[AccountNumber =

$t/PayerAccount])

return

<Name>

{$p1/OwnerName/text()}

</Name>

sortby (Name ascending)

}

</Qb>

**c)** Display all the account numbers owned by each person sorted by the account number in descending order. The output should be in the following format. (8 points)

<Qc>

<Owner>

<Name>Shahriar Shamsian</Name>

<Number>123-456-789</Number>

<Number>213-456-689</Number>

</Owner>

…

</Qc>

<Qc>

{

for $o in distinct-values(document("accounts.xml")//Account/OwnerName)

let $p := document("accounts.xml")//Account[OwnerName = $o]

return

<Owner>

<Name>

{$o/text()}

</Name>

{$p/AccountNumber}

sortby (AccountNumber descending)

</Owner>

}

</Qc>

**Problem 2 (12 points)**

Consider the following 2-dimensional cube. Each cell defines salary for the corresponding state and age. Rows represent ages, columns represent states.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **CA** | **NY** | **DC** | **OR** | **AZ** | **TX** | | **20** | 20 | 20 | 40 | 20 | 30 | 30 | | **25** | 50 | 40 | 50 | 40 | 40 | 50 | | **30** | 50 | 50 | 40 | 50 | 60 | 50 | | **35** | 70 | 50 | 70 | 70 | 50 | 60 | | **40** | 60 | 60 | 80 | 80 | 70 | 70 | | **45** | 70 | 80 | 90 | 90 | 70 | 80 | |

1. Specify what the measure attributes and dimension attributes are. (2 points)

“salary” is measure attribute while “age” and “state” are dimension attributes

1. Draw the corresponding Prefix-Sum cube. (4 points)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 20 | 40 | 80 | 100 | 130 | 160 |
| 70 | 130 | 220 | 280 | 350 | 430 |
| 120 | 230 | 360 | 470 | 600 | 730 |
| 190 | 350 | 550 | 730 | 910 | 1100 |
| 250 | 470 | 750 | 1010 | 1260 | 1520 |
| 320 | 620 | 990 | 1340 | 1660 | 2000 |

**c)** Use the Prefix-Sum cube to answer the following query. (4 points)

(Note: Identify the Prefix-Sum cells that you use to answer the query.)

*“What is the average salary in states AZ and TX and for ages between 30 and 40?”*

*(AZ<=state<=TX and 30<=age<=40)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 20 | 40 | 80 | 100 | 130 | 160 |
| 70 | 130 | 220 | 280 | 350 | 430 |
| 120 | 230 | 360 | 470 | 600 | 730 |
| 190 | 350 | 550 | 730 | 910 | 1100 |
| 250 | 470 | 750 | 1010 | 1260 | 1520 |
| 320 | 620 | 990 | 1340 | 1660 | 2000 |

Sum= 1520-1010-430+280=360

AVG=360/6=60

**d)** How much does the above query cost? (2 points)

(Note: Count the number of I/O operations required to answer the query, assuming each cell is stored in a separate disk block.)

4

**Problem 3 (12 points)**

DotCom, an online retailer, carries three categories of products: books, music, and video. DotCom has three warehouses, one in New Jersey (NJ), one in California (CA), and one in Colorado (CO).

* The CA warehouse carries books, music and videos, and ships orders for the US-West and US-Central regions.
* The NJ warehouse carries music and videos, and ships orders for the US-East and US-Central regions.
* The CO warehouse carries books and music, and ships orders anywhere in the US.

The DotCom database has three tables:

* PROD(PID, CAT, DESC, ...)
* INV(PID, WID, UNITS, ...)
* ORD(OID, PID, SHIP, ...)

Each table has some additional fields we are not interested in. Some additional details:

* PID is the primary key for the products relation PROD.
* (PID, WID) is a key for the inventory relation INV.
* (OID, PID) is a key for the orders relation ORD.
* CAT is one of “book”, “music”, and “video”.
* WID is one of “CA”, “CO”, and “NJ”.
* SHIP is one of “US-West”, “US-Central”, and “US-East.”

The CA warehouse issues queries of the form:

*SELECT \* FROM PROD, ORD, INV*

*WHERE PROD.PID = ORD.PID AND*

*PROD.PID = INV.PID AND*

*WID = "CA" AND*

*(SHIP = "US-West" OR SHIP = "US-Central")*

The CO warehouse issues queries of the form:

*SELECT \* FROM PROD, ORD, INV*

*WHERE PROD.PID = ORD.PID AND*

*PROD.PID = INV.PID AND*

*WID = "CO" AND*

*(CAT= "book" OR CAT= "music")*

The NJ warehouse issues queries of the form:

*SELECT \* FROM PROD, ORD, INV*

*WHERE PROD.PID = ORD.PID AND*

*PROD.PID = INV.PID AND*

*WID = "NJ" AND*

*(CAT="music" OR CAT="video") AND*

*(SHIP = "US-East" OR SHIP = "US-Central")*

**a)** What are the sets of relevant simple predicates to (horizontally) fragment each of the tables PROD, ORD, and INV? (3 points)

* Predicates for PROD:

PROD: CAT = “book” , CAT = “music” , CAT = “video”

* Predicates for ORD:

ORD: SHIP = “US-West” , SHIP = “US-Central” , SHIP = “US-East”

* Predicates for INV:

INV: WID = “CA” , WID = “CO” , WID = “NJ”

**b)** What is the corresponding primary fragment for each predicate derived in part (a) above? (3 points)

* Fragments for PROD:

PROD: CAT=“book” (PROD), CAT=“music” (PROD), CAT=“video” (PROD)

* Fragments for ORD:

ORD: SHIP=“US-West” (ORD), SHIP=“US-Central” (ORD), SHIP=“US-East” (ORD)

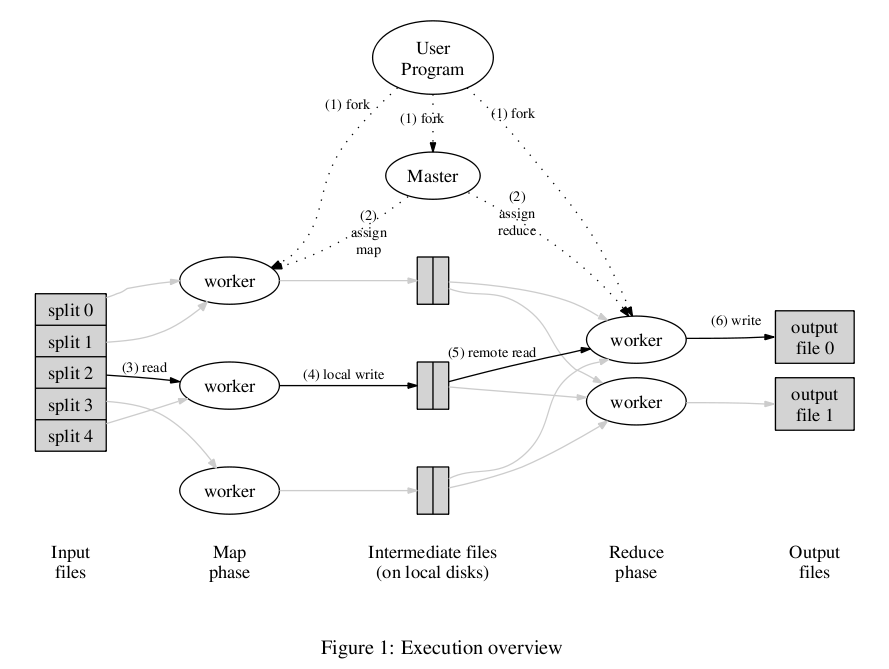
* Fragments for INV:

INV: WID=“CA” (INV), WID=“CO” (INV), WID=“NJ” (INV)

1. Give an optimal assignment of the above PROD and INV fragments to the three warehouse locations. You can replicate fragments and assume that for all fragments, their update frequency is less than their read frequency. ( 6 points)

**Problem 4 (15 points)**

1. Draw and briefly describe the execution overview of MapReduce. (7 points)



1. Implement map and reduce functions for the following problem: “Count of URL Access Frequency: Given multiple logs of URL accesses, return number of accesses to each URL.” (8 points)

map(string key, string value)‏

//key:

//value:

reduce(string key, iterator values)‏

//key:

//values:

map(string key, string value)‏

//key: log name

//value: document contents

for each URL url in value

EmitIntermediate(url, “1”);

reduce(string key, iterator values)‏

//key: url

//values: list of counts

int results = 0;

for each v in values

result += ParseInt(v);

Emit(AsString(result));

**Problem 5 (7 points)**

Give a brief description to each item

a) Advantage of JDBC over embedded SQL (2 points)  
  
Main advantage is that the application becomes DB vendor independent. The same executable can run on top of different databases.

b) When and why stored procedures are used (2 points)

When you have a sequence of queries where the input of one is directly or indirectly related to the output of another. A stored procedure can improve performance of this sequence of queries by eliminating the intermediate communication costs between the queries.

c) Advantage of hybrid query optimization as opposed to static or dynamic query optimization

(3 points)

Static query optimization is cheap by can become suboptimal after a while if relation sizes change

Dynamic query optimization can always be optimal but the cost is high for each query

Hybrid query optimization, combines the positive aspects of these two approaches by computing optimal query evaluation methods and storing them for static optimization, but re-computing the evaluation methods whenever the relation sizes change beyond a certain threshold, so the results are always close to optimal.

**Problem 6 (20 points)**

Consider the following database at GrocerySuppliesDotCom:

* Supplier (Supplier#, Sname, Scity, Sstate)
* Item (Item#, Supplier#, Price, category, ....)
* PurchaseOrder (PO#, Item#, Warehouse#, PO\_date, ....)
* Warehouse (Warehouse#, Wstate, ....)

Each table has some additional fields that we are not interested in.

**a)** Write the following query in SQL. (3 points)

List the item#, Price, and warehouse# of purchase orders issued after Jan-1-2010 for items purchased from suppliers in Chicago that cost more than $100 per item and end up at warehouses in California.

Select I.Item#, I.Price, W.warehouse#

From Supplier S, Item I, PurchaseOrder P, Warehouse W

Where I.Item# = P.Item#

AND I.Supplier# = S.Supplier#

AND P.Warehouse# = W.Warehouse#

AND P.PO\_date > “01-JAN-2010”

AND I.price > 100

AND S.Scity = “Chicago”

AND W.Wstate = “CA”

**b)** Draw a **suboptimal** relational algebra query tree for the above query. (3 points)

**σB.price>50**

**Π**

**σW.Wstate = ‘CA’**

**P**

**I**

**⋈P.Item#=I.Item#**

**S**

**W**

**⋈S.Supplier# = I.Supplier#**

**⋈P.Warehouse# =W.Warehouse#**

**σP.PO\_DATE>2010**

**σS.Scity = ‘Chicago’**

**c)** Use the heuristics for algebraic query optimization to transform/restructure the query-tree you generated in part (b) above into a more efficient query-tree. (4 points)

**Π**

**⋈I.Item#=P.Item#**

**⋈S.Supplier# =I.Supplier#**

**⋈P.Warehouse# = W.Warehouse#**

**σI.price>100**

**σS.Scity** = “Chicago”

**σP.PO\_DATE >** 2010

**σW.Wstate =**“CA”

**S**

**P**

**W**

**I**

**d)** Given the following database catalog information and the fragmentation of relations across sites, show how this query can be optimally executed in a distributed environment. Present your solution by generating a list of operations that must be performed in sequence to execute the query. Each operation is either a local query performed at a particular site or a data transfer between sites. In your list, number the operations in chronological order, and use the same number for the operations that can happen in parallel. (10 points)

Catalog information:

* Cardinality of Supplier is 300
* Cardinality of Item is 200,000
* Cardinality of PurchaseOrder is 10 million
* Cardinality of Warehouse is 156
* Orders are saved from Jan-01-1985 up to now
* Item prices range from $10 to $110.
* 2% of suppliers are based in Chicago
* Warehouses exist in all 52 states
* Assume uniform distributions

Fragmentation:

* Site 1: Suppliers with Supplier# ≤ S200
* Site 2: Suppliers with Supplier# > S200
* Site 3: Items corresponding to suppliers with Supplier# ≤ S200
* Site 4: Items corresponding to suppliers with Supplier# > S200
* Site 5: Warehouses with warehouse# ≤ W70
* Site 6: Warehouses with warehouse# > W70
* Site 7: Purchase orders corresponding to warehouses with warehouse# ≤ W70
* Site 8: Purchase orders corresponding to warehouses with warehouse# > W70

1. Select Suppliers with Scity = ‘chicago’ in site 1, Select Items with price > 100 in site 3 🡪 join in site 1 or 3 (consider 1)
2. Select Suppliers with Scity = ‘chicago’ in site 2, Select Items with price > 100 in site 4 🡪 join in site 2 or 4 (consider 2)
3. Select Warehouse with WState = ‘CA’ in site 5, Select PurchaseOrders with PO\_date > “2010” in site 7 🡪 join in site 5 or 7 (consider 5)
4. Select Warehouse with WState = ‘CA’ in site 6, Select PurchaseOrders with PO\_date > “2010” in site 8 🡪 join in site 6 or 8 (consider 6)
5. send data from site 6 to site 5
6. send data from site 2 to site 1
7. send all data from site 5 to site 1
8. join over Item# in site 1 and project the results

**Problem 7 (10 points)**

For each statement below mark whether it is true (T) or false (F)

F\_\_\_\_\_\_\_\_\_\_For a 3 dimensional OLAP cube, it is possible to respond to any range query by

doing at most 8 lookups in the prefix sum matrix.

T\_\_\_\_\_\_\_\_\_\_A data warehouse may keep data at a higher level of aggregation (larger buckets)

from that stored at the source of the data

F\_\_\_\_\_\_\_\_\_\_A data warehouse may keep data at a lower level of aggregation (smaller buckets)

from that stored at the source of the data

T\_\_\_\_\_\_\_\_\_\_The number one reason why a DDBMS performs better than a centralized DBMS in

a distribute environment is that in a DDBMS data can be made local to nodes that

use it most

F\_\_\_\_\_\_\_\_\_\_In vertical fragmentation, a given field can only be part of one fragment

T\_\_\_\_\_\_\_\_\_\_In horizontal fragmentation, a given row can only be part of one fragment