DSCI-551 Midterm2

Chaoyu Li, 100/100

QUESTION 1 [HFDS]

Consider the data pipelining process for writing a block in HDFS.

(1.a.) What is the default block size?

A: 128MB.

(1.b.) Suppose packet size is 64KB. How many packets will a block be split into?

A: Packets Number = 128MB / 64KB = 2048

(1.c) How many acknowledge messages does the client need to receive? Show your derivation.

A: The client needs to receive 2050 acknowledge messages. One acknowledge message for each packet, so there will be 2048 acknowledge messages. One acknowledge message for set up and one for close. Therefore, the total number of acknowledge messages will be 2050.

(1.d.) What control messages are needed for the process?

A: 1. A control message for pipeline setup. It should be sent to all DataNode that will be written before the first packet.

2. A control message for closing the pipeline. It should be sent after the last packet. NameNode will update the metadata for the file when receiving the closing message.

(1.e.) Is the process synchronous or asynchronous? Explain your answer.

A: The process is asynchronous because it will greatly increase efficiency. After the client sends a packet, it can continue to send the next packet without waiting for the response from the DataNode.

QUESTION 2 [XML & XPath]

Consider the person.xml file. Part of its content is shown below.

```
<persons>
  <person id=”123”>
    <name>John Smith</name>
    <age>25</age>
    <address><city>LA</city><state>CA</state></address>
  </person>
  <person id=”124”>
    <name>David Smith</name>
    <phone>312-123-4567</phone>
  </person>
  …
```

</persons>

Write an XPath expression for each of the following questions. 5 points each question.

(2.a.) Find names of people who are at least 25 years old. Output name only (not the element)

A: /persons/person[age>=25]/name/text()

(2.b.) Find phone numbers of people whose phone number contains “123”. Output phone number only.

A: /persons/person[contains(phone,'123')]/phone/text()

(2.c.) Find names of people who live in LA. Output name only.

A: /persons/person[address[city='LA']]/name/text()

(2.d.) Find address (element) of person whose id is 123.

A: /persons/person[@id='123']/address

QUESTION 3 [ER and relational data models]

Consider the tables in a beers database as shown below. Note that it has one more table Ales than you have seen in class.

Beers(name, manf)

Ales(name, color)

Drinkers(name, addr, phone)

Bars(name, addr)

Likes(drinker, beer)

Frequents(drinker, bar)

Sells(bar, beer, price)

(3.a.) Reverse engineer the above tables into an ER diagram. State the entities and relationships in the diagram. Remember to state the keys for the entities.

A: #The keys will be underlined.

Entities:

Beers(<u>name</u>, manf)

Ales(<u>name</u>, color)

Drinkers(<u>name</u>, addr, phone)

Bars(<u>name</u>, addr)

Relationships:

Likes(<u>drinker</u>, beer): many-many relationships

Frequents(drinker, bar): many-many relationships
Sells(bar, beer, price): one-many relationships
is-a(subclass): between Beers and Ales.

(3.b.) Explain if it is a good idea to merge the table Frequents with either Drinkers or Bars table.
A: It is not a good idea to merge Frequents with neither Drinkers nor Bars.
Because Frequents here is a many-many relationship. If we merge it with either Drinkers or Bars, it would cause redundancy of addr and phone attributes, or manf attribute.

QUESTION 4 [SQL]

Consider the same tables as in Question 3:

Beers(name, manf)

Ales(name, color)

Drinkers(name, addr, phone)

Bars(name, addr)

Likes(drinker, beer)

Frequents(drinker, bar)

Sells(bar, beer, price)

Write an SQL query for each of the following questions. 6 points each question.

(4.a.) Find colors of Ales beers NOT made by Heineken. Output the same colors once.
A: SELECT DISTINCT A.color
FROM Ales A, Beers B
WHERE A.name=B.name AND B.manf!='Heineken';
(4.b.) Find out, for each manufacturer, how many of its beers are sold in bars with a price of at least 3 dollars.
A: SELECT Beers.manf,count(*)
FROM Beers, Sells
WHERE Beers.name=Sells.beer and Sells.price>=3
GROUP BY Beers.manf;
(4.c.) Find manufacturers of beers that nobody likes.
A: SELECT DISTINCT B.manf
FROM Beers B
WHERE not exists (
SELECT L.beer FROM Likes L WHERE L.beer=B.name);
(4.d.) Find out drinkers who frequents at least two different bars.

A: SELECT DISTINCT F1.drinker
FROM Frequents F1, Frequents F2
WHERE F1.drinker=F2.drinker AND F1.bar!=F2.bar;
(4.e.) Find out bars which sell beers made by Heineken.
A: SELECT S.bar
FROM Sells S, Beers B
WHERE S.beer=B.name AND B.manf='Heineken';

QUESTION 5 [Constraints]
Consider the same tables as in Question 3:
Beers(name, manf)
Ales(name, color)
Drinkers(name, addr, phone)
Bars(name, addr)
Likes(drinker, beer)
Frequents(drinker, bar)
Sells(bar, beer, price)

(5.a.) State the foreign keys in these tables. For each foreign key, indicate which attribute in which table it refers to.
A: 1. Ales.name is a foreign key refers to Beers.name.
2. Likes.drinker is a foreign key refers to Drinkers.name.
3. Likes.beer is a foreign key refers to Beers.name.
4. Frequents.drinker is a foreign key refers to Drinkers.name.
5. Frequents.bar is a foreign key refers to Bars.name.
6. Sells.bar is a foreign key refers to Bars.name.
7. Sells.beer is a foreign key refers to Beers.name.
(5.b.) Write a SQL command to create a new table called CanDrink that records which drinker can drink which beer. Suppose both attributes have the type: varchar(20). Make sure you properly define the primary key and foreign key in the table.
A: Create table CanDrink(
drinker varchar(20) NOT NULL,
beer varchar(20),
PRIMARY KEY(drinker, beer),
FOREIGN KEY(drinker) REFERENCES Drinkers(name) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(beer) REFERENCES Beers(name) ON DELETE CASCADE ON UPDATE CASCADE
);

(5.c.) State when (i.e., insert/delete/update of which table) the database should check possible violations of foreign key(s) in the CanDrink table.

A: Insert into CanDrink should check Drinkers and Beers table.

Delete Drinkers/Beers should check drinker/beer in the CanDrink table.

Update Drinkers/Beers should check drinker/beer in the CanDrink table.