Name

CWID

Quiz
1

Feburary 19th, 2014
Due Feburary 26th, 11:59pm

Quiz 1: CS525 - Advanced Database Organization

Please leave this empty! 1.1 1.2 1.3 1.4 Sum

Instructions

- **You have to hand in the assignment using your bitbucket account**
- **This is an individual and not a group assignment**
- Multiple choice questions are graded in the following way: You get points for correct answers and points subtracted for wrong answers. The minimum points for each questions is **0**. For example, assume there is a multiple choice question with 6 answers - each may be correct or incorrect - and each answer gives 1 point. If you answer 3 questions correct and 3 incorrect you get 0 points. If you answer 4 questions correct and 2 incorrect you get 2 points. ...
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are 4 parts in this quiz
  1. SQL
  2. Relational Algebra
  3. Index Structures
  4. Result Size Estimation

Part 1.1 SQL (Total: 31 + 10 bonus points Points)

Consider the following transportation database schema and example instance. The example data should not be used to formulate queries. SQL statements that you write should return the correct result for every instance of the schema!

city

name	state
New York	NY
Chicago	IL
Schaumburg	IL
Seattle	WA

bus

company	number	price	fromCity	toCity
Whitedog	13	210	New York	Chicago
Whitedog	102	56	Schaumburg	Chicago
Picobus	2	115	Seattle	Chicago

schedule

company	bnum	departureTime	arrivalTime
Whitedog	13	2014-01-12 08:13	2014-01-12 19:56
Whitedog	102	2014-01-13 12:15	2014-01-13 15:13
Picobus	2	2014-01-12 10:30	2014-01-13 05:44
Picobus	2	2014-01-13 10:30	2014-01-14 05:44

Hints:

- Attributes with black background are the primary key attributes of a relation
- The attributes *fromCity* and *toCity* of relation *bus* are both foreign keys to attribute *name* of relation *city*.
- The attributes *company* and *bnum* of relation *schedule* form a foreign key to attributes *company* and *number* of relation *bus*.

Question 1.1.1 (2 Points)

Write a query that returns cities for which both incoming and outgoing connections exists (*toCity* and *fromCity* attributes of relation bus). Make sure that each such city is only returned once by the query.

Question 1.1.2 (3 Points)

Write an SQL query that returns the bus number, destination, and arrival time of 'Whitedog' busses departing from 'Chicago' on Jan 1st 2014. You can assume that the data type of the *arrivalTime* and *departureTime* columns are of type `VARCHAR`.

Question 1.1.3 (2 Points)

What is the result of evaluating the following SQL query over the example data? Write down the content of the relation that will be returned.

```
SELECT DISTINCT company
FROM schedule s
WHERE (SELECT count(*) FROM schedule o WHERE s.company = o.company) = 2;
```

Question 1.1.4 (4 Points)

Write an SQL query that returns the cheapest price for round trips between Chicago and Washington (a bus ride from Chicago to Washington plus a bus ride from Washington to Chicago). For this query you can ignore the actual scheduled times of the busses.

Question 1.1.5 (3 Points)

Write an SQL query that returns companies that operate at least 5 bus lines.

Question 1.1.6 (3 Points)

Write an SQL query that returns a departure time table for busses. This table should have three attributes: the city, the bus number, and the departure time. The table should be ordered by city, then by bus number, and finally by departure time.

Question 1.1.7 (4 Points)

Write an SQL query that returns the name of cities which are neither the destination nor start point (fromCity or toCity) of any bus route.

Question 1.1.8 (4 Points)

Write a query that returns the number of intra-state busses (bus lines with both fromCity and toCity within the same state) per state.

Question 1.1.9 (6 Points)

Write an SQL query that returns the price of the cheapest route from New York to Houston using at most 3 bus lines. For example, a trip New York - Chicago, Chicago - Seattle, Seattle - Houston uses 3 bus lines. Do not consider actual scheduling times for determining routes for this query. That is you are allowed to return routes that would not work in practice because, e.g., one of the buses arrives after the next one has departed.

Hint: This is a relatively complex query. Recall that you can use `WITH` in SQL to define temporary views.

Question 1.1.10 Optional Bonus Question (10 bonus points Points)

Improve the previous query by taking the bus schedule into account. Now you should only return bus trips which actually work. E.g., if you are using two bus lines, then the second line should departe after the first one arrives. Furthermore, trips that are longer than 18 hours should not be considered. You can assume that the DBMS implements a '-' operator for the departureTime and arrivalTime columns that returns the difference between the two dates in hours.

Part 1.2 Relational Algebra (Total: 29 Points)

Question 1.2.1 Relational Algebra (3 Points)

Write a relational algebra expression over the schema from the SQL part (part 1) that returns bus numbers for all busses from Picobus and Whitedog (bag semantics).

Question 1.2.2 Relational Algebra (4 Points)

Write a relational algebra expression over the schema from the SQL part (part 1) that returns all companies that do not have busses leaving from Chicago (bag semantics).

Question 1.2.3 Relational Algebra (4 Points)

Write a relational algebra expression over the schema from the SQL part (part 1) that returns the number of all buses with more than 3 scheduled times (schedule table). (bag semantics)

Question 1.2.4 SQL $\rightarrow$ Relational Algebra (3 Points)

Translate the SQL query from Question 1.1.1 into relational algebra (bag semantics).

Question 1.2.5 SQL $\rightarrow$ Relational Algebra (5 Points)

Translate the SQL query from question 1.1.3 into relational algebra (bag semantics).

Question 1.2.6 SQL $\rightarrow$ Relational Algebra (5 Points)

Translate the SQL query from question 1.1.4 into relational algebra (bag semantics).

Question 1.2.7 Equivalences (5 Points)

Consider the following relation schemas:

$R(A, B)$, $S(B, C)$, $T(C, D)$.

Check equivalences that are correct under **set semantics**. For example $R \bowtie R \equiv R$ should be checked, whereas $R \equiv S$ should not be checked.

- ☐ $R \cup (S - R) \equiv S \cup R$
- ☐ $R \cap S \equiv R - (R - S)$
- ☐ $R \ltimes S \equiv \pi_{A,B}(R \bowtie S)$
- ☐ $\pi_A(R \mathbin{⟕} S) \equiv \pi_A(S \mathbin{⟖} R)$
- ☐ $\sigma_{B=5}(R \rhd S) \equiv R \rhd \sigma_{B=5}(S)$
- ☐ $(R \cap S) \cup (R \cap T) \equiv R \cap (S \cup T)$
- ☐ $\sigma_{A=5}(R \mathbin{⟗}_{B=C} T) \equiv \sigma_{A=5}(R) \mathbin{⟗}_{B=C} T$
- ☐ $\sigma_{C=5}(R \mathbin{⟗}_{B=C} T) \equiv \sigma_{B=5}(R) \mathbin{⟗}_{B=C} T$
- ☐ $R \ltimes S \equiv R - (R \rhd S)$
- ☐ $(R \cup S) - T \equiv R \cup (S - T)$

Part 1.3 Index Structures (Total: 30 Points)

Assume that you have the following table:

Item

id	name	price
15	Shovel	13
44	Spate	23
3	Lawnmover	233
47	Lawnmover XL	499
48	Fertilizer	45
60	Sunflower seeds	3
32	Pine tree	299
23	Hop seeds	14

Question 1.3.1 Construction (12 Points)

Create a B+-tree for table *Item* on key *id* with $n = 2$ (up to two keys per node). You should start with an empty B+-tree and insert the keys in the order shown in the table above. Write down the resulting B+-tree after each step.

When splitting or merging nodes follow these conventions:

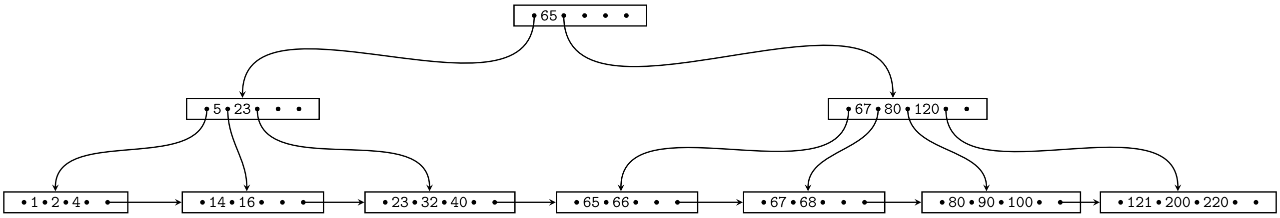
- **Leaf Split**: In case a leaf node needs to be split during insertion and n is even, the left node should get the extra key. E.g, if $n = 2$ and we insert a key 4 into a node [1,5], then the resulting nodes should be [1,4] and [5]. For odd values of n we can always evenly split the keys between the two nodes. In both cases the value inserted into the parent is the smallest value of the right node.
- **Non-Leaf Split**: In case a non-leaf node needs to be split and n is odd, we cannot split the node evenly (one of the new nodes will have one more key). In this case the "middle" value inserted into the parent should be taken from the right node. E.g., if $n = 3$ and we have to split a non-leaf node [1,3,4,5], the resulting nodes would be [1,3] and [5]. The value inserted into the parent would be 4.
- **Node Underflow**: In case of a node underflow you should first try to redistribute values from a sibling and only if this fails merge the node with one of its siblings. Both approaches should prefer the left sibling. E.g., if we can borrow values from both the left and right sibling, you should borrow from the left one.

Question 1.3.2 Operations (10 Points)

Given is the B+-tree shown below ($n = 4$). Execute the following operations and write down the resulting B+-tree after each operation:

delete(68), delete(80), insert(3), delete(200), delete(66), delete(100)

Use the conventions for splitting and merging introduced in the previous question.



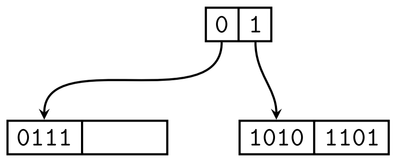
Question 1.3.3 Extensible Hashing (8 Points)

Consider the extensible Hash index shown below that is the result of inserting values 3, 4, and 5. Each page holds two keys. Execute the following operations

`insert(0),insert(7),insert(6),insert(1),delete(5)`

and write down the resulting index after each operation. Assume the hash function is defined as:

x	h(x)
0	1100
1	0001
2	0000
3	1010
4	1101
5	0111
6	1110
7	0000
8	1010



Part 1.4 Result Size Estimations (Total: 10 Points)

Consider a table *beer* with attributes *brand*, *name*, *type*, *alc*, a table *brewery* with *brand*, *city*, *revenue*, and a table *loc* with attributes *city* and *state*. *beer.brand* is a foreign key to brewery *brand*. Attribute *city* of relation brewery is a foreign key to attribute *city* of relation loc. Given are the following statistics:

$T(beer) = 10,000$	$T(brewery) = 400$	$T(loc) = 2,000$
$V(beer, brand) = 300$	$V(brewery, brand) = 400$	$V(loc, city) = 2,000$
$V(beer, name) = 8,000$	$V(brewery, city) = 50$	$V(loc, state) = 50$
$V(beer, type) = 10$	$V(brewery, revenue) = 200$	
$V(beer, alc) = 10,000$		

Question 1.4.1 Estimate Result Size (3 Points)

Estimate the number of result tuples for the query $q = \sigma_{type=Wheat}(beer)$ using the first assumption presented in class (values used in queries are uniformly distributed within the active domain).

Question 1.4.2 Estimate Result Size (3 Points)

Estimate the number of result tuples for the query $q = \sigma_{revenue>20,000,000}(brewery)$ using the first assumption presented in class. The minimum and maximum values of attribute *revenue* are $300,000$ and $4,500,000,000$.

Question 1.4.3 Estimate Result Size (4 Points)

Estimate the number of result tuples for the query $q = \mathit{beer} \bowtie \mathit{brewery} \bowtie \mathit{city}$ using the first assumption presented in class.