

Name

CWID

Quiz 1

Feb 08, 2023
Due Feb 20, 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 diderot
- This is an individual and not a group assignment
- Do not rename the .sql files
- The SQL part is autograded, each question gives full points if it returns the expected query answers and 0 points otherwise.
- 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 publication 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 possible instance of the schema!

Bar

name	city	owner	closedon
Murphys	Chicago	McDonald	NULL
The Elephant	Chicago	Smith	Saturday
Fiesta	Schaumburg	Gordot	Sunday

Drink

drink	type	alcoholperc
Cranberry Juice	juice	0.0
Millers light	beer	3.5
Lagunitas maximus	beer	9.0
Jack Daniels	liquor	38.0
Oban	liquor	45.0

Menu

drink	bname	bcity	price
Cranberry Juice	Murphys	Chicago	3.50
Millers light	Murphys	Chicago	3.00
Oban	Murphys	Chicago	14.50
Guinness	Murphys	Chicago	7.50
Lagunitas maximus	The Elephant	Chicago	6.00
Jack Daniels	The Elephant	Chicago	9.50
Guinness	The Elephant	Chicago	8.30
Lagunitas maximus	Fiesta	Schaumburg	5.50
Jack Daniels	Fiesta	Schaumburg	12.50
Cranberry Juice	Fiesta	Schaumburg	6.50

Receipt

id	patron	bname	bcity	paymentmethod
1	Smitty	Murphys	Chicago	card
2	Smitty	The Elephant	Chicago	cash
3	Fran	Murphys	Chicago	card
4	Otto	Murphys	Chicago	cash
5	Schmotto	Fiesta	Schaumburg	cash
6	Potto	Fiesta	Schaumburg	cash

Item

receiptid	drink	quantity
1	Millers light	1
1	Oban	3
2	Guinness	2
3	Guinness	4
3	Cranberry Juice	2
4	Oban	1
5	Lagunitas maximus	1
5	Jack Daniels	1
6	Lagunitas maximus	5

Hints:

- Attributes with black background are the primary key attributes of a relation
- Attribute `Drink` of relation `Menu` is a foreign keys to relation `Drink`.
- Attributes `banem`, `bcity` of relation `Menu` are a foreign key to relation `Bar`.
- Attributes `banem`, `bcity` of relation `Receipt` are a foreign key to relation `Bar`.
- Attribute `drink` of relation `item` is a foreign keys to relation `Drink`.

Question 1.1.1 (3 Points)

Write an SQL query that returns names of patrons who have only ordered drinks of type beer. The result schema should be (`patron`).

Question 1.1.2 (5 Points)

Write an SQL query that returns pairs of patrons that have never ordered the same drink. That is, the first patron never ordered any drink that the second patron has ordered and vice versa. The result schema should be (patron_1, patron_2).

Question 1.1.3 (5 Points)

Write a SQL query that returns for each bar the name of the patron who has spend the most on drinks ordered at this bar. Return the bar name, city where the bar is located, and name of the patron. The result schema should be (name,city,patron).

Question 1.1.4 (2 Points)

Write a SQL query that returns for each bar the 2 most expensive items on this bar's menu. Return the bar name, city, item (drink), and the drink's price. The result schema should be (bname, bcity, drink, price).

Question 1.1.5 (2 Points)

Write an SQL query that returns all bars sorted in descending order based on their total income (the total dollar amount spend by patrons at this bar). Return the bar name, city where the bar is located, and the total amount. The result schema should be (**bname**, **bcity**, **ttl**).

Question 1.1.6 (3 Points)

Write an SQL query that computes the average price per beverage type and city and returns for each bar the number of drinks that are more than 20% more expensive than the average price for their beverage type in the city the bar is located in. The result schema should be (bname,bcity,numdrinks).

Question 1.1.7 (5 Points)

Write an SQL query that computes distances between pairs of patrons (return `patron_1`, `patron_2`, and `distance`). The distance between two patrons is defined as the length of the shortest path between the two patrons in a graph that is defined as follows. The nodes of the graph are the patrons. There is an edge between patron p_1 and patron p_2 if p_1 and p_2 have both visited the same bar. Pairs of patrons that are not connected in the graph should not be returned.

Question 1.1.8 (2 Points)

Write an SQL query that returns for each patron how much they have paid in total in cash and using a credit card. The result schema should be (`patron`, `cashamount`, `cardamount`).

Question 1.1.9 (4 Points)

Write an SQL query that returns the number of nodes in the largest clique of the social network graph from question 1.1.7. A clique is a set of nodes such that each pair of nodes from this set is reachable from each other. The result schema should be `cnt` (the number of nodes in the clique).

Question 1.1.10 Optional Bonus Question (10 Bonus Points)

For this question, you will write a SQL **query** implementing a SAT solver. Given a predicate logic formula, a SAT solver determines whether there exists a satisfying assignment for the formula. That is, can we assign the value true or false to each variable in the formula such that the formula evaluates to true. We assume that the formula is given in conjunctive normal form (CNF), i.e., as a conjunction (**AND**) of disjunctions (**OR**) of terms called clauses. A term is either a variable or its negation. The formula is given as input as a table with schema **formula**(**clause**,**var**,**negated**). The formula is encoded in this table as follows. Each clause (disjunction) is assigned an id (start with id 0). Each term of a clause with id i is encoded as one row in the table with **clause** equal to i , **var** stores an identifier of the variable (variable identifiers start at 0 and are consecutive) and **negated** is 0 if the term is not negated and 1 if it is negated. For example, the unsatisfiable formula $(x_0 \vee \neg x_1) \wedge (x_1 \vee x_2)$ would be encoded as

conjunct	var	negated
0	0	0
0	1	1
1	1	0
1	2	0

Your query should take such a table as input and return a single row with a single column. The value of this column should be the boolean value **TRUE** if the formula is satisfiable and **FALSE** otherwise.

Part 1.2 Relational Algebra (Total: 29 Points)

Question 1.2.1 Relational Algebra (4 Points)

Write a relational (bag semantics) algebra expression over the schema from the SQL part (part 1) that returns bars that have at least 3 items on their menu.

Question 1.2.2 Relational Algebra (3 Points)

Write a relational (bag semantics) algebra expression over the schema from the SQL part (part 1) that returns the maximum alcohol percentage per drink `type`.

Question 1.2.3 Relational Algebra (5 Points)

Write a relational (bag semantics) algebra expression over the schema from the SQL part (part 1) that returns bars which have no receipts for patrons.

Question 1.2.4 SQL \rightarrow Relational Algebra (5 Points)

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

Question 1.2.5 SQL \rightarrow Relational Algebra (6 Points)

Translate the SQL query from question 1.1.2 into relational algebra (bag semantics).

Question 1.2.6 SQL \rightarrow Relational Algebra (6 Points)

Translate the SQL query from question 1.1.4 into relational algebra (bag semantics), but only return the most expensive item per bar.

Part 1.3 Index Structures (Total: 30 Points)

Assume that you have the following table:

Item		
SSN	name	age
325-63-232	Joe	68
287-07-858	Pete	66
778-48-806	Lily	71
276-34-872	Bob	39
368-11-536	Heinz	20
600-63-751	John	62
788-27-365	Alice	20
306-36-224	Gertrud	32

Question 1.3.1 Construction (12 Points)

Create a B+-tree for table *Item* on key *age* 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:

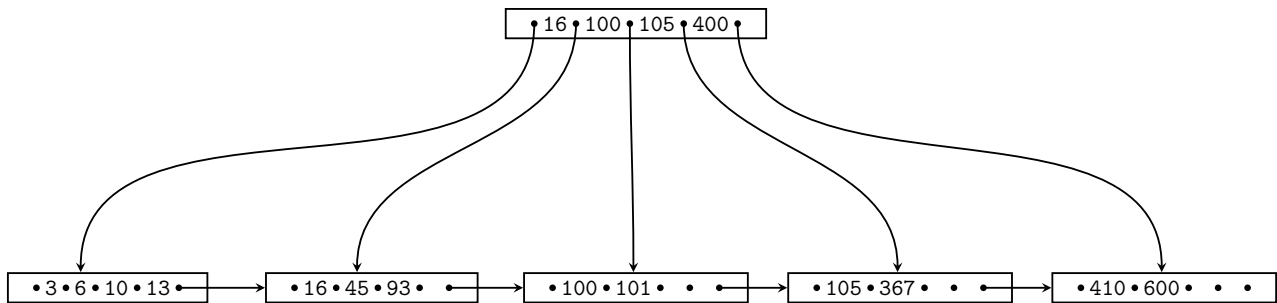
- **Leaf Split:** In case a leaf node needs to be split during insertion and n is even, the left node should get the extra key. E.g, if $n = 2$ and we insert a key 4 into a node $[1,5]$, then the resulting nodes should be $[1,4]$ and $[5]$. For odd values of n we can always evenly split the keys between 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)

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

insert(7), insert(8), insert(9), insert(11), delete(101), delete(105)

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



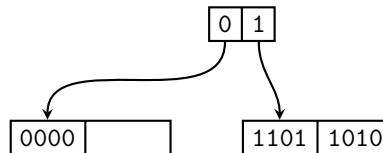
Question 1.3.3 Extensible Hashing (8 Points)

Consider the extensible Hash index shown below that is the result of inserting values 0, 1, and 2. Each page holds two keys. Execute the following operations

`insert(5), insert(8), insert(0), insert(3)`

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

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



Part 1.4 Result Size Estimations (Total: 10 Points)

Consider a table `lecture` with attributes `title`, `campus`, `topic`, `roomSize`, a table `student` with `name`, `major`, `age`, and a table `attendsLecture` with attributes `student`, `lecture`, and `hoursAttended`. `attendsLecture.student` is a foreign key to `student`. Attribute `lecture` of relation `attendsService` is a foreign key to of relation `lecture`. Given are the following statistics:

$$\begin{array}{lll} T(lecture) = 10 & T(student) = 8,000 & T(attendsLecture) = 40,000 \\ V(lecture, title) = 10 & V(student, name) = 7,500 & V(attendsLecture, student) = 8,000 \\ V(lecture, campus) = 2 & V(student, major) = 4 & V(attendsLecture, lecture) = 8 \\ V(lecture, topic) = 3 & V(student, age) = 40 & V(attendsLecture, hoursAttended) = 100 \\ V(lecture, roomSize) = 5 & & \end{array}$$

Question 1.4.1 Estimate Result Size (3 Points)

Estimate the number of result tuples for the query $q = \sigma_{age=24}(student)$ 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 $q = \sigma_{hoursAttended > 10 \wedge hoursAttended \leq 20}(attendsLecture)$ using the first assumption presented in class. The minimum and maximum values of attribute `hoursAttended` are 0 and 100.

Question 1.4.3 Estimate Result Size (4 Points)

Estimate the number of result tuples for the query q below using the first assumption presented in class.

$$q = (attendsLecture \bowtie_{lecture=title} \sigma_{campus=Mies}(lecture)) \bowtie_{name=student} \sigma_{major=CS \vee major=Math}(student)$$