Name

CWID

Exam
2

May 10th, 2013

CS525 - Final Exam
Solutions

Please leave this empty! 1 2 3 4 5 Sum

Instructions

- Things that you are **not** allowed to use
  - Personal notes
  - Textbook
  - Printed lecture notes
- The quiz is **120** minutes long
- 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 5 parts in this quiz
  1. SQL and Relational Algebra
  2. Index Structures
  3. Sorting
  4. Schedules and Concurrency Control
  5. Physical Optimization

Part 1 SQL and Relational Algebra (Total: 40 Points)

Consider the following database schema and instance:

politician

name	party	gender
Obama	Democrats	m
Bush	Republicans	m
Paul	Republicans	m
Clinton	Democrats	f

office

title	salary
President	500.000
Governor	100.000
Mister of Defence	600.000
Secretary of State	30.000

lobbyist

company	industry
BP	gas & oil
Nestle	food
Bushmaster	arms
Boeing	arms

held

politician	office	year
Obama	President	1990
Obama	Governor	2004
Clinton	President	2030
Bush	President	1985
Bush	Secretary of State	2013
Paul	Minister of Defence	2003
Paul	Minister of Defence	2004

sponsored

lobbyist	politician	year	amount
BP	Obama	1990	1,000,000
BP	Obama	2004	30,000,000
Bushmaster	Bush	1985	100,000
Bushmaster	Bush	2013	15,000,000
Boeing	Bush	2013	5,000,000
BP	Bush	1985	350,000
Boeing	Paul	2003	10,000
Boeing	Paul	2004	400,000
Nestle	Clinton	2030	40,000

Hints:

- When writing queries do **only** take the schema into account and **not** the example data given here. That is you queries should return correct results for all potential instances of this schema.
- Underlined attribute form the primary key of an relation.
- The attribute *politician* and *office* of relation *help* are foreign keys to the attributes *name* in relation *politician* and *title* in relation *office* respectively.
- The attribute *lobbyist* and *politician* of relation *sponsored* are foreign key to the attributes *company* in relation *lobbyist* and *name* in relation *politician* respectively.

Question 1.1 (8 Points)

Write an SQL statement that returns the company of all lobbyists that have sponsored at least one democrat politician during a year where this politician became President.

Solution

```
SELECT DISTINCT l.company
FROM politician p, office o, lobbyist l, held h, sponsored s
WHERE p.name = h.politician
      AND o.title = h.office
      AND l.company = s.company
      AND p.name = s.politician
      AND s.year = h.year
      AND o.title = 'President'
      AND p.party = 'Democrats'
```

The access to relations office and lobbyist can actually be avoided here:

```
SELECT DISTINCT s.lobbyist AS company
FROM politician p, held h, sponsored s
WHERE p.name = h.politician
      AND p.name = s.politician
      AND s.year = h.year
      AND h.office = 'President'
      AND p.party = 'Democrats'
```

Question 1.2 (8 Points)

Write an SQL statement that returns the total amount of lobbyist money retrieved per political party and company. For example, an example output that may be produced by this query is `(Democrats, BP, 10,000)` that is the format is `(Party, Company, Sum of money)`. The example was wrong, solutions that group by politician and company are correct too.

Solution

```
SELECT party, lobbyist, sum(amount) AS total
FROM politician p, sponsored s
WHERE p.name = s.politician
GROUP BY party, lobbyist
```

Alternatively, an redundant additional join with relation lobbyist may be used.

Question 1.3 (10 Points)

Write an SQL statements that returns the name of the politician that retrieved the highest total amount of sponsorship.

Solution

```
SELECT name
FROM
     (SELECT p.name, sum(amount) AS total
     FROM politician p JOIN sponsored s ON (p.name = s.politician)
     GROUP BY name) AS sub
WHERE total = (SELECT max(total)
               FROM
                    (SELECT p.name, sum(amount) AS total
                    FROM politician p JOIN sponsored s ON (p.name = s.politician)
                    GROUP BY name) AS sub
               );
```

or using limit and order by

```
SELECT p.name, sum(amount) AS total
FROM politician p JOIN sponsored s ON (p.name = s.politician)
GROUP BY name
ORDER BY sum(amount) DESC
LIMIT 1;
```

Question 1.4 SQL → Relational Algebra (7 Points)

Translate the following SQL query into relational algebra (bag semantics).

```
SELECT sum(salary) AS allSal, party
FROM politician p,
     office o,
     held h,
     (SELECT DISTINCT politician
     FROM sponsored
     WHERE lobbyist = 'BP' OR lobbyist = 'Boeing') s
WHERE p.name = h.politician AND o.title = h.office AND s.politician = p.name
GROUP BY party
```

Solution

$$
\begin{aligned}
q &= \rho_{allSal \leftarrow sum(salary)}(\pi_{sum(salary),party}({}_{sum(salary)}\alpha_{party}(join))) \\
join &= ((\textbf{politician} \bowtie_{name=politician} \textbf{held}) \bowtie_{office=title} \textbf{office}) \bowtie_{name=politician} s \\
s &= \delta(\pi_{politician}(\sigma_{lobbyist='BP' \vee lobbyist='Boeing'}(\textbf{sponsored})))
\end{aligned}
$$

Question 1.5 Equivalences (7 Points)

Consider the following relational schemata:

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

Prove or disprove the equivalence (set semantics) shown below using the operator definition and algebra equivalence discussed in class. In each step of the proof write down which equivalence you are using (e.g., associativity of union or pushing selections).

$$\sigma_{(A=5)\wedge(B>3)}(R \bowtie_{B=C} S) \equiv \sigma_{(A=5)\wedge(B>3)}(R) \bowtie \sigma_{C>3}(S)$$

The join condition is missing in the right-hand side: $B = C$

Solution

Since the equivalence above is missing a join condition this is actually wrong. Thus, everybody that is disproving this is correct too.

$$\sigma_{(A=5)\wedge(B>3)}(R \bowtie_{B=C} S) \equiv \sigma_{A=5}(\sigma_{B>3}(R \bowtie_{B=C} S))$$
(Splitting of conjunctive selections into several selections)

$$\sigma_{A=5}(\sigma_{B>3}(R \bowtie_{B=C} S)) \equiv \sigma_{B>3}(\sigma_{A=5}(R \bowtie_{B=C} S))$$
(Commutativity of selection)

$$\sigma_{B>3}(\sigma_{A=5}(R \bowtie_{B=C} S)) \equiv \sigma_{B>3}(\sigma_{A=5}(R) \bowtie_{B=C} S)$$
(Direct pushdown of selections through joins)

$$\sigma_{B>3}(\sigma_{A=5}(R) \bowtie_{B=C} S) \equiv \sigma_{B>3}(\sigma_{A=5}(R)) \bowtie_{B=C} \sigma_{C>3}(S)$$
(Direct pushdown of selections through joins, and implied selection conditions)

$$\sigma_{B>3}(\sigma_{A=5}(R)) \bowtie_{B=C} \sigma_{C>3}(S) \equiv \sigma_{(A=5)\wedge(B>3)}(R) \bowtie \sigma_{C>3}(S)$$
(Combining adjacent selections using conjunction and commutativity of boolean AND.)

Part 2 Index Structures (Total: 25 Points)

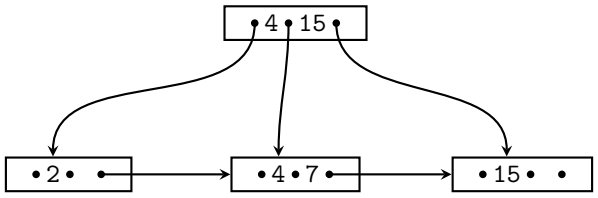
Question 2.1 B+-tree Operations (15 Points)

Given is the B+-tree shown below (**n = 2**). When splitting or merging nodes follow the conventions used in the last assignment:

- **Leaf Split**: In case a leaf node need 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.
- **Leaf Underflow**: In case of a leaf underflow your implementation 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.

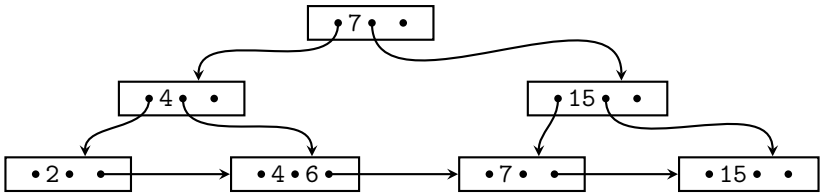
Execute the following operations and **write down the resulting B+-tree after each operation**.

insert(6),insert(9),insert(8),delete(2),delete(6)

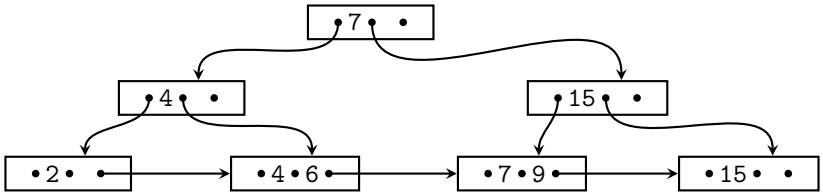


Solution

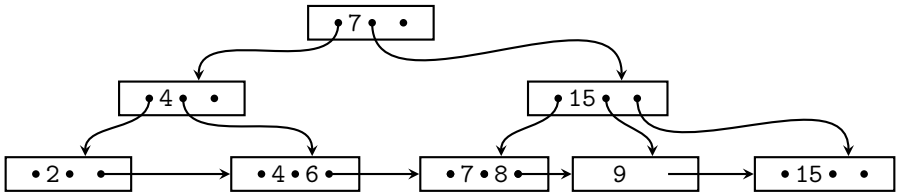
insert(6)



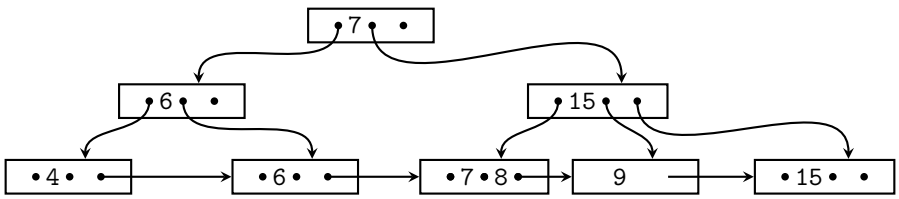
insert(9)



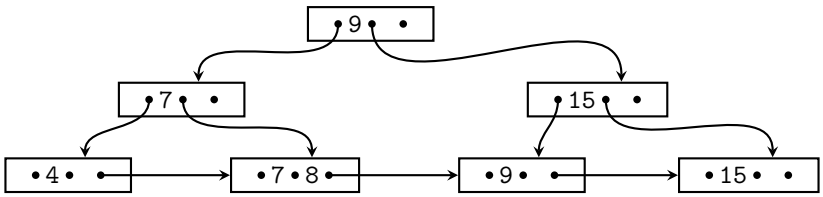
insert(8)



delete(2)

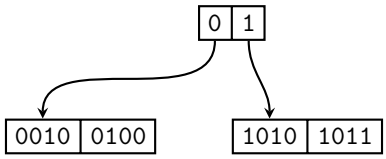


delete(6)



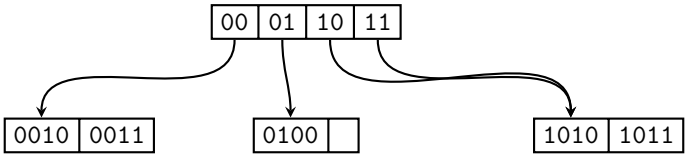
Question 2.2 Extensible Hashing (10 Points)

Consider the extensible Hash index shown below. Each page holds two keys. Execute the following operations `insert(6),insert(2),insert(7),delete(2)` and write down the resulting index after each operation. Assume the hash function assigns values as follows: $h(6) = 0011, h(2) = 0000, h(7) = 1111$. The original version was $h(6) = 0010$. Using this variant is correct too!

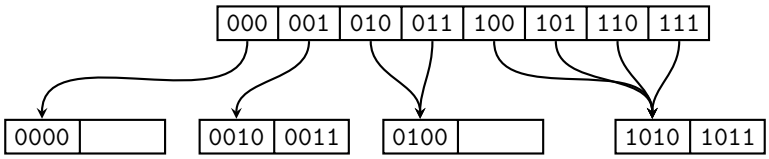


Solution

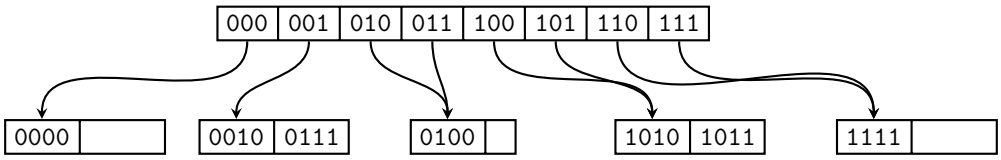
insert(6)



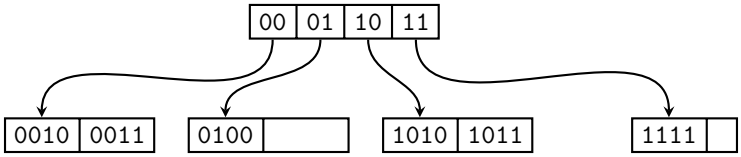
insert(2)



insert(7)



delete(2)



Part 3 Sorting (Total: 10 Points)

Question 3.1 External Sorting (10 = 5 + 5 Points)

Assume we have $M = 101$ memory pages available for sorting and have to sort a relation R with $B(R) = 20,000$ pages and $N(R) = 80,000$ records.

1. Compute the number of I/O operations needed to sort R using external merge sort without using a min-heap during run generation.
2. Assume there is a **clustered B+-tree** index on the sort attribute(s) with $K = 50$ keys in each leaf node and height 4. How many I/Os operations are needed to sort relation R if at the beginning of the sort none of the pages of the B+-tree are in memory.

Solution

1. $2 \cdot B(R) \cdot (1 + \lceil log_{M-1}(B(R)/M) \rceil) = 2 \cdot 20,000 \cdot (1 + 2) = 120,000$ I/Os
2. clustered B+-tree
   - $N(R)/K + HT(I) + B(R) = 80,000/50 + 4 + 20,000 = 21,604$ I/Os

Part 4 Schedules and Concurrency Control (Total: 15 Points)

Question 4.1 Schedule Classes (15 Points)

Indicate which of the following schedules belong to which class. Recall transaction operations are modelled as follows:

$w_1(A)$	transaction 1 wrote item A
$r_1(A)$	transaction 1 read item A
c_1	transaction 1 commits
a_1	transaction 1 aborts

$$S_1 = r_1(A), w_2(A), r_1(B), c_1, w_3(B), r_3(B), c_3, c_2$$

$$S_2 = r_2(B), r_1(B), r_3(A), w_1(B), r_2(C), w_2(C), c_2, w_1(B), c_1, w_3(A), c_3$$

$$S_3 = r_1(A), w_2(B), r_1(B), w_1(B), r_2(A), c_1, w_2(A), c_2$$

$$S_4 = r_2(B), r_1(B), r_2(A), w_2(A), w_1(B), c_2, c_1$$

- ■ S_1 is recoverable
- ■ S_1 is cascade-less
- ■ S_1 is conflict serializable
- ■ S_1 is 2PL
- ❑ S_1 is SS2PL

- ■ S_2 is recoverable
- ■ S_2 is cascade-less
- ■ S_2 is conflict serializable
- ■ S_2 is 2PL
- ❑ S_2 is SS2PL

- ❑ S_3 is recoverable
- ❑ S_3 is cascade-less
- ❑ S_3 is conflict serializable
- ❑ S_3 is 2PL
- ❑ S_3 is SS2PL

- ■ S_4 is recoverable
- ■ S_4 is cascade-less
- ■ S_4 is conflict serializable
- ■ S_4 is 2PL
- ❑ S_4 is SS2PL

Part 5 Physical Optimization (Total: 30 Points)

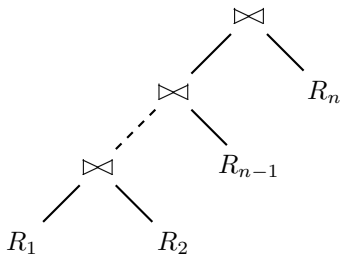
Consider the following relations $R(A, B)$, $S(C, D)$, $T(E, F)$ with $S(R) = S(S) = S(T) = \frac{1}{5}$ (5 tuples fit on each page). The sizes and value distributions are shown below. Recall that $N(R)$ is the number of tuples in R, $B(R)$ is the number of pages of relation R, and $V(R, A)$ is the number of distinct values or relation R in attribute A.

$N(R) = 3000$	$V(R, A) = 3000$	$V(R, B) = 1000$
$N(S) = 1000$	$V(S, C) = 500$	$V(S, D) = 500$
$N(T) = 50$	$V(T, E) = 10$	$V(T, F) = 50$

Question 5.1 Dynamic Programming (30 Points)

Use the dynamic programming join enumeration algorithm to find the cheapest **left-deep plan** for the join $R \bowtie_{B=C} S \bowtie_{D=E} T$. Assume that **block nested-loop** is the only available join implementation with the left input being the "outer" (for each chunk from the outer we have to scan the whole inner relation). Furthermore, there are no indicies defined on any of the relations (that is you have to use **sequential scan** for each of the relations). As a cost model consider the **total number of I/O operations**. For example, if you join two relations with $5,000$ and $10,000$ tuples with $S = \frac{1}{10}$, where the $5,000$ tuple relation is the outer and $M = 101$ memory pages are available, then the cost would be $5,500$. Assume that the system does not support pipelining at all. That is you need to consider the cost of writing the result of a join to disk and reading it from disk when executing the next level join. Note that you do not have to consider any I/O to write the final result. **Ignore plans containing cross-products**. Hint: You will have to estimate the size of intermediate results. Use the estimation based on the number of values and not the one based on the size of the domain. Use the assumption that the number of values in a join attribute of a join result is the minimum of the number of values in the join attribute of each input.

Write down the memoized query plans in `optPlan` after each iteration of the algorithm using the following notation. Write $\{R_1, R_2, \ldots, R_n\} : (R_1, R_2, \ldots, R_n)^{C,S}$ to denote that the plan shown below with I/O cost C and result size S is the bet plan for the set $\{R_1, \ldots, R_n\}$.



Solution

Initialization:

$$\{R\} = (R)^{600,3000} \qquad \{S\} = (S)^{200,1000} \qquad \{T\} = (T)^{10,50}$$

---

i = 2:

Here we have 4 different options how to join two of the plans from the initialization (the other options contain cross-products and, thus, are not considered):

$$(R,S)^{2400,3000}, (S,R)^{2000,3000}, (S,T)^{240,100}, (T,S)^{230,100}$$

As an example take the join (R,S). Here R is the outer and S is the inner. Using the formula from class the estimated result size is $\frac{N(R)\cdot N(S)}{max(V(R,B),V(S,C))} = \frac{3000\cdot 1000}{1000} = 3000$. The cost is computed as: For each chunk of size $M-1$ from R ($\frac{B(R)}{M-1}$) we have to scan S once (200 I/Os) plus we have to read R once (600 I/Os). Thus, the cost is $B(R) + B(R)/100 \cdot B(S) = 600 + 6 \cdot 200 = 1800$ I/Os. In addition we have to account for the cost of writing the results to disk: 600 I/Os. Thus, the total cost is 2400 I/Os.

$$\{R,S\} = (S,R)^{2400,3000} \qquad \{S,T\} = (T,S)^{230,100}$$

---

i = 3:

Now we need to consider all partitions of $\{R,S,T\}$ that do not contain cross-products. These are $(R),(S,T)$ and $(R,S),T$. For each partition S,O we consider all join options between $optPlan(\{S\}$ with $optPlan(\{O\})$. Since we only allow for left-deep plans the only valid option is to join the result of the two element subset with the singleton subset.

$$(T,S,R)^{980,300}, (S,R,T)^{3050,300}$$

The final state of `optPlan` is

$$\{R,S,T\} = (T,S,R)^{980,300}$$