

Name

CWID

Exam 1

March 12th, 2014

CS525 - Midterm Exam Solutions

Please leave this empty!

1	<input type="text"/>	2	<input type="text"/>	3	<input type="text"/>	4	<input type="text"/>	5	<input type="text"/>	Sum	<input type="text"/>
---	----------------------	---	----------------------	---	----------------------	---	----------------------	---	----------------------	-----	----------------------

Instructions

- Things that you are **not** allowed to use
 - Personal notes
 - Textbook
 - Printed lecture notes
 - Phone
- The exam is **90** 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 exam
 1. Disk Organization and Buffering (15)
 2. SQL (24)
 3. Relational Algebra (20)
 4. Index Structures (22)
 5. Operator Implementations (19)

Part 1 Disk Organization and Buffering (Total: 15 Points)

Question 1.1 Page Replacement Clock (15 Points)

Consider a buffer pool with 5 pages using the **Clock** page replacement strategy. Initially the buffer pool is in the state shown below. We use the following notation $^{flag}[page]_{fix}^{dirty}$ to denote the state of each buffer frame. $page$ is the number of the page in the frame, fix is its fix count, $dirty$ is indicating with an Asterix that the page is dirty, and $flag$ is the reference bit used by the Clock algorithm. E.g., $^1[5]_2^*$ denotes that the frame stores page 5 with a fix count 2, that the page is dirty, and that the reference bit is set to 1. Recall that Clock uses a pointer S that points to the current page frame (the one to be checked for replacement next). The page frame S is pointing to is shown in bold. In your solution draw an array to the page frame that S is pointing to.

Current Buffer State

$$^1[11]_0 \ ^0[2]_1 \ ^1[\mathbf{5}]_1 \ ^1[7]_0^* \ ^0[10]_1$$

Execute the following requests and write down state of the buffer pool after each request.

- p stands for pin
- u for unpin
- d for marking a page as dirty

$$u(5), u(10), p(1), p(7), p(20)$$

Solution

u(5)

$$^1[11]_0 \ ^0[2]_1 \ ^1[\mathbf{5}]_0 \ ^1[7]_0^* \ ^0[10]_1$$

u(10)

$$^1[11]_0 \ ^0[2]_1 \ ^1[\mathbf{5}]_0 \ ^1[7]_0^* \ ^0[10]_0$$

p(1)

$$^1[\mathbf{11}]_0 \ ^0[2]_1 \ ^0[5]_0 \ ^0[7]_0^* \ ^1[1]_1$$

p(7)

$$^1[\mathbf{11}]_0 \ ^0[2]_1 \ ^0[5]_0 \ ^1[7]_1^* \ ^1[1]_1$$

p(20)

$$^0[11]_0 \ ^0[2]_1 \ ^1[20]_1 \ ^1[\mathbf{7}]_1^* \ ^1[1]_1$$

Part 2 SQL (Total: 24 Points)

Consider the following database schema and instance:

country

cName	size	population	continent
USA	900,000	320,000,000	America
Canada	1,200,000	60,000,000	America
Germany	300,000	80,000,000	Europe

party

pName	ctry	leaning
Republicans	USA	conservative
Democrats	USA	centrist
CDU	Germany	conservative
SPD	Germany	left-centrist

leader

party	ctry	year	lName
CDU	Germany	1996	Helmut Kohl
CDU	Germany	1998	Helmut Kohl
Republicans	USA	1860	Lincoln

Hints:

- When writing queries do only take the schema into account and **not** the example data given here. That is your queries should return correct results for all potential instances of this schema.
- Attributes with black background form the primary key of an relation. For example, **cName** is the primary key of relation **country**.
- The attribute **ctry** of relation **party** is a foreign key to the attribute **cName** of relation **country**.
- The attributes **party** and **ctry** of relation **leader** are a foreign key to the attributes **pName** and **ctry** of relation **party**.

Question 2.1 (6 Points)

Write an SQL query that returns the names of leader of european parties (in a county in Europe) that did lead their party in a year between 1995 (inclusive) and 2000 (inclusive). Make sure that your query returns each such leader only once.

Solution

```
SELECT DISTINCT lName
FROM leader l, party p, country c
WHERE p.pName = l.party
      AND p.ctrY = l.ctrY
      AND p.ctrY = c.cName
      AND year BETWEEN 1995 AND 2000
      AND continent = 'Europe';
```

Question 2.2 (8 Points)

Write an SQL statement that returns the names of the leaders with the greatest number of total years as party leaders. For example, if the database contains three leaders A, B, and C where leader A did lead parties for a total of 20 years and the other two lead parties for 10 years, then the name of A should be returned. Note that the total number of years a person did lead any party should be counted, e.g., if one persons leads two parties for 5 years each, then this should be counted as 10 years.

Solution

```
SELECT lName
FROM leader
GROUP BY lName
HAVING count(*) = (SELECT max(tltYears)
                  FROM (SELECT count(*) AS tltYears
                        FROM leader
                        GROUP BY lName) AS domination);
```

Question 2.3 (10 Points)

Write a query that returns each country paired with the number of conservative parties (leaning is 'conservative') in this country and the number of conservative parties in the continent the country is located in. For example, if there are 5 conservative parties in USA and 10 conservative parties in America (the continent in which USA is located in), then one of the returned rows would be (*USA*, 5, 10). Ensure that your query also returns countries without conservative parties.

Solution

```
SELECT c.cName,
       COALESCE(ctryParties, 0) AS ctryParties,
       COALESCE(contParties, 0) AS contParties
FROM
  country c
  LEFT OUTER JOIN
    (SELECT count(*) AS ctryParties, ctry AS cName
     FROM party
     WHERE leaning = 'conservative'
     GROUP BY ctry) AS ctr
  ON (c.ctry = ctr.ctry)
  LEFT OUTER JOIN
    (SELECT count(*) AS contParties, continent
     FROM party p, country c
     WHERE leaning = 'conservative' AND ctry = cName
     GROUP BY continent) AS con
  ON (c.continent = con.contient);
```

Experienced people may use analytical functions (OVER clause).

Part 3 Relational Algebra (Total: 20 Points)

Question 3.1 Relational Algebra (8 Points)

Write an relational algebra expression over the schema from the SQL part (part 2) that returns all countries which have at least one conservative party, but no socialist party (the party *leaning*). Use the **set semantics** version of relational algebra.

Solution

$$\begin{aligned}cParties &= \pi_{ctry}(\sigma_{leaning='conservative'}(party)) \\sParties &= \pi_{ctry}(\sigma_{leaning='socialist'}(party)) \\q &= cParties \triangleright sParties\end{aligned}$$

Alternatively, set difference to remove countries with socialist parties.

Question 3.2 Equivalences (12 Points)

Consider the following relational schemata:

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

Check the equivalences that are correct (**bag semantics**). For example $R \ltimes R \equiv R$ should be checked, whereas $R \equiv S$ should not be checked.

☐ $\pi_{A,B}(R \bowtie \rho_{C \leftarrow A}(R)) \equiv R$

☐ $\delta(R \bowtie \rho_{C \leftarrow A}(R)) \equiv \delta(R)$

☒ $\delta(R \bowtie \delta(S)) \equiv \delta(R \bowtie S)$

☒ ${}_B\alpha(R) \equiv \delta(\pi_B(R))$

☐ $S \triangleright (S - R) \equiv S \ltimes (R \cap S)$

☐ $R \ltimes S \equiv S \triangleright R$

Part 4 Index Structures (Total: 22 Points)

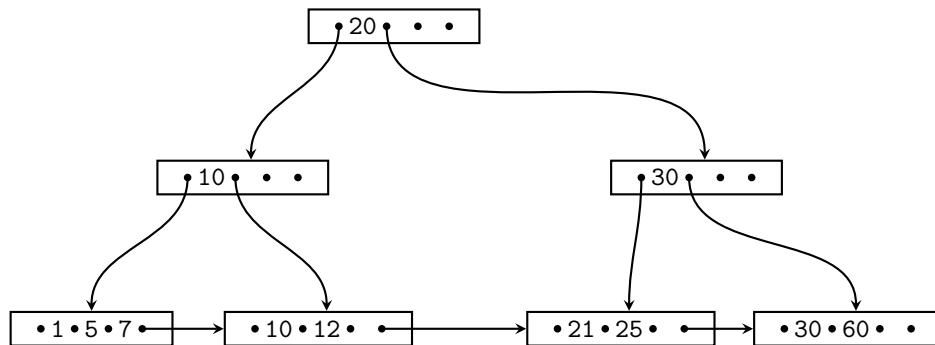
Question 4.1 B+-tree Operations (22 Points)

Given is the B+-tree shown below ($n = 3$). Execute the following operations and write down the resulting B+-tree after each step:

delete(10),delete(60),insert(8),insert(29)

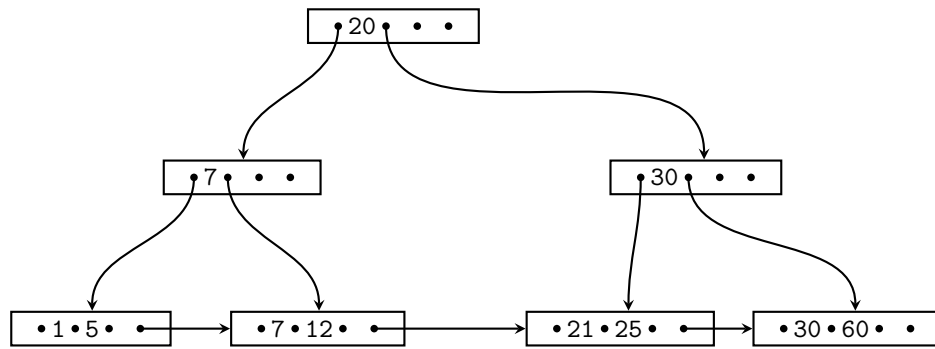
When splitting or merging nodes follow these conventions:

- **Leaf Split:** In case a leaf node needs to be split, the left node should get the extra key if the keys cannot be split evenly.
- **Non-Leaf Split:** In case a non-leaf node is split evenly, the “middle” value should be taken from the right node.
- **Node Underflow:** In case of a node underflow you should first try to redistribute and only if this fails merge. Both approaches should prefer the left sibling.

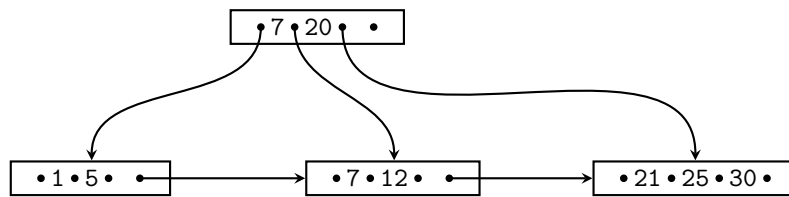


Solution

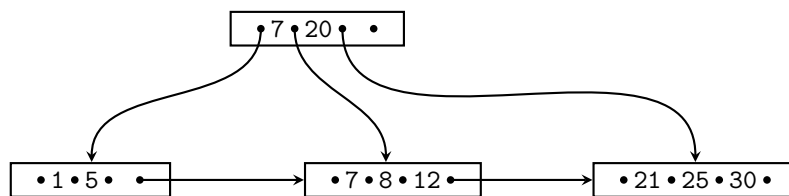
delete(10)



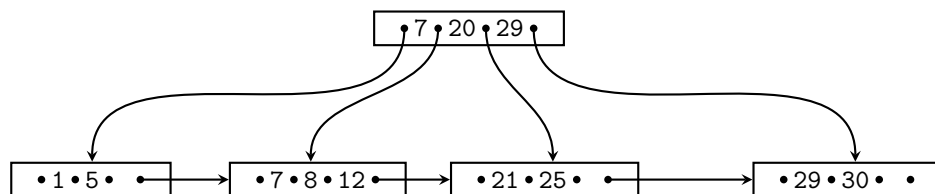
delete(60)



insert(8)



insert(29)



Part 5 Operator Implementations (Total: 19 Points)

Question 5.1 External Sorting (8 = 4 + 4 Points)

Mention two methods that improve external sorting using a priority queue and describe how they work.

Solution

1. A priority queue can be used to reduce the number of comparisons during merging of runs in external sorting. If we do an M -way merge, then to find the next smallest elements from the M runs we have to compare the current value in each run which is $O(M)$. If the current elements from each run are stored in a priority queue, then finding the lowest value is $O(\log_2(n))$. For this to work correctly, we have to store for each value in the queue from which run it originates. If an element e is removed from the queue, we have to insert the next value from the run of e into the queue (if it exists).
2. A priority queue can be used to generate longer runs. While reading the table we insert elements in a priority queue. Once the queue is full, we repeatedly remove the smallest element e from the queue. If e is larger than the largest element with have written to the current run so far, then e is written to the current run. Otherwise, we have to create a new run. After each element removal, we insert a new element from the input into the queue. This process is repeated until we have read the complete input table.

Question 5.2 I/O Cost Estimation (11 = 3 + 3 + 3 + 2 Points)

Consider two relations R and S with $B(R) = 3,000,000$ and $B(S) = 900$. You have $M = 1,001$ memory pages available. Compute the minimum number of I/O operations needed to join these two relations using **block-nested-loop join**, **merge-join** (the inputs are not sorted), and **hash-join**. You can assume that the hash function evenly distributes keys across buckets. Justify your result by showing the I/O cost estimation for each join method. If there is a draw between two methods (two methods require the same number of I/Os) then explain why we still may choose one over the other (e.g., CPU, memory requirement, ...).

Solution

- **BNL**: S fits into memory.
 $\lceil \frac{B(S)}{M-1} \rceil \cdot [B(R) + \min(B(S), (M-1))] = 1 \cdot [3,000,000 + 900] = 3,000,900$ I/Os
- **MJ**: We can generate sorted runs of size 1,000 that means the number of sorted runs from R is not low enough to keep one page from each run in memory (3,000 runs for R). We need 2 merge passes for the sort, but can execute the last merge phase and join in one pass. $(5 \cdot B(R)) + B(S) = 5 \cdot 3,000,000 + 900 = 15,000,900$ I/Os.
- **HJ**: Build in memory hash table of S (assume that it fits) and stream R once probing the hash table. $3,000,000 + 900 = 3,000,900$ I/Os.

We would pick HJ over BNL, because HJ requires $N(R)$ probes to a hash table ($O(1)$ each) and BNL does $N(R) * N(S)$ comparisons.

