Name

CWID

Exam
1

Oct 22th, 2018

CS525 - Midterm Exam
Solutions

Please leave this empty! 1 2 3 4 Sum

Instructions

- Things that you are **not** allowed to use
  - Personal notes
  - Textbook
  - Printed lecture notes
  - Phone
- The exam is **75** 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 4 parts in this exam
  1. SQL (32)
  2. Relational Algebra (26)
  3. Index Structures (24)
  4. I/O Estimation (18)

Part 1 SQL (Total: 32 Points)

Consider the following database schema and instance:

location

lName	city	owner	sizeSf
Windsor Castle	Windsor	Queen	40,000
Big Ben	London	Public	3,500
Stonehedge	Amesbury	Public	14,000

account

witness	time	suspect	crimeId
Bob	10:30	Peter	1
Peter	10:30	Bob	1
Queen	11:00	Bob	2

crime

id	location	time	type	victim
1	Big Ben	10:30	murder	Alice
2	Windsor Castle	11:00	theft	Queen

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, **lName** is the primary key of relation **location**.
- The attribute **crimeId** of relation **account** is a foreign key to the attribute **id** of relation **crime**.

Question 1.1 (6 Points)

Write an SQL query that returns the names of persons that are both suspects for a theft in Windsor and for a murder in London. Make sure that your query returns each such person only once.

Solution

```
(SELECT suspect
FROM location l, account a, crime c
WHERE city = 'Windsor'
      AND c.type = 'theft'
      AND a.crimeId = c.id
      AND l.lName = c.location)
INTERSECT
(SELECT suspect
FROM location l, account a, crime c
WHERE city = 'London'
      AND c.type = 'murder'
      AND a.crimeId = c.id
      AND l.lName = c.location)
```

Question 1.2 (8 Points)

Write a query that returns witnesses that are not suspects for any crime.

Solution

```
SELECT DISTINCT witness
FROM account
WHERE witness NOT IN (SELECT suspect FROM account);
```

Question 1.3 (9 Points)

Write a query that returns for each witness and city, the number of accounts of this witness for crimes that took place in a location in that city.

Solution

```
WITH cityCrimes AS (
   SELECT id AS cid, city
   FROM location l, crime c
   WHERE l.lName = c.location
),
witnessCity AS (
   SELECT DISTINCT witness, city
   FROM account a, location l
),
accCount AS (
    SELECT witness, city, count(*) AS numAcc
    FROM (SELECT DISTINCT witness, crimeId
          FROM account) a,
         cityCrimes c
    WHERE a.crimeId = c.cid
    GROUP BY witness, city
)
SELECT w.witness, w.city,
       CASE WHEN numAcc IS NULL THEN 0 ELSE numAcc END AS numAcc
FROM witnessCity w
     LEFT OUTER JOIN
     accCount a ON (w.witness = a.witness AND w.city = a.city);
```

for grading we also except a solution that does not return pairs of witnesses and cities where the witness has not witnessed any crimes in that city.

Question 1.4 (9 Points)

Write an SQL query that returns for each city the crime with the most suspects. Return the city name and for each such crime the location, time, type, and victim.

Solution

```
WITH numSus AS (
    SELECT l.city, a.crimeId, count(DISTINCT suspect) AS nSus
    FROM account a, crime c, location l
    WHERE a.crimeId = c.id AND c.location = l.lName
    GROUP BY l.city, a.crimeId
),
maxSus AS (
    SELECT s.city, s. crimeId
    FROM numSus s
    WHERE s.nSus = (SELECT max(nSus)
                    FROM numSus s2
                    WHERE s2.city = s.city)
)
SELECT l.city, c.location, c.time, c.type, c.victim
FROM location l, crime c
WHERE l.location = c.lName
      AND c.id IN (SELECT crimeId
                   FROM maxSus s);
```

Part 2 Relational Algebra (Total: 26 Points)

Question 2.1 Relational Algebra (6 Points)

Write a relational algebra expression over the schema from the SQL part that returns cities in which no murder has taken place yet. Use the **bag semantics** version of relational algebra.

Solution

$$q_{murderCity} = \pi_{city}(location \bowtie_{lName=location} \sigma_{type=murder}(crime))$$
$$q = \delta(\pi_{city}(location) - q_{murderCity})$$

The duplicate elimination is optional since we did not require that no duplicates are returned.

Question 2.2 Relational Algebra (12 Points)

Write a relational algebra expression over the schema from the SQL part that returns for each crime, the type, victim, the city of the location of the crime, and the number of witnesses for the crime. Use the **bag semantics** version of relational algebra.

Solution

$$numW = {}_{crimeId}\alpha_{count(*)}(\delta(\pi_{witness,crimeId}(crime \bowtie_{crimeId=id} account)))$$
$$q = crime \rtimes_{id=crimId} numW$$

Question 2.3 Relational Algebra (8 Points)

Write a relational algebra expression over the schema from the SQL part that returns victims of crimes which are also suspects in some crime.

Solution

$$q = \delta(\pi_{suspect}(account \bowtie_{suspect=victim} \pi_{victim}(crime)))$$

Part 3 Index Structures (Total: 24 Points)

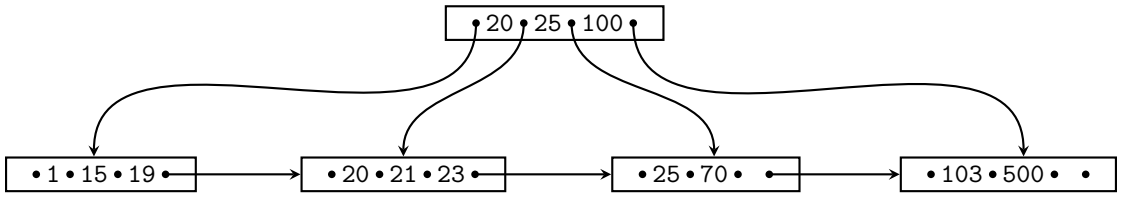
Question 3.1 B+-tree Operations (24 Points)

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

insert(22),delete(23),delete(500),insert(16)

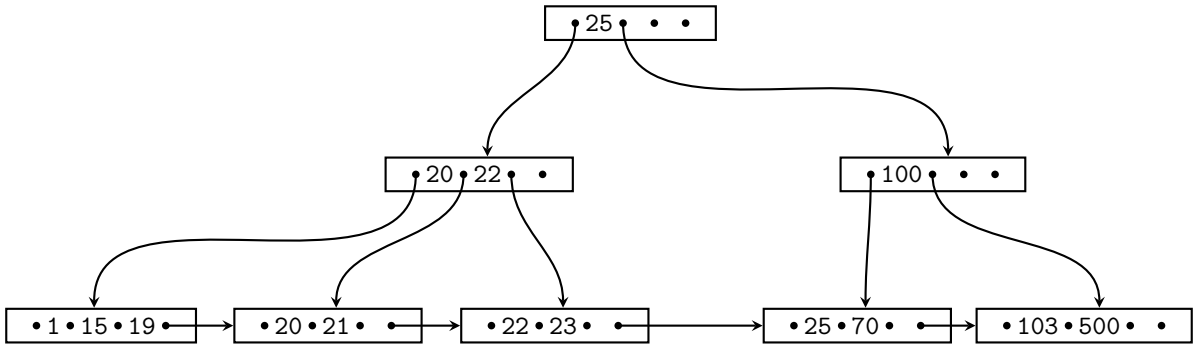
When splitting or merging nodes follow these conventions:

- **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.

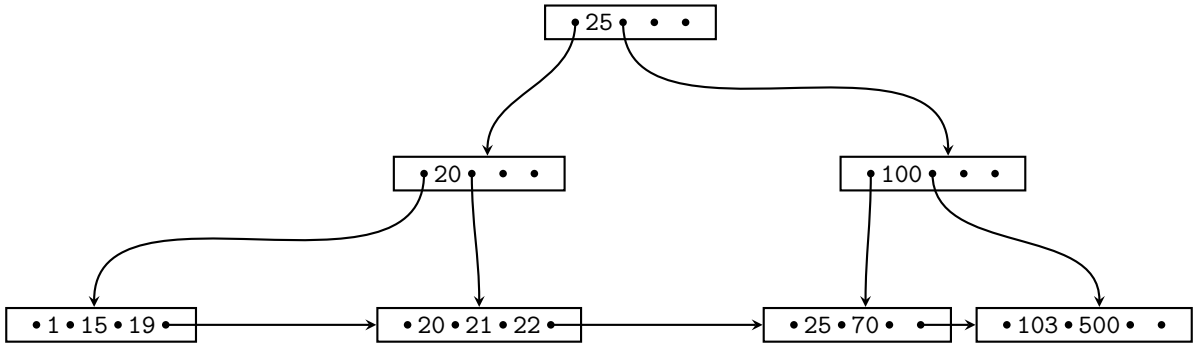


Solution

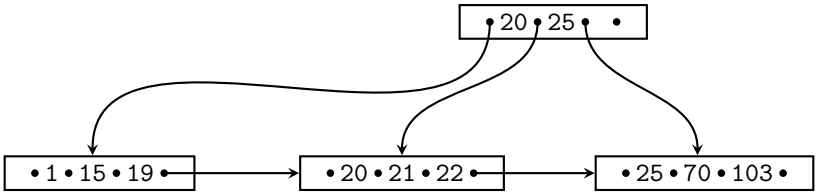
insert(22)



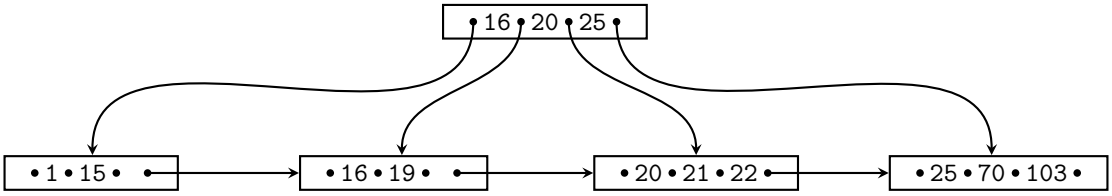
delete(23)



delete(500)



insert(16)



Part 4 I/O Estimation (Total: 18 Points)

Question 4.1 I/O Cost Estimation (12 = 4 + 4 + 4 Points)

Consider two relations R and S with $B(R) = 80,000,000$ and $B(S) = 200,000$. You have $M = 201$ memory pages available. Compute the 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 distributes keys evenly across buckets. Justify you result by showing the I/O cost estimation for each join method.

Solution

Block Nested-loop:
Use smaller table S as the inner. We only have $1,000$ chunks of size 200. Thus, we get $1,000 \times (200 + B(R))$ = 80,000,200,000 I/Os.

Merge-join:
Relation R can be sorted with three merge phases resulting in $4*2*B(R) = 640,000,000$ I/Os merging 10 runs in the last phase. Relation S requires two merge phases, merging 5 runs in the last phase: $3 \times 2 \times B(S) = 1,200,000$ I/Os. The last merge phase of relation S can be combined with the last merge phase of R ($10 + 5 = 15$ blocks of memory required). The merge join can be execute during these merge phases avoiding on read of relations R and S. Without optimizations we get $9 * B(R) + 7 * B(S) = 721,400,000$. If we execute the merge-join during the last merge phases we get $7 * B(R) + 5 * B(S) = 561,000,000$.

Hash-join:
We need two partitioning phases for the partitions of relation S to fit into memory. Thus, the hash-join requires $5 * (B(R) + B(S)) = 401,000,000$ I/Os.

Question 4.2 External Sorting (6 Points)

Consider a relation R with $B(R) = 500,000$. Assume that $M = 127$ memory pages are available for sorting. How many I/O operations are needed to sort this relation using no more than M memory pages.

Solution

External sorting requires $2 \times (1 + \lceil \log_{M-1}(\frac{B(R)}{M}) \rceil) \times B(R) = 2 \times 3 \times 500,000 = 3,000,000$ I/Os.