Name

CWID

Exam
1

October 20th, 2015

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 **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. 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 suspects for murders on locations owned by the `Queen`. Make sure that your query returns each such person only once.

Solution

```
SELECT DISTINCT suspect
FROM account a, crime c, location l
WHERE a.crimeId = c.id
      AND c.type = 'murder'
      AND l.owner = 'Queen'
      AND l.lName = c.location;
```

Question 1.2 (8 Points)

Write an SQL query that returns total size (sizeSf) of the real estate (locations) owned by who owns the most real estate (in terms of total size of locations owned by this person).

Solution

```
SELECT DISTINCT sum(sizeSf)
FROM location
GROUP BY owner
HAVING sum(sizeSf) = (SELECT max(sumSize)
                      FROM (SELECT sum(sizeSf) AS sumSize
                            FROM location
                            GROUP BY owner) ttlPerOwner);
```

Question 1.3 (8 Points)

Write a query that returns pairs of suspects that have provided account suspecting each other of the same crime, e.g., in the example instance Bob is suspecting Peter to be the murderer of Alice while at the same time Peter is suspecting Bob to be her murderer. Make sure to return each such pair only once. Hint: (Bob,Peter) and (Peter,Bob) are the same pair of people.

Solution

```
SELECT DISTINCT l.witness,l.suspect
FROM account l, account r
WHERE l.crimeId = r.crimeId
      AND l.witness = r.suspect
      AND l.suspect = r.witness
      AND l.witness < r.suspect;
```

DISTINCT is needed here, because there may be multiple crimes that two persons are accusing each other of.

Question 1.4 (10 Points)

Write a query that returns for each city the number crimes per type, e.g., the number of murders in London, the number of thefts in Windsor, and so on. For each city, crime type pair return how many of these crimes have been solved. A crime is considered solved if there is at least one account for this crimes and all accounts for that crime agree on the suspect. For example, in the example database the murder at Big Ben is not solved (contradictory accounts) whereas the theft at Windsor Castle is (one account cannot contradict itself).

Solution

```
WITH solvedCrimes AS (
    SELECT city, type, crimeId
    FROM location l, account a, crime c
    WHERE l.lName = c.location AND c.id = a.crimeId
    GROUP BY city, type, crimeId
    HAVING count(DISTINCT suspect) = 1
),

numSolved AS (
   SELECT city, type, count(*) AS numSolved
   FROM solvedCrimes
   GROUP BY city, type
),

numCrimes AS (
    SELECT city, type, count(*)
    FROM location l, crime c
    WHERE l.lName = c.location
    GROUP BY city, type
)

SELECT DISTINCT l.city, c.type,
                CASE WHEN numTotal IS NULL THEN 0 ELSE numTotal END AS numTotal,
                CASE WHEN numSolved IS NULL THEN 0 ELSE numSolves END AS numSolved
FROM location l, crime c
     LEFT OUTER JOIN numSolved s ON (l.city = s.city AND c.type = s.type)
     LEFT OUTER JOIN numCrimes n ON (l.city = n.city AND c.type = n.type);
```

Part 2 Relational Algebra (Total: 26 Points)

Question 2.1 Relational Algebra (6 Points)

Write an relational algebra expression over the schema from the SQL part that returns the time, type, and victim for all crimes in Windsor (the location city). Use the **set semantics** version of relational algebra.

Solution

$$q = \pi_{time,type,victim}(crime \ltimes_{location=lName} \sigma_{city=Windsor}(location))$$

Question 2.2 Relational Algebra (12 Points)

Write an relational algebra expression over the schema from the SQL part that returns for each person the number of crimes the person is accused of committing. Do not count multiple witnesses for the same crime twice (e.g., if Peter is accused by both Alice and Bob of committing a crime with id 1 then this counts as one crime and not 2). Persons that are not accused of committed any crime should be returned too. Use the **bag semantics** version of relational algebra.

Solution

$$
\begin{aligned}
persons &= \pi_{p,0\rightarrow cnt}(\delta(\pi_{witness\rightarrow p}(account) \cup \pi_{suspect}(account) \cup \pi_{victim}(crime))) \\
numAcc &= {}_{suspect}\alpha_{count(*)}(\delta(\pi_{suspect,crimeId}(account))) \\
q &= {}_{p}\alpha_{max(cnt)}(persons \cup numAcc)
\end{aligned}
$$

Question 2.3 Relational Algebra (8 Points)

Write an relational algebra expression over the schema from the SQL part that returns cities where no crimes have been taken place so far (there was no crime in any location that is in this city). Use the **set semantics** version of relational algebra.

Solution

$$citiesWCrime = \pi_{city}(location \bowtie_{lName=location} crime)$$
$$q = \pi_{city}(location) - citiesWCrime$$

Part 3 Index Structures (Total: 24 Points)

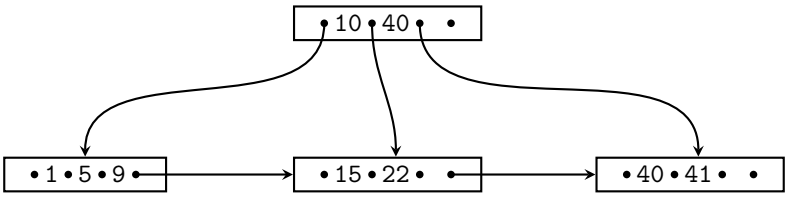
Question 3.1 B+-tree Operations (24 Points)

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

insert(6),insert(4),insert(3),delete(40)

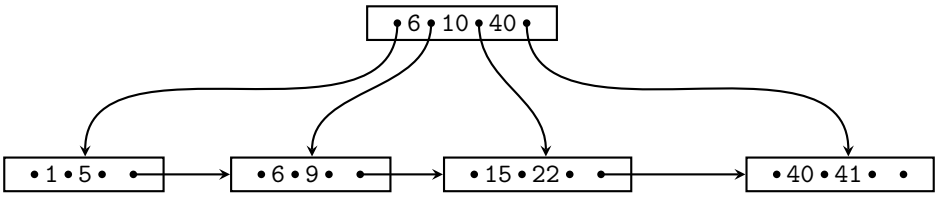
When splitting or merging nodes follow these conventions:

- **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.

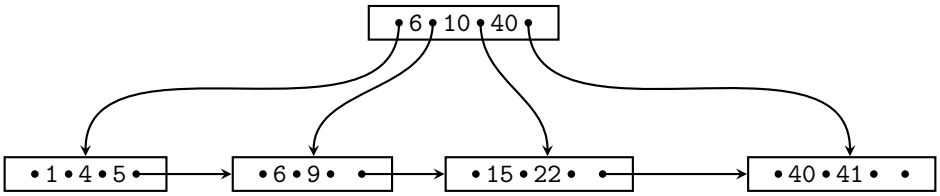


Solution

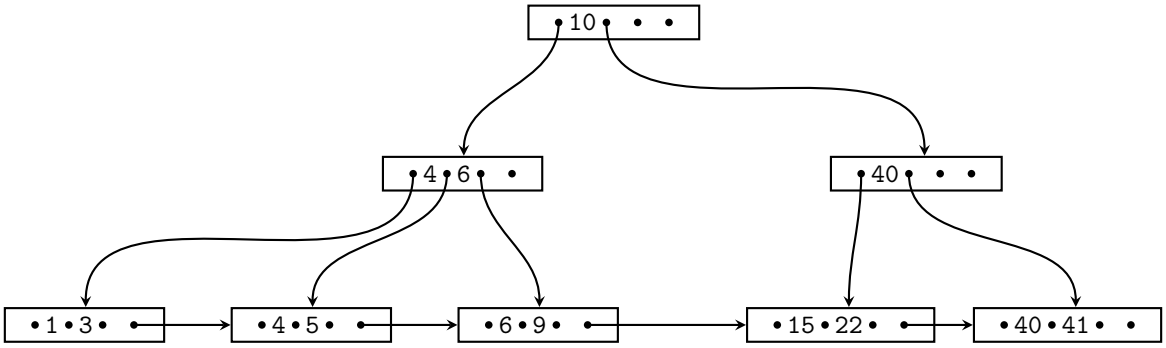
insert(6)



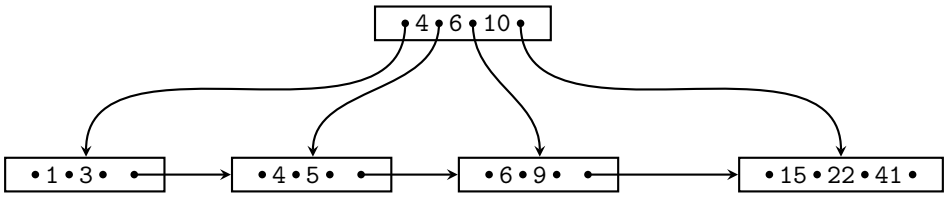
insert(4)



insert(3)



delete(40)



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 $B(R) = 100$ and $B(S) = 2,000$. You have $M = 101$ 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 evenly distributes keys across buckets. Justify you result by showing the I/O cost estimation for each join method.

Solution

Block Nested-loop:
Use smaller table R as the inner. We only have one chunk of size $100 = B(R)$. Thus, we get $1 \times (B(R) + B(S))$ = 2,100 I/Os.
Merge-join:
Relation R can be sorted in memory resulting in $2 * B(R) = 200$ I/Os. Relation S requires one merge phase, merging 20 runs: $2 \times 2 \times B(S) = 8,000$ I/Os. The last merge phase of relation S cannot be combined with sorting R (121 blocks of memory required). However, the merge join can be execute during this merge phase avoiding one read of relation S. Without optimizations we get $8,200 + B(R) + B(S) = 10,300$. If we execute the merge-join during the last merge phase for S we get $8,200 + B(R) = 8,300$.
Hash-join:
Relation R fits into memory. Thus, the hash-join requires $B(R) + B(S) = 2,100$ I/O.

Question 4.2 External Sorting (6 Points)

Consider a relation R with $B(R) = 10,000,000$. Assume that $M = 101$ 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 $2 \times (1 + \lceil \log_{M-1}(\frac{B(R)}{M}) \rceil) \times B(R) = 2 \times 4 \times 10,000,000 = 80,000,000$ I/Os.