Name

CWID

Exam
2

Dec 7th, 2018

CS525 - Final Exam
Solutions

Please leave this empty! 1 2 3 4 5 6 7

Sum

Instructions

- Things that you are **not** allowed to use
  - Personal notes
  - Textbook
  - Printed lecture notes
  - Phone
  - Calculator
- The exam 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. You do not have to answer every subquestion. 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. If you answer 3 questions correct, 1 incorrect, and do not answer 2, then you get $3 - 1 = 2$ points. ...
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are 7 parts in this exam (100 points total)
  1. SQL (26)
  2. Relational Algebra (15)
  3. Index Structures (20)
  4. I/O Estimation (12)
  5. Result Size Estimation (12)
  6. Schedules (15)

Part 1 SQL (Total: 26 Points)

Consider the following tvshow database schema and instance:

show

title	genre	rating	costPerEpisode
Game of Thrones	Fantasy	7.8	32,000,000
Monk	Crime	6.2	1,500,000
Dick Tracy	Crime	4.3	800,000
Ancient Aliens	Conspiracy	2.2	200,000

channel

cName	category	networth
Fox News	news	4,000,000,000
HBS	subscription	3,000,000,000
History	documentary & aliens	400,000,000
MSMBC	news	5,000,000,000

timeslot

tId	start	end	avgViewers
1	00:00	02:00	500,000
2	02:00	04:00	300,000
3	04:00	06:00	50,000
4	06:00	08:00	1,000,000
5	08:00	10:00	3,000,000
6	10:00	12:00	2,000,000
7	12:00	10:00	1,000,000
8	14:00	16:00	1,500,000
9	16:00	18:00	4,000,000
10	18:00	20:00	6,000,000
11	20:00	22:00	4,500,000
12	22:00	00:00	2,300,000

schedule

date	tId	cName	show
2013-01-01	11	HBS	Monk
2013-01-01	11	History	Ancient Aliens
2013-01-01	12	Fox News	Dick Tracy
2013-01-02	1	HBS	Game of Thrones

episodes

showtitle	eTitle	season	episode
Monk	Monkish	1	1
Monk	Hello World	1	2
Dick Tracy	The Murder Popoerder	15	6

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, **title** is the primary key of relation **show**.
- The attributes **tId** of relation **schedule** is a foreign key to relation **timeslot**.
- The attributes **cName** of relation **schedule** is a foreign key to relation **channel**.
- The attributes **show** of relation **schedule** is a foreign key to relation **show**.
- Attribute **showtitle** of relation **episode** is a foreign key to relation **show**.

Question 1.1 (4 Points)

Write an SQL statement that returns the names (`cName`) of the channels with a networth that is higher than the average networth of all channel.

Solution

```
SELECT cName
FROM channel
WHERE networth > (SELECT avg(networth) FROM channel);
```

Question 1.2 (5 Points)

Write an SQL query that computes the total cost of each TV show. The total cost is the costPerEpisode multiplied with the number of episodes (table `episodes`) for this show.

Solution

```
SELECT title, costPerEpisode * count(*) AS totalCost
FROM show s, episodes e
WHERE s.title = e.showtitle
GROUP BY title, costPerEpisode;
```

Also ok to use join of course.

Question 1.3 (5 Points)

Write an SQL query that returns for each show the cost it has per viewer (the total number of viewers per show is the sum of `avgViewers` over all timeslots where the show is shown).

Solution

```
SELECT title, costPerEpisode / sum(avgViewers) AS costPerViewer
FROM show s, schedule d, timeslot t
WHERE s.title = d.show AND d.tId = t.tId
GROUP BY s.title
```

Question 1.4 (6 Points)

Write an SQL query that returns the show with the highest amount of total viewers per `category`. For that calculate the number of viewers by summing up the average viewers for each time slot when the show is shown (table `schedule`).

Solution

```
WITH viewersPerChannel AS (
  SELECT s.cName, category, sum(avgViewers) AS totalViewers
  FROM channel c, schedule s, timeslot t
  WHERE c.cName = s.cName AND s.tId = t.tId
  GROUP BY s.cName, category
  )
SELECT cName, category
FROM viewersPerChannel v1
WHERE v1.totalViewers = (SELECT max(v2.totalViewers)
                         FROM viewersPerChannel v2
                         WHERE v2.category = v1.category);
```

Question 1.5 (6 Points)

Write an SQL query that returns shows (their title and genre) with high costs (above $1,000,000 per episode) that have low ratings (3.0 or lower), have less than 2,000,000 viewers in total, and have run for more than 3 seasons.

Solution

```
SELECT title, genre
FROM show s
WHERE rating < 3.0
      AND costPerEpisode > 1000000
      AND 2000000 < (SELECT sum(avgViewers)
                     FROM timeslot t, schedule d
                     WHERE t.tId = d.tId AND d.show = s.title)
      AND title IN (SELECT showtitle
                    FROM episodes
                    GROUP BY showtitle
                    HAVING max(season) > 3)
```

Part 2 Relational Algebra (Total: 15 Points)

Question 2.1 Relational Algebra (4 Points)

Write a relational algebra expression over the schema from the SQL part that returns the titles of all episodes for a named "Monk". Use the **bag semantics** version of relational algebra.

Solution

$$\pi_{eTitle}(\sigma_{showtitle=Monk}(episodes))$$

Question 2.2 Relational Algebra (5 Points)

Write a relational algebra expression over the schema from the SQL part that returns the combined networth of all `news` channels. Use the **bag semantics** version of relational algebra.

Solution

$$\alpha_{sum(networth)}(\sigma_{category=news}(channel))$$

Question 2.3 Relational Algebra (6 Points)

Write a relational algebra expression over the schema from the SQL part that returns the average rating of shows per channel. Use the **bag semantics** version of relational algebra.

Solution

$$_{cName}\alpha_{avg(rating)}(schedule \bowtie_{title=show} show \bowtie_{cName=cName} channel)$$

Part 3 Index Structures (Total: 20 Points)

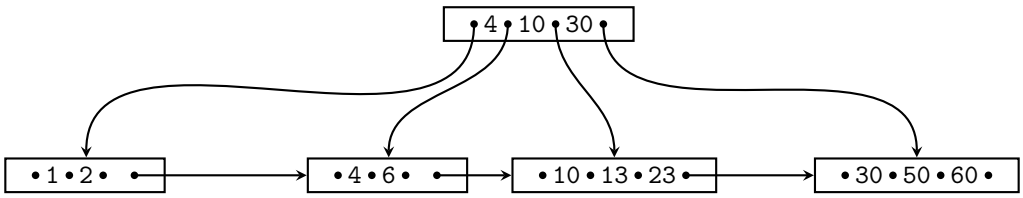
Question 3.1 B+-tree Operations (20 Points)

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

insert(55), insert(7), delete(13), delete(23), insert(3)

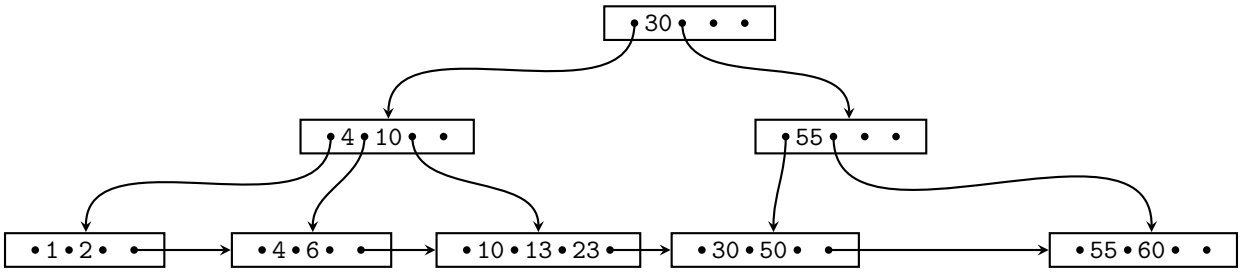
When splitting or merging nodes follow these conventions:

- **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.

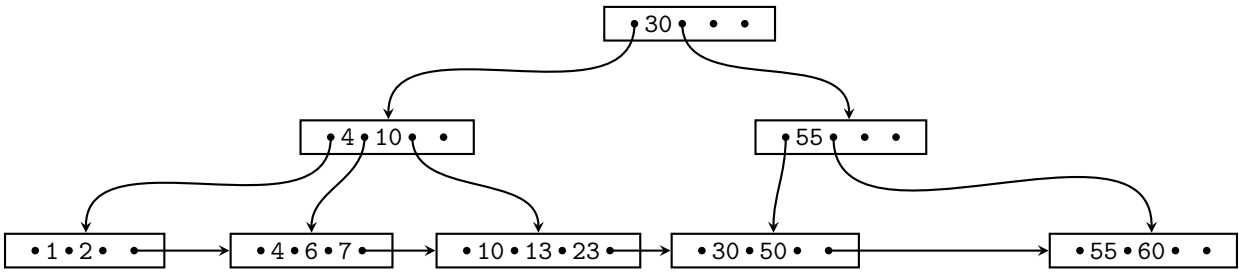


Solution

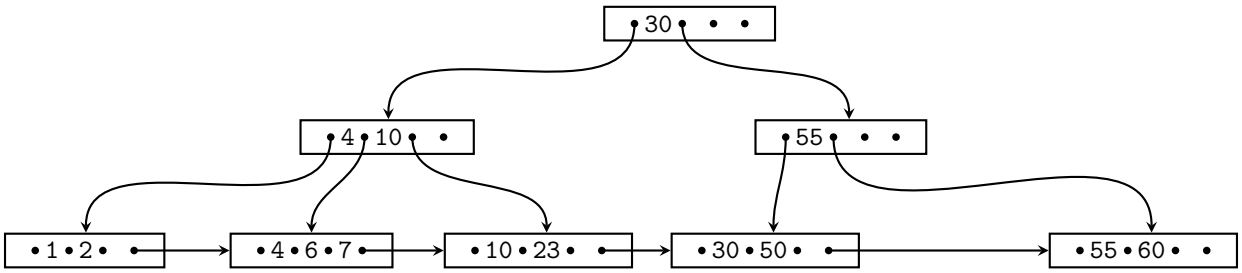
insert(55)



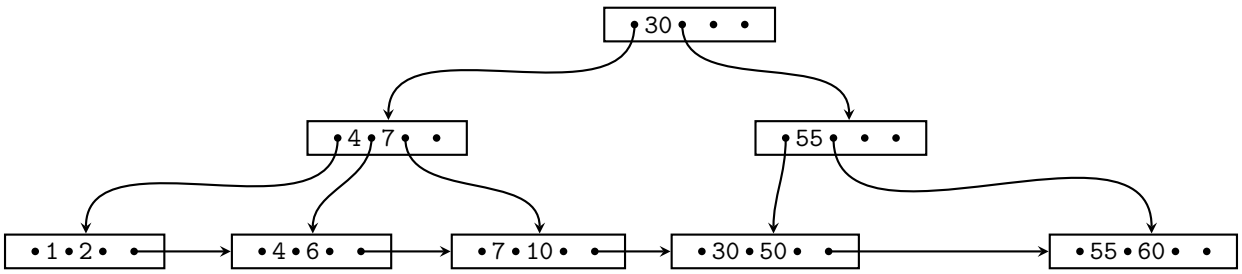
insert(7)



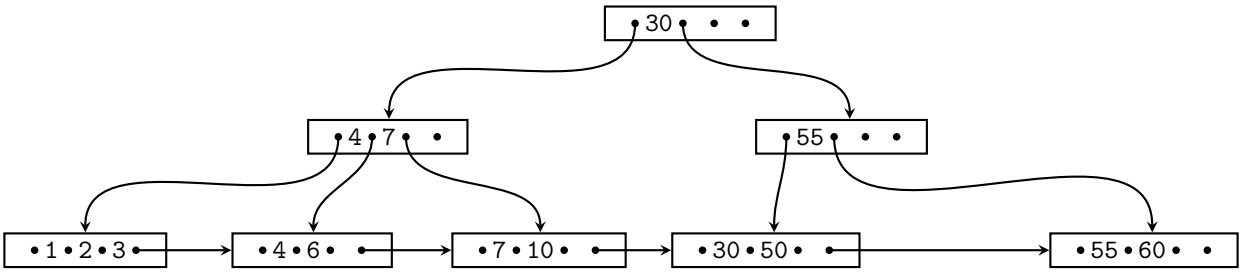
delete(13)



delete(23)



insert(3)



Part 4 I/O Cost Estimation (Total: 12 Points)

Question 4.1 External Sorting (6 Points)

You have $M = 101$ memory pages available and should sort a relation R with $B(R) = 2,000$ blocks. Estimate the number of I/Os necessary to sort R using the external merge sort algorithm introduced in class.

Solution

$$\begin{aligned} IO &= 2 \cdot B(R) \cdot (1 + \lceil log_{M-1}(\frac{B(R)}{M}) \rceil) \\ &= 2 \cdot 2,000 \cdot (1 + 1) \\ &= 8,000 \end{aligned}$$

Question 4.2 External Sorting (6 Points)

You have $M = 101$ memory pages available and should sort a relation R with $B(R) = 800,000$ blocks. Estimate the number of I/Os necessary to sort R using the external merge sort algorithm introduced in class.

Solution

$$\begin{aligned} IO &= 2 \cdot B(R) \cdot (1 + \lceil log_{M-1}(\frac{B(R)}{M}) \rceil) \\ &= 2 \cdot 800,000 \cdot (1 + 2) \\ &= 4,800,000 \end{aligned}$$

Part 5 Result Size Estimations (Total: 12 Points)

Consider the table *show* relation from the SQL part with attributes *title*, *genre*, *rating*, and *costPerEpisode*. Attribute *title* is the primary key of this relation.

Given are the following statistics:

$$
\begin{array}{rrr}
T(show) = 2,000 & & \\
V(show, title) = 2,000 & & \\
V(show, genre) = 10 & & \\
V(show, rating) = 50 & min(rating) = 0.0 & max(rating) = 10.0 \\
V(show, costPerEpisode) = 500 & min(costPerEpisode) = 100,000 & max(costPerEpisode) = 2,099,999
\end{array}
$$

Question 5.1 Estimate Result Size (5 Points)

Estimate the number of result tuples for the query $q = \sigma_{genre=Fantasy \wedge costPerEpisode \geq 1,000,000}(show)$ using the first assumption presented in class (values used in queries are uniformly distributed within the active domain).

Solution

Calculate probability that a tuple fulfills the conditions using the independence assumption.

$$P(genre = Fantasy \wedge costPerEpisode \geq 1,000,000) = \frac{1}{10} \cdot \frac{2,099,999 - 1,000,000 + 1}{2,099,999 - 100,000 + 1} = \frac{1}{20} = 0.05$$

$$T(q) = T(show) * P(genre = Fantasy \wedge costPerEpisode \geq 1,000,000) = 2,000 \cdot 0.05 = 10$$

Question 5.2 Estimate Result Size (7 Points)

Estimate the number of result tuples for the query $q = \sigma_{genre=Crime \vee genre=News}(show)$ using the first assumption presented in class.

Solution

Since $genre = Crime$ and $genre = News$ are disjoint events we can sum up their probabilities to calculate $P(genre = Crime \vee genre = News)$.

$$T(q) = \frac{2}{V(show, genre)}) \cdot T(show) = \frac{2}{10} \cdot 2,000 = 400$$

We also accept the standard formula for disjunction even though it is not the right one to apply here.

$$T(q) = (1 - [(1 - \frac{1}{V(show, genre)}) \cdot (1 - \frac{1}{V(show, genre)})]) \cdot T(show) = (1 - [(1 - \frac{1}{10}) \cdot (1 - \frac{1}{10})] \cdot 2,000 = 380$$

Part 6 Schedules (Total: 15 Points)

Question 6.1 Schedule Classes (15 = 5 + 5 + 5 Points)

Indicate which of the following schedules belong to which class. **Every correct answer is worth 1 point. Every incorrect answer results in 1 point being deducted. You are allowed to skip questions (0 points).** 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

$$S_1 = w_1(A), w_4(A), r_2(B), r_3(A), w_1(A), w_4(B), c_1, c_2, c_3, c_4$$
$$S_1 = r_4(B), w_1(B), w_1(D), c_1, r_3(D), w_3(C), c_3, r_2(C), w_2(A), r_4(A), c_2, c_4$$
$$S_3 = w_4(A), w_1(B), r_1(A), w_3(B), w_4(C), r_2(B), w_2(B), c_2, c_1, c_3, c_4$$

S_1 is recoverable	■ no	❑ yes
S_1 is cascade-less	■ no	❑ yes
S_1 is strict	■ no	❑ yes
S_1 is conflict-serializable	■ no	❑ yes
S_1 is 2PL	■ no	❑ yes

S_3 is recoverable	❑ no	■ yes
S_3 is cascade-less	■ no	❑ yes
S_3 is strict	■ no	❑ yes
S_3 is conflict-serializable	■ no	❑ yes
S_3 is 2PL	■ no	❑ yes

S_2 is recoverable	■ no	❑ yes
S_2 is cascade-less	■ no	❑ yes
S_2 is strict	■ no	❑ yes
S_2 is conflict-serializable	❑ no	■ yes
S_2 is 2PL	❑ no	■ yes