Name

CWID

Exam
2

May 3rd, 2017

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

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

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

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**.

Question 1.1 (4 Points)

Write an SQL statement that returns the sum of the networth of channels per category.

Solution

```
SELECT category, sum(networth) AS totalNW
FROM channel
GROUP BY category
```

Question 1.2 (5 Points)

Write an SQL query that for each channel computes its total expenditure on tv shows. Assume that channels are showing each episode of a show at most once. That is, for each tv show it broadcasts during a time slot, the channel has to pay the `costPerEpisode` for this show.

Solution

```
SELECT sum(COALESCE(costPerEpisode,0)) AS total, cName
FROM show w LEFT OUTER JOIN schedule s ON (s.show = w.title)
GROUP BY cName
```

Also ok to not join and just use show or to not outer join.

```
SELECT sum(costPerEpisode) AS total, show
FROM show w
GROUP BY show
```

Question 1.3 (6 Points)

Write an SQL query that returns the name of the channel with the highest networth.

Solution

```
SELECT cName
FROM channel c
WHERE c.networth = (SELECT max(networth) FROM channel);
```

Question 1.4 (5 Points)

Write an SQL query that returns the expected number of viewers per time slot and channel. Assume that the expected number of viewers is calculated as the average number of viewers per timeslot (column `avgViewers`) multiplied by a factor that is based on the rating of the show being shown. This factor is calculated as $1+(0.1*(\texttt{rating}-5))$. For example, for a time slot with `avgViewers` = 1,000,000 and a show with rating 4 the expected number of viewers would be $1,000,000*(1+(0.1*(4-5))) = 1,000,000*(1-0.1) = 1,000,000*0.9 = 900,000$.

Solution

```
SELECT date, tId, cName, avgViewers * (1 + (0.1 * (rating - 5))) AS expView
FROM schedule s, show w
WHERE s.show = w.title
```

or if interpreted as aggregating this over all dates:

```
SELECT tId, cName, SUM(avgViewers * (1 + (0.1 * (rating - 5)))) AS expView
FROM schedule s, show w
WHERE s.show = w.title
GROUP By cName, tId
```

Question 1.5 (6 Points)

Write an SQL query that returns the channel (if such a channel exists) that has more expected viewers for every time slot of date `2017-01-01` than any other channel. Use the formula from the previous question to calculate the expected number of viewers per timeslot and channel.

Solution

```
WITH numView AS (
     SELECT tId, cName, avgViewers * (1 + (0.1 * ( 5 - rating))) AS expView
     FROM schedule s, show w
     WHERE s.show = w.title AND date = '2017-01-01'
   ),
   maxView AS (
     SELECT tId, cName
     FROM numView n
     WHERE expView = (SELECT max(a.expView) FROM numView a WHERE a.tId = n.tId)
   )
SELECT cName
FROM maxView
GROUP BY cName
HAVING count(*) = (SELECT count(*) FROM timeslot)
```

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 tId of timeslots with more than 1,000,000 viewers (avgViewers).

Solution

$$\pi_{tId}(\sigma_{avgViewers>1{,}000{,}000}(customer))$$

Question 2.2 Relational Algebra (5 Points)

Write a relational algebra expression over the schema from the SQL part that returns the number of `Crime` shows that are rated higher than 4.9. Use the **bag semantics** version of relational algebra.

Solution

$$\alpha_{count(*)}(\sigma_{rating>5.0 \wedge genre=Crime}(show))$$

Question 2.3 Relational Algebra (6 Points)

Write a relational algebra expression over the schema from the SQL part that returns the title of Fantasy shows broadcasted by news channels (`category`). Use the **bag semantics** version of relational algebra.

Solution

$$\pi_{title}(show \bowtie_{title=show} schedule \bowtie \sigma_{category=news}(channel))$$

Part 3 Index Structures (Total: 20 Points)

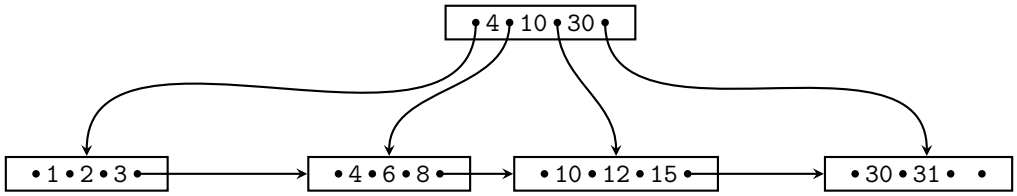
Question 3.1 B+-tree Operations (20 Points)

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

insert(7), insert(13), delete(12), delete(30), insert(32)

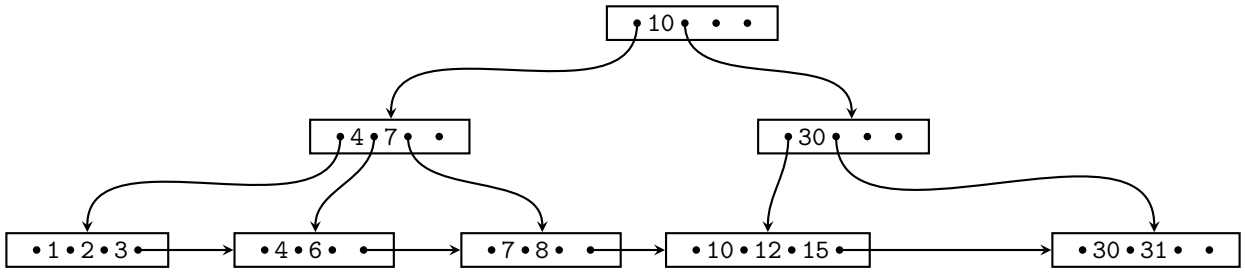
When splitting or merging nodes follow these conventions:

- **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.

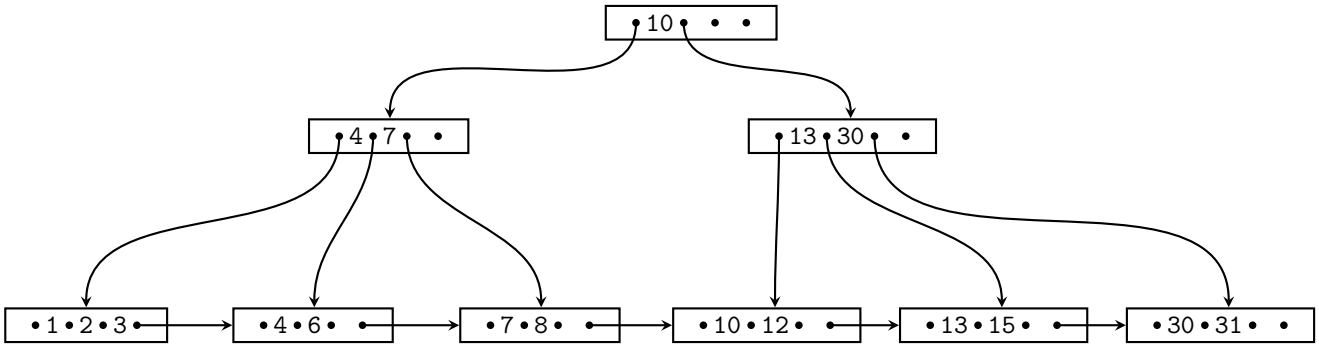


Solution

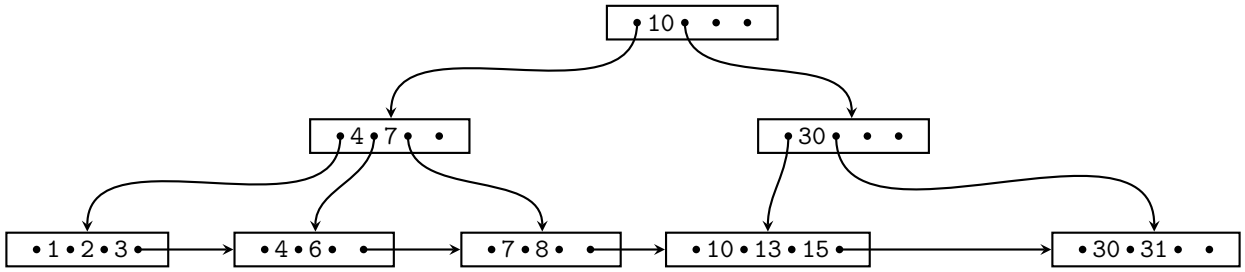
insert(7)



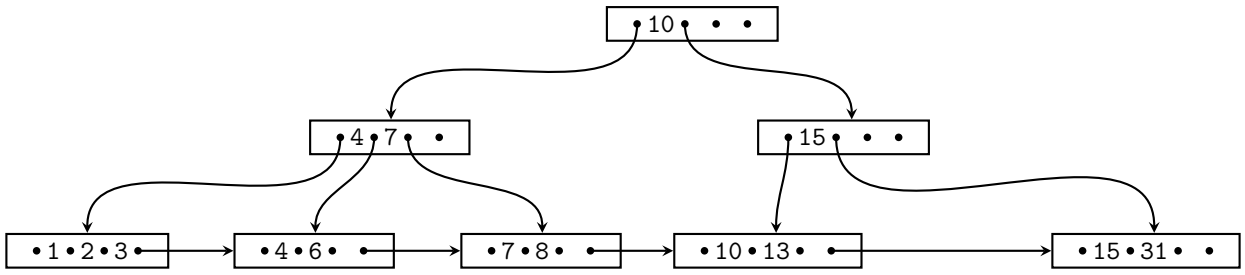
insert(13)



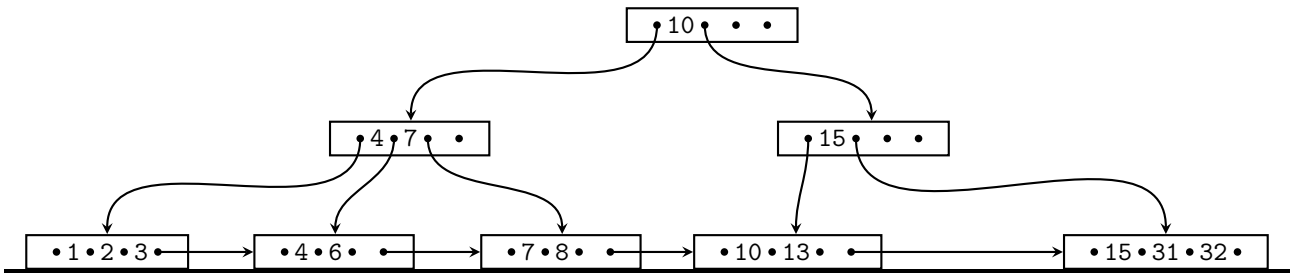
delete(12)



delete(30)



insert(32)



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

Question 4.1 External Sorting (3 Points)

You have $M = 1001$ memory pages available and should sort a relation R with $B(R) = 4,000,000$ blocks. Estimate the number of I/Os necessary to sort R using the external merge sort algorithm introduced in class.

Solution

$$\begin{aligned} IO &= 2 \cdot B(R) \cdot (1 + \lceil log_{M-1}(\frac{B(R)}{M}) \rceil) \\ &= 2 \cdot 4,000,000 \cdot (1 + 2) \\ &= 24,000,000 \end{aligned}$$

Question 4.2 External Sorting (3 Points)

You have $M = 5$ memory pages available and should sort a relation R with $B(R) = 200,000$ blocks. Estimate the number of I/Os necessary to sort R using the external merge sort algorithm introduced in class.

Solution

$$\begin{aligned} IO &= 2 \cdot B(R) \cdot (1 + \lceil log_{M-1}(\frac{B(R)}{M}) \rceil) \\ &= 2 \cdot 200,000 \cdot (1 + 8) \\ &= 3,600,000 \end{aligned}$$

Question 4.3 I/O Cost Estimation (6 = 2 + 2 + 2 Points)

Consider two relations R and S with $B(R) = 200,000$ and $B(S) = 200,000$. 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 you result by showing the I/O cost estimation for each join method.

Solution

- **BNL**: S is the smaller relation.
  $\lceil \frac{B(S)}{M-1} \rceil \cdot [B(R) + min(B(S), (M-1))] = 200 \cdot [200,000 + 1000] = 40,200,000$ I/Os
- **MJ**: We can generate sorted runs of size $1,000$ that means we need 1 merge pass for R and one for S. The number of runs in the last phase of sorting is 200 for both R and S - low enough to keep one page from each run in memory. Thus, we can execute the last merge phase and join in one pass. $3 \cdot (B(R) + B(S)) = 3 \cdot 400,000 = 1,200,000$ I/Os.
- **HJ**: After one partition phase the size of the partitions for R and S (200 pages) is small enough to fit one partition into memory, build an in-memory hash table of each partition of R (or S), and stream a partition of R (or S) once probing the hash table. $(2+1) \cdot (B(R) + B(S)) = 3 \cdot 400,000 = 1,200,000$ I/Os.

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:

$$
\begin{aligned}
T(show) &= 1,000 \\
V(show, title) &= 1000 \\
V(show, genre) &= 10 \\
V(show, rating) &= 50 \qquad min(rating) = 0.0 \qquad max(rating) = 10.0 \\
V(show, costPerEpisode) &= 500
\end{aligned}
$$

Question 5.1 Estimate Result Size (5 Points)

Estimate the number of result tuples for the query $q = \sigma_{genre=Fantasy \wedge rating \leq 5.0}(show)$ 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.

$$P(genre = Fantasy \wedge rating \leq 5.0) = \frac{5.0 - 0.0 + 1}{10.0 - 0.0 + 1} \cdot \frac{1}{10} = \frac{6}{11} \approx 0.54$$

$$T(q) = T(show) * P(genre = Fantasy \wedge rating \leq 5.0) = 1,000 \cdot 0.54 \approx 540$$

Alternative assumptions we would consider correct are:

1. Assume that all ratings have one digit behind the dot, i.e., there are 101 possible ratings. Under this assumption the probability would be $\frac{51}{101}$
2. Assume that ratings are contiguous, then the probability would be $\frac{5}{10}$

Question 5.2 Estimate Result Size (7 Points)

Estimate the number of result tuples for the query $q = \sigma_{title=GameofThrones \vee rating=10.0}(show)$ using the first assumption presented in class.

Solution

$$T(q) = (1 - [(1 - \frac{1}{V(show, title)}) \cdot (1 - \frac{1}{V(show, rating)})]) \cdot T(show) = (1 - [(1 - \frac{1}{1,000}) \cdot (1 - \frac{1}{50})] \cdot 1,000 \approx 21$$

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

$$S_1 = w_4(C), r_1(A), r_1(B), r_2(A), w_3(B), c_3, c_1, w_4(B), w_2(A), c_2, r_4(A), c_4$$
$$S_2 = w_1(A), w_2(B), r_2(A), w_3(B), w_1(C), r_4(B), w_4(B), c_4, c_2, c_3, c_1$$
$$S_3 = r_1(A), w_2(A), w_2(B), c_2, r_3(B), w_3(C), c_3, r_4(C), w_4(D), r_1(D), c_4, c_1$$

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_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

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