Name

CWID

Exam
2

December 9th, 2015

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. Disk Organization and Buffering (9)
  2. SQL (20)
  3. Relational Algebra (12)
  4. Index Structures (20)
  5. I/O Estimation (12)
  6. Result Size Estimation (12)
  7. Schedules (15)

Part 1 Disk Organization and Buffering (Total: 9 Points)

Question 1.1 Page Replacement Clock (9 Points)

Consider a buffer pool with 3 pages using the **Clock** page replacement strategy. Initially the buffer pool is in the state shown below. We use the following notation ${}^{flag}[page]^{dirty}_{fix}$ 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 arrow to the page frame that S is pointing to.

Current Buffer State

${}^{0}[3]_{1}$ ${}^{0}[2]_{0}$ $\mathbf{{}^{1}[1]_{0}}$

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

- p stands for pin
- u for unpin

```
p(5),p(7),u(3),u(5),p(6),p(8)
```

Solution

u(5)

$^{0}[3]_{1}\ ^{1}[5]_{1}\ \mathbf{^{0}[1]_{0}}$

p(7)

$\mathbf{^{0}[3]_{1}}\ ^{1}[5]_{1}\ ^{1}[7]_{1}$

u(3)

$\mathbf{^{0}[3]_{0}}\ ^{1}[5]_{1}\ ^{1}[7]_{1}$

u(5)

$\mathbf{^{0}[3]_{0}}\ ^{1}[5]_{0}\ ^{1}[7]_{1}$

p(6)

$^{1}[6]_{1}\ \mathbf{^{1}[5]_{0}}\ ^{1}[7]_{1}$

p(8)

$^{0}[6]_{1}\ ^{1}[8]_{1}\ \mathbf{^{0}[7]_{1}}$

Part 2 SQL (Total: 20 Points)

Consider the following database schema and instance:

hotel

hName	hCity	rating
Hilton	Chicago	3
Watertower Hotel	Chicago	10
Carton	New York	0

room

hotel	roomNum	price
Hilton	1	1000
Hilton	234	100
Carton	3	75

booking

customer	hotel	room	date	nights
Gert Gertsen	Hilton	3	2015-08-08	3
Frank Frankhold	Carton	1	2015-12-23	5

customer

cusName	prefPay	balance
Gert Gertsen	credit	0
Frank Frankhold	debit	300

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, **hName** is the primary key of relation **hotel**.
- The attributes **hotel** and **room** of relation **booking** is a foreign key to relation **room**.
- The attribute **customer** of relation **booking** is a foreign key to relation **customer**.
- The attribute **hotel** of relation **room** is a foreign key to relation **hotel**.

Question 2.1 (3 Points)

Write an SQL query that returns the average rating of hotels per city.

Solution

```
SELECT hCity, avg(rating)
FROM hotel
GROUP BY hCity
```

Question 2.2 (3 Points)

Write an SQL statement that returns the average price for a room per city.

Solution

```
SELECT hCity, avg(price) AS avgRoomPrice
FROM hotel h,
     room r
WHERE r.hotel = h.hName
GROUP BY hCity
```

Question 2.3 (5 Points)

Write an SQL query that returns customers ordered by the number of problematic bookings they have done (return customer name and the number of problematic bookings). A booking is considered problematic if the outstanding balance (attribute `balance` of relation `customer`) of the customer that did book the room is larger than the price of the reservation (the number of nights times the price of the room).

Solution

```
WITH probBook(name,price) AS
(
   SELECT cusName AS name, price * nights AS price
   FROM booking b, customer c, room r
   WHERE b.customer = c.cusName
         AND b.hotel = r.hotel
         AND b.room = r.roomNum
         AND (price * nights) < balance
),

probBookCount(name,numBook) AS
(
   SELECT name, count(*) AS numBook
   FROM probBook
   GROUP BY name
)

SELECT *
FROM probBookCount
ORDER BY numBook DESC;
```

Question 2.4 (4 Points)

Write an SQL query that returns for each hotel the 3 most important customers. Return pairs of hotels and customers ordered by hotel and then by importance. The importance of a customer is defined as the total price of all bookings of the customer at the hotel.

Solution

```
WITH totalBookPrice(hotel,customer,tot) AS
(
   SELECT hotel, customer, sum(price * nights)
   FROM booking b, room r
   WHERE b.hotel = r.hotel AND b.room = r.roomNum
   GROUP BY hotel, customer
)

SELECT hotel, customer
FROM totalBookPrice
WHERE RANK() OVER (PARTITION BY hotel ORDER BY tot DEC) <= 3
ORDER BY hotel ASC, tot DESC
```

Question 2.5 (5 Points)

Write an SQL query that returns for each hotel 1) whether the hotel was never booked (does not occur in the booking table at all) and 2) whether there is at least one room in the hotel that has never been booked. Return one tuple per hotel with a boolean attribute for 1) and 2). For instance, if the Hilton has some bookings (1 is false), but the room number 1 of this hotel was never booked then the result tuple for Hilton would be `(Hilton,false,true)`.

Solution

```
SELECT hName AS hotel,
       NOT EXISTS (SELECT *
                   FROM booking b
                   WHERE b.hotel = h.hName) AS notBooked,
       EXISTS (SELECT *
               FROM room r
               WHERE r.hotel = h.hName AND
                     (r.hotel,r.roomNum) NOT IN (SELECT b2.hotel,b2.room
                                                 FROM booking b2)) AS roomNotBooked
FROM hotel h
```

Part 3 Relational Algebra (Total: 12 Points)

Question 3.1 Relational Algebra (3 Points)

Write a relational algebra expression over the schema from the SQL part (part 2) that returns the names of customers with a non-zero balance. Use the **bag semantics** version of relational algebra.

Solution

$$\pi_{cusName}(\sigma_{balance>0}(customer))$$

Question 3.2 Relational Algebra (4 Points)

Write a relational algebra expression over the schema from the SQL part (part 2) that returns for each hotel the number of rooms that cost more than 999 dollars. You do not need to return hotels that do not have such rooms. Use the **bag semantics** version of relational algebra.

Solution

$$_{hotel}\alpha_{count(*)}(\sigma_{price>999}(room))$$

Question 3.3 Relational Algebra (5 Points)

Write a relational algebra expression over the schema from the SQL part (part 2) that returns double bookings, i.e., pairs of bookings (return all the booking relation attributes of each booking in such a pair) for the same hotel room where the dates overlap. Use the **bag semantics** version of relational algebra.

Solution

$$sameRoom = \rho_l(booking) \bowtie_{l.hotel=r.hotel \wedge l.room=r.room \wedge (l.customer \neq r.customer \vee l.date \neq r.date)} \rho_r(booking)$$
$$q = \sigma_{(l.date \leq r.date) \wedge (l.date+nights \geq r.date)}(sameRoom)$$

Part 4 Index Structures (Total: 20 Points)

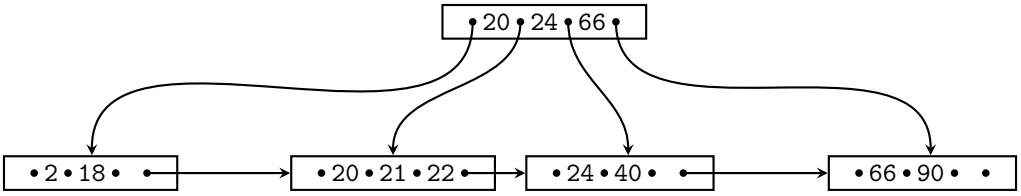
Question 4.1 B+-tree Operations (20 Points)

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

insert(23), insert(19), delete(20), delete(21), insert(41)

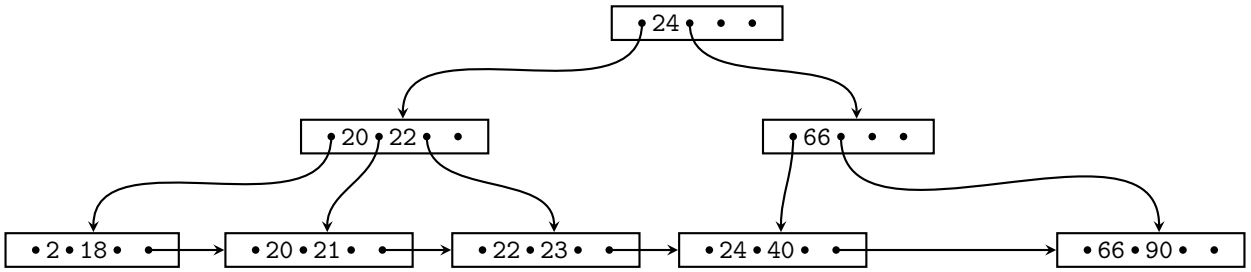
When splitting or merging nodes follow these conventions:

- **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.

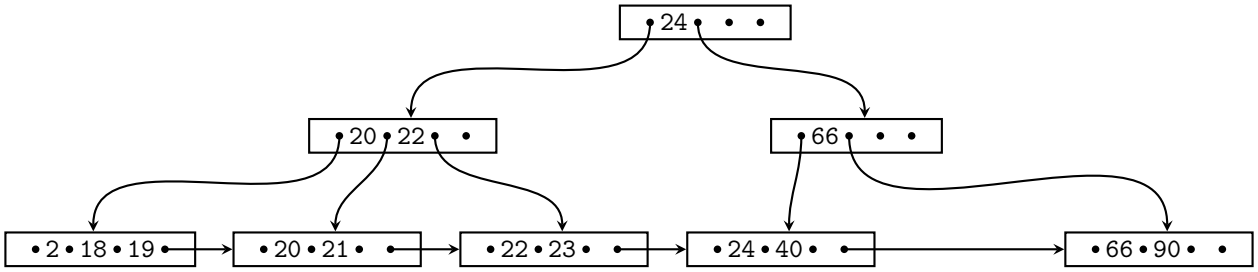


Solution

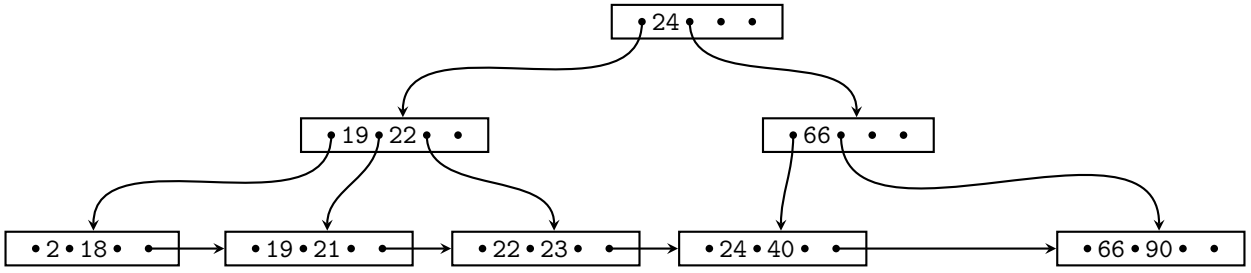
insert(23)



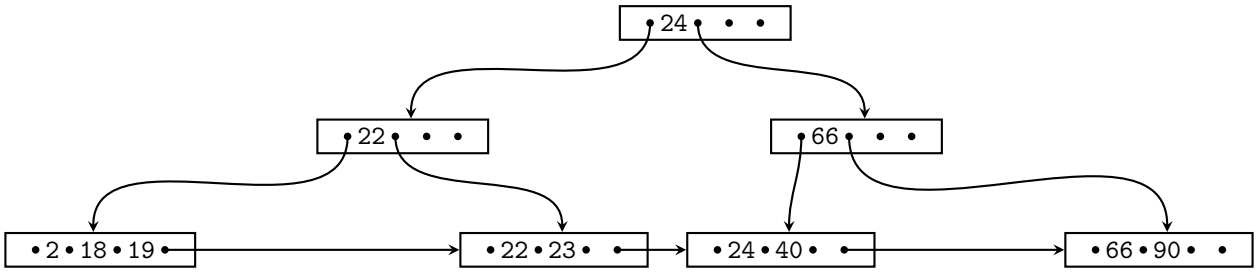
insert(19)



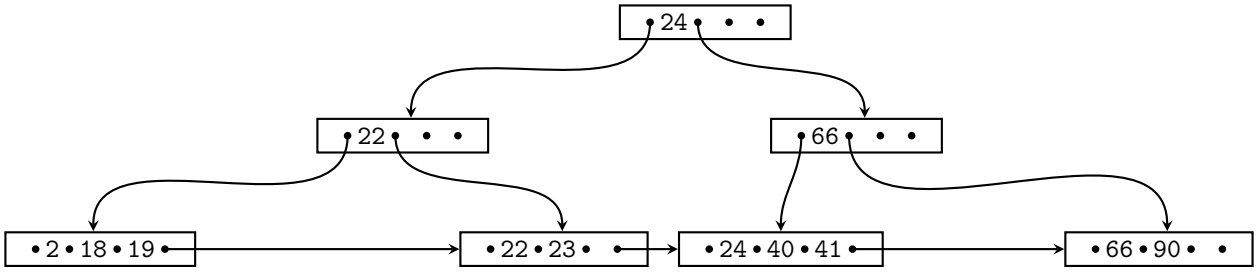
delete(20)



delete(21)



insert(41)



Part 5 I/O Cost Estimation (Total: 12 Points)

Question 5.1 External Sorting (3 Points)

You have $M = 1001$ memory pages available and should sort a relation R with $B(R) = 40,000$ blocks. Estimate the number of I/Os necessary to sort R using the external merge sort algorithm introduced in class.

Solution

$$\begin{aligned} IO &= 2 \cdot B(R) \cdot (1 + \lceil log_{M-1}(\frac{B(R)}{M}) \rceil) \\ &= 2 \cdot 40,000 \cdot (1 + 1) \\ &= 160,000 \end{aligned}$$

Question 5.2 External Sorting (3 Points)

You have $M = 3$ memory pages available and should sort a relation R with $B(R) = 1,200,000$ blocks. Estimate the number of I/Os necessary to sort R using the external merge sort algorithm introduced in class.

Solution

$$\begin{aligned} IO &= 2 \cdot B(R) \cdot (1 + \lceil log_{M-1}(\frac{B(R)}{M}) \rceil) \\ &= 2 \cdot 1,200,000 \cdot (1 + 19) \\ &= 48,000,000 \end{aligned}$$

Question 5.3 I/O Cost Estimation (6 = 2 + 2 + 2 Points)

Consider two relations R and S with $B(R) = 1,200,000$ and $B(S) = 4,000$. 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.
  $\lceil \frac{B(S)}{M-1} \rceil \cdot [B(R) + min(B(S), (M-1))] = 4 \cdot [1,200,000 + 1000] = 4,804,000$ I/Os
- **MJ**: We can generate sorted runs of size $1,000$ that means we need 2 merge passes for R and one for S. The number of runs in the last phase of sorting is 2 for R and 4 for 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. $5 \cdot B(R) + 3 \cdot B(S) = 5 \cdot 1,200,000 + 3 \cdot 4,000) = 6,012,000$ I/Os.
- **HJ**: After one partition phase the size of the partitions for S (4 pages) is small enough to fit one partition into memory, build an in-memory hash table of each partition of S, and stream a partition of R once probing the hash table. $(2+1) \cdot (B(R) + B(S)) = 3,612,000$ I/Os.

We would use HJ, because it has the lowest I/O requirements.

Part 6 Result Size Estimations (Total: 12 Points)

Consider the table *employee* with attributes *ssn*, *name*, *manager*, *salary*, and *gender*. Attribute *ssn* is the primary key of this relation and attribute *manager* is a foreign key to the relation itself (storing the manager for each employee).

Given are the following statistics:

$$
\begin{aligned}
T(employee) &= 100,000 \\
V(employee, ssn) &= 100,000 \\
V(employee, name) &= 95,000 \\
V(employee, manager) &= 10,000 \\
V(employee, salary) &= 200 \\
V(employee, gender) &= 2
\end{aligned}
$$

Question 6.1 Estimate Result Size (5 Points)

Estimate the number of result tuples for the query $q = \sigma_{manager=12345 \vee gender=female}(employee)$ 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 $P(\neg A) = 1 - P(A)$ and $a \vee b = \neg(\neg A \wedge \neg B)$.

$$
\begin{aligned}
P(manager = 12345 \vee gender = female) &= 1 - ((1 - P(manager = 12345)) \cdot (1 - P(gender = female))) \\
&= 1 - ((1 - \frac{1}{V(employee, manager)}) \cdot (1 - \frac{1}{V(employee, gender)})) \\
&= 1 - ((1 - \frac{1}{10,000}) \cdot (1 - \frac{1}{2})) = 1 - (\frac{9999}{10,000} \cdot \frac{1}{2}) \approx 0.5
\end{aligned}
$$

$$
T(q) = T(employee) * P(manager = 12345 \vee gender = female) = \frac{100,000}{2} \approx 50,000
$$

Question 6.2 Estimate Result Size (7 Points)

Estimate the number of result tuples for the query $q = \sigma_{y.manager=Peter}(\rho_x(employee) \bowtie_{x.manager=y.ssn} \rho_y(employee))$ using the first assumption presented in class.

Solution

We first need to estimate the number of expected join results and the number of distinct values in attribute $y.manager$ in the join result:

$$q_1 = \rho_x(employee) \bowtie_{x.manager=y.ssn} \rho_y(employee)$$

$$T(q_1) = \frac{T(employee)^2}{max(V(employee, manager), V(employee, ssn))} = \frac{100,000 \cdot 100,000}{100,000} = 100,000$$

According to the assumption from class: $V(q_1, y.manager) = 10,000)$. Thus,

$$T(q) = \frac{T(q_1)}{V(q_1, manager)} = \frac{100,000}{10,000} = 10$$

Part 7 Schedules (Total: 15 Points)

Question 7.1 Schedule Classes (15 = 5 + 5 + 5 Points)

Indicate which of the following schedules belong to which class. 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

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

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_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