CS150: Database and Data Mining
Final Exam Solution

December 21, 2022

I SQL and Relational Algebra [14 points]

You want to use your SQL skills to infer what is happening during the Battle of Hogwarts, and you could assume you have the following tables:

CREATE TABLE Wizards(wizid integer, name text, house text, evil boolean, PRIMARY KEY(wizid));

CREATE TABLE Spells(sid integer, name text, offensive boolean, PRIMARY KEY (sid));

CREATE TABLE Attacks(attackid integer, attacker integer, attacked integer, spell integer,
PRIMARY KEY (attackid), FOREIGN KEY(spell) REFERENCES Spells,
FOREIGN KEY(attacker) REFERENCES Wizards, FOREIGN KEY(attacked) REFERENCES Wizards);

Note: For all of the following questions, you do not need any Harry Potter knowledge. Any understanding of Wizards or Spells will not be helpful.

1. (2 points) Select all of the following queries that return the name of each wizard who has been an attacker more than 3 times. Do not assume that names are unique.

   (a) SELECT name FROM Wizards, Attacks
   WHERE wizid = attacker
   GROUP BY attacker, name
   HAVING COUNT(*) >3;

   (b) SELECT name FROM Wizards, Attacks
   WHERE wizid = attacker
   GROUP BY name
   HAVING COUNT(*) >3;

   (c) SELECT name FROM
   (SELECT name FROM Wizards, Attacks
   WHERE wizid = attacker) AS a
   GROUP BY attacker,name
   HAVING COUNT(*) >3
   ORDER BY COUNT(name);

2. (2 points) Select all of the following queries that select the names of wizards (A) that another individual wizard (B) attacked twice, where the attacker (B) used two different spells. There should be no duplicates in this list. Do not assume that names are unique.

   (a) SELECT w1.name AS A
   FROM Wizards w1, Wizards w2, Attacks a1, Attacks a2

WHERE w1.wizid = a1.attacked AND a1.spell != a2.spell
AND w2.wizid = a2.attacked AND a1.attacked = a2.attacked;

(b) SELECT DISTINCT w1. name AS A
FROM Wizards w1, Attacks a1, Attacks a2
WHERE w1.wizid = a1.attacked AND a1.spell != a2.spell
AND w1.wizid = a2.attacked AND a1.attacked = a2.attacked;

(c) SELECT DISTINCT w1.name AS A
FROM Wizards w1, Wizards w2, Attacks a1, Attacks a2
WHERE a1.attacker = a2.attacker AND w1.wizid = a1.attacked
AND a1.spell != a2.spell AND w2.wizid = a2.attacked AND a1.attacked = a2.attacked;

3. (2 points) Select all of the following queries that return the name of all of the spells that have never been used in an attack.

(a) SELECT name
FROM Spells
WHERE sid NOT IN
(SELECT spell FROM Attacks);

(b) SELECT name
FROM Spells
WHERE NOT EXISTS
(SELECT * FROM Attacks WHERE spell = sid);

(c) SELECT Spells.name
FROM Spells, Wizards, Attacks
WHERE wizid = attacker and spell = sid;

Fill in the following blanks to create a query that returns true if more evil wizards cast offensive spells, and returns false if more good wizards cast offensive spells.
Note: there could be more than one correct answer(s).

WITH countEevilGoodSpells(evil, numOffensive) AS
(SELECT evil, count(*)
FROM attacks, wizards, spells
WHERE ___1___ and spell = sid
AND offensive = ___2___
GROUP BY ___3___)
SELECT evil
FROM countEevilGoodSpells
WHERE numOffensive ≥ ALL
(SELECT numOffensive
FROM ___4___);

1. (1 points) Fill in the blank labeled (1).
   (a) wizid=attackid
   (b) wizid=attacked
   (c) wizid=attacker
2. (1 points) Fill in the blank labeled (2).
   (a) True

(b) False

3. (1 points) Fill in the blank labeled (3).

   (a) evil

   (b) spell

   (c) attacker

4. (1 points) Fill in the blank labeled (4).

   (a) Wizards

   (b) countEevilGoodSpells

   (c) Attacks

5. (2 points) Select all of the following answers that return the wizid of all of the wizards who have not attacked anybody.

   (a) $\pi_{\text{wizid}}\ (Attacks - Wizards)$

   (b) $\pi_{wizid}\ (\pi_{wizid}(Wizards) - \pi_{attacker}(Attacks))$

   (c) $\pi_{\text{wizid}}\ (Wizards) - \pi_{\text{attacker}}\ (\ Spells \bowtie_{\text{sid} = spell}\ Attacks \bowtie_{\text{attacker} = wizid}\ Wizards)$

   (d) $\pi_{\text{wizid, name}}\ (Wizards) - \pi_{\text{attacker}}(Attacks)$

6. (2 points) Select all of the following answers that return the wizid of all of the wizards who were attacked by Wizards whose house is Gryffindor.

   (a) $\pi_{\text{attacked}}\ (\ Spells \bowtie_{\text{sid} = spell}\ Attacks \bowtie_{\text{attacker} = wizid}\ \sigma_{\text{house} = 'Gryffindor'}(Wizards\ ))$

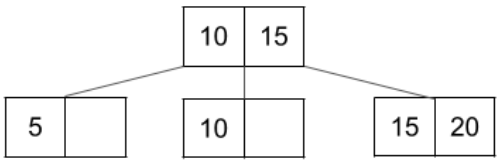
   (b) $\pi_{\text{attacked}}\ (\ Spells \bowtie_{\text{sid} = spell}\ Attacks \bowtie_{\text{attacker} = wizid}\ \sigma_{\text{house} = Gryffindor'}\ (\pi_{\text{wizid}}\ ((\ Wizards\ )))$

   (c) $\pi_{\text{wizid}}\ (Wizards) - \pi_{\text{attacked}}\ (\ Spells \bowtie_{\text{sid} = spell}\ Attacks \bowtie_{\text{attacked} = wizid}\ \sigma_{\text{house} != 'Gryffindor'}(Wizards\ ))$

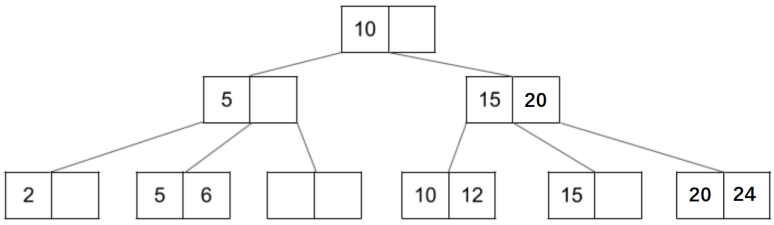
   (d) $\pi_{\text{attacked}}\ (\ Attacks \bowtie_{\text{attacker = wizid}}\ (\sigma_{\text{house} = 'Gryffindor'}(Wizards) \bowtie_{\text{sid = spell}}\ Spells\ ))$

II B+ Trees [10 points]

Given the following (degree d = 1) B+ tree:



(a) [6 points] Draw what the tree looks like after adding 2, 6, and 12 in that order.



For the following questions, consider the tree directly after 2, 6, and 12 have been added.

(b) [2 points] What is the maximum number of inserts we can do without changing the height of the tree?
Answer: 10 inserts.
The maximum number of keys for a fixed height h is given by $2d \cdot (2d + 1)h$. We have d = 1 from the question, and h = 2 (we must remember h is the number of non-root layers). Plugging these numbers in gives us $2 * 32 = 18$.
Now, we already have 8 keys in the tree, so we can insert $18 - 8 = 10$ more.

(c) [2 points] What is the minimum number of inserts we can do that will change the height of the tree?
Answer: 3 inserts (e.g. 25, 26, 27). The idea is to repeatedly insert into the fullest nodes; this is hopefully apparent from understanding how and when B-trees split.

III Buffer Management and Relational Algebra [12 points]

1. [8 points] We're given a buffer pool with 4 pages, which is empty to begin with. Answer the following questions given this access pattern:

   A B C D E B A D C A E C

   (a) [4 points] What is the hit rate for LRU? Answer: 4/12
   (b) [4 points] In order of when they were first placed into the buffer pool, what pages remain in the buffer pool after this sequence of accesses? Answer: C E A D

2. [4 points] Consider two relations with the same schema: R(A,B,C) and S(A,B,C). Which one of the following relational algebra expressions is not equivalent to the others?

   (A) $\pi_{R.A}((R \cup S) - S)$
   (B) $\pi_{R.A}(R) - \pi_{R.A}(R \cap S)$
   (C) $\pi_{R.A}(R - S) \cap \pi_{R.A}(R)$
   (D) $\pi_{R.A}(R - S)$

   Answer: B. Consider the counter example where R(A, B) = (1,2), (1,3) and S(A, B) = (1,2). a and c will result in a size 1 set, while b will result in a size 0 set.

IV External Sorting and Hashing [18 points]

- True or False (2 points)

  1. Increasing the number of buffer pages does not affect the number of I/Os performed in Pass 0

     Solution: True. Regardless of the number of buffer pages, every pass of an external sort (including Pass 0) performs the same number of IOs. of an external sort.

  2. Double buffering reduces the time it takes to sort records within a single page.

     Solution: False. Double buffering allows a program to concurrently perform computations and fetch data from disk. Once a page of data is resident in memory, double buffering will not help speed up computation on it.

- Let's Sort and Hash This Out! (16 points)

  Now you're facing the problem of figuring out the steps to sort a table in your DBMS. Assume your table is 100 pages and you have 5 buffer pages in memory. Please fill out the following pass-by-pass guide that details the number and the length of runs created after each pass in External Merge Sort.

  In the **second** blank for each pass, write all unique run lengths in **increasing** order, separated by commas. In the **first** blank for each pass, write the number of runs corresponding to each run length specified in the second blank, separated by commas.

  For example, if there were 3 runs of length 50 created after Pass 0, then write 3 for the first blank and 50 for the second blank on the line for Pass 0. If instead there were 6 runs created in total: 2 runs of length 10, 1 run of length 25, and 3 runs of length 50, then write 2, 1, 3 for the first blank and 10, 25, 50 for the second blank.

  If the algorithm terminates before the last pass, write 0 for all blanks in unused passes.

  1. Fill out the numbers for each blank in the following passes. (5 points)

     - Pass 0: __20__ run(s) of __5__ page(s) are created.
     - Pass 1: __5__ run(s) of __20__ page(s) are created.
     - Pass 2: __1,1__ run(s) of __20,80__ page(s) are created.
     - Pass 3: __1__ run(s) of __100__ page(s) are created.
     - Pass 4: __0__ run(s) of __0__ page(s) are created.

  2. Based on the guide, how many I/Os does the entire External Merge Sort on this table take? (2 points)

     Solution: 800 (Using External Merge Sort formula: $2N\left(1+\left\lceil\log_{B-1}\left\lceil\frac{N}{B}\right\rceil\right\rceil\right) = 2*100*(1+\left\lceil\log_4\left\lceil\frac{100}{5}\right\rceil\right\rceil) = 800$)
     Full credit was also awarded to 760 if the student noticed that the run of 20 created after pass 2 is not merged so it doesn't need to be read or written in that pass, saving 20 + 20 = 40 I/Os.

  Now you also want to hash the table which requires an exhaustive of the hashing process as well. Still assuming table is 100 pages and you have 5 buffer pages in memory, please fill out the following guide which details the number and size of partitions created after each partitioning pass in External Hashing. You could assume uniform hash functions are used in each pass.

In the **first** blank for each pass, write the total number of partitions created after the pass. In the **second** blank, write the number of pages in each of those partitions. If not all partition passes are needed, write 0 for all blanks in unneeded passes.

1. Fill out the numbers for each blank in the following passes. (5 points)

   - Partitioning Pass 1: ___4___ partition(s) of ___25___ page(s) are created.
   - Partitioning Pass 2: ___16___ partition(s) of ___7___ page(s) are created.
   - Partitioning Pass 3: ___64___ partition(s) of ___2___ page(s) are created.
   - Partitioning Pass 4: ___0___ partition(s) of ___0___ page(s) are created.
   - Partitioning Pass 5: ___0___ partition(s) of ___0___ page(s) are created.

2. Based on the guide, how many I/Os does the entire External Hashing process on this table take? Remember that External Hashing consists of both the divide phase and the conquer phase. (4 points)

   Solution: 908
   I/Os: (I/Os are in bold) In the 1st partitioning pass we read 100 pages and write 100 pages (4 partitions of 25 pages each). In the 2nd partitioning pass we read 100 pages and write 112 pages (16 partitions of 7 pages each). In the 3rd partitioning pass we read 112 pages and write 128 pages (64 partitions of 2 pages each). Now, we can build the hash tables which involves reading 128 pages and writing 128 pages. Adding all I/Os results in 908 I/Os.

V Iterators and Joins [6 points]

Consider a case that, $B > 4$ pages worth of buffer space, and relations M and N of size $> B$. Please fill the blanks below with "always", "sometimes", or "never".

(a) [2 points] Block nested loop join is _____ better than page-oriented nested loop join.
Answer: Always. Block nested loop join is page-oriented on steroids, and steroids and always good.

(b) [2 points] Sort-merge join is _____ better than hash-join.
Answer: Sometimes. Hash join is cooler if one relation is really small and one is very large. Sort dominates if you have lots of dumlicated join keys.

(c) [2 points] Hybrid Hash-Join is _____ better than block-nested loops join.
Answer: Sometimes. Nested loops works for non-equijoins and hash does not. For equijoins, hash is often better since it only makes a small number of passes over each relation, whereas block nested-loops still may visit the inner relation many times. If one relation fits in memory, the two algorithm are about equivalent.

VI Query Optimization [10 points]

Assume the following:

- Column values are uniformly distributed and independent from one another
- Use System R defaults (1/10) when selectivity estimation is not possible
- Primary key IDs are sequential, starting from 1
- Our optimizer **does not consider interesting orders**

We have the following schema:

Table Schema	Records	Pages	Indices
CREATE TABLE Student (sid INTEGER PRIMARY KEY, name VARCHAR(32), major VARCHAR(64)), semesters_completed INTEGER)	25,000	500	• Index 1: Clustered(major). There are 130 unique majors • Index 2: Unclustered(semesters completed). There are 11 unique values in the range [0, 10]
CREATE TABLE Application (sid INTEGER REFERENCES Student, cid INTEGER REFERENCES Company, status TEXT, (sid, cid) PRIMARY KEY)	100,000	10,000	• Index 3: Clustered(cid, sid). • Given: status has 10 unique values
CREATE TABLE Company (cid INTEGER PRIMARY KEY, open_roles INTEGER))	500	100	• Index 4: Unclustered(cid) • Index 5: Clustered(open roles). There are 500 unique values in the range [1, 500]

Consider the following query:
SELECT Student.name, Company.open_roles, Application.referral
FROM Student, Application, Company
WHERE Student.sid = Application.sid – (Selectivity 1)
AND Student.semesters_completed > 4 – (Selectivity 2)
AND (Student.major='CS' OR Company.open_roles <= 50) – (Selectivity 3)
ORDER BY Company.open_roles;

1. [4 points] For the following questions, calculate the selectivity of each of the labeled Selectivities above.

   (a) Selectivity 2
   (10 - 4) / (10 - 0 + 1) = 6/11.
   We have 11 unique values, assumed to be equally distributed. Therefore we use the equation for less than or equal to which is (high key - value) / (high key - low key + 1).

   (b) Selectivity 3
   Sel(Student.major='CS') = 1/distinct values = 1/130
   Sel(Company.open_roles <= 50) = (v - low key) / ((high key - low key + 1) + (1 / number distinct)) = (50-1) / (500-1+1) + 1/500 = 1/10
   Selectivity 3 = S(p1) + S(p2) - S(p1)S(p2) = (1/130 + 1/10) - (1/130 * 1/10) = 10/1300 + 130/1300 - 1/1300 = 139/1300.

2. [6 points] For each predicate, which is the first pass of Selinger's algorithm that uses its selectivity to estimate output size? (**Pass 1, 2 or 3**?)

   (a) Selectivity 1:
   ☐ Pass 1 ✓ Pass 2 ☐ Pass 3

   (b) Selectivity 2
   ✓ Pass 1 ☐ Pass 2 ☐ Pass 3

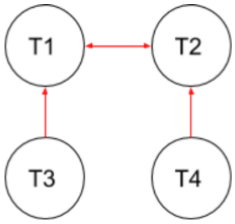
(c) Selectivity 3
☐ Pass 1 ☐ Pass 2 ✓ Pass 3

(b) is pass 1 because they only involve filtering one table.
(a) is pass 2 because they represent a join.
Note that (c), the OR predicate, is over 2 tables that have no associated join predicate, so the selection is postponed along with the cross-product, until after 3-way joins are done.

VII Concurrency Control [10 points]

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

(a) [6 points] Draw the conflict dependency graph for the schedule.



(b) [4 points] Is this schedule conflict serializability? If so, write down all the conflict equivalent serial schedules. If not, explain the reason.
No, there's a cycle between T1 and T2. T1 must come before T2 and T2 before T1. Thus the schedule does not conflict serializable.

VIII Recovery [8 points]

For the following statements, please answer whether it is True or False. (8 points)

1. When a transaction commits, any modified buffer pages must be written to durable storage.
   False. ARIES uses a NO FORCE policy.
2. When aborting a transaction, it may be necessary to modify pages on disk.
   True. ARIES uses a STEAL policy.
3. During recovery, the ARIES protocol may redo aborted transactions.
   True. This is a key feature of ARIES and essential for the correctness of the protocol.
4. The pageLSN contains the LSN of the last operation to modify the page.
   True. This is the definition of the pageLSN.
5. The tail of the log is always flushed after every update operation.
   False. We only require that the tail of the log be flushed on commit. Remember that flushedLSN keeps track of the log tail.
6. A system that uses a FORCE, STEAL policy does not need to undo any operations after a crash.
   False. A system with a STEAL policy needs UNDO logging to ensure atomicity.
7. The recovery manager is responsible for Atomicity and Durability, as defined by the ACID acronym.
   True.
8. During a transaction abort, we redo all updates made by the transaction.
   False.

IX FDs and BCNF [8 points]

1. [4 points] Decompose $R = ABCDEFGH$ into BCNF in the order of the given functional dependencies: $F = \{C \to A, B \to EF, H \to BCG, F \to CD, G \to B\}$.
   Select all the following tables included in the final decomposition.

   (a) AC

   (b) BCDGH

   (c) BCGH

   (d) BG

   (e) DH

   Answer: ADE
   $C \to A$: ABCDEFGH decomposes into BCDEFGH and AC
   $B \to EF$: BCDEFGH decomposes into BCDGH and BEF
   $H \to BCG$: BCDGH decomposes into DH and BCGH
   $F \to CD$: skip
   $G \to B$: BCGH decomposes into CGH and BG
   Final decomposition: AC, BEF, BG, CGH, DH

2. [4 points] Two students are writing a decomposition of tables based on the attributes set $R = ABCDEF$ and the functional dependency set $F = \{C \to D, A \to B, B \to EF, F \to A\}$.
   Another student decides to write a decomposition of tables as $ABDE$ and $ABCF$. Right now, this is not a lossless decomposition. Which of the following changes (applied individually, not combined together) would make this a lossless decomposition?

   (a) Adding $D \to ABCDEF$ to the functional dependency set F.

   (b) Adding $E \to C$ to the functional dependency set F.

   (c) Adding $B \to D$ to the functional dependency set F.

   Answer: BC.
   The intersection is $AB \to ABEF$. To be lossless, either $AB \to ABDE$ or $AB \to ABCF$

X Parallel Querying [4 points]

Suppose we have a table of 1200 rows, perfectly range-partitioned across 2 machines in order.
We just bought the 3th machine for our database, and we want to run parallel sorting using all 4 machines.
The first step in parallel sorting is to re-partition the data across all 3 machines, using range partitioning. (The new machine will get the last range.)
For each of the first 2 machines, **how many rows** will it send across the network during the re-partitioning? (You can assume the new ranges are also perfectly uniform.)
Answer: 200 rows and 400 rows.
The original partitions were 2 ranges of 600 rows each; the new 3 ranges will have 400 rows each.
The first machine held the first 600 rows originally, and now only needs to hold the first 400. It will send the remaining 200 rows over the network (to machine 2).
The second machine held rows 601-1200 initially, but now needs to hold rows 401-800. It will send rows 801-1200 (400 rows) to machine 3.