CS 186/286 Spring 2018 Final

- Do not turn this page until instructed to start the exam.
- You should receive 1 single-page *answer sheet* and a 25-page *exam packet*.
- All answers should be written on the answer sheet. The exam packet will be collected but not graded.
- You have *170 minutes* to complete the final.
- The final has *7 questions*, each with multiple parts.
- The final is worth a total of *108 points*.
- For each question, place only your *final answer* on the answer sheet; do not show work.
- For multiple choice questions, please fill in the bubble or box completely, **do not mark the box with an X or checkmark**.
- Use the blank spaces in your exam for scratch paper.
- You are allowed **three** 8.5" × 11" double-sided page of notes.
- Only standard scientific calculators are allowed.

1 SQL and Relational Algebra (14 points)

Imagine you need to build a site like Piazza and want to maintain its data in a relational database. In this question, we will use the following schema to represent this data.

Users represent the users posting on the site. Assume there are only two types of users, either "students" or "instructors".
Folders represent various topics in which posts are categorized.
Posts represent a post with its summary title.
Postings represent the individual messages in a post. Each message has a unique position in the post.

```
CREATE TABLE Users(
  uid INTEGER PRIMARY KEY,
  name VARCHAR(32),
  dob DATE,
  user_type VARCHAR(10) NOT NULL
)

CREATE TABLE Folders(
  fid integer PRIMARY KEY,
  fname VARCHAR(128)
)

CREATE TABLE Posts(
  pid INTEGER PRIMARY KEY,
  folder INTEGER REFERENCES folders(fid),
  summary VARCHAR(128)
)

CREATE TABLE Postings(
  post INTEGER REFERENCES posts(pid),
  position INTEGER,
  user INTEGER REFERENCES users(uid),
  ptext TEXT,
  PRIMARY KEY(post, position)
)
```

For the questions below, choose the relational algebra expressions that compute the results for the provided English queries. You may choose one or more options. Assume all joins are *natural* joins unless otherwise specified.

1. (2 points) Return names of all users that posted in the "lecture" folder with the summary title "Midterm 2 grades".

   A. $\Pi_{name}(\sigma_{summary=\text{“}Midterm\ 2\ grades\text{''}}(\sigma_{fname=\text{“}lecture\text{”}}(Folders) \bowtie_{Folders.fid=Posts.folder} Posts \bowtie_{Posts.pid=Postings.post} Postings) \bowtie_{Postings.user=User.uid} Users)$

   B. $\Pi_{name}(\sigma_{summary=\text{“}Midterm\ 2\ grades\text{''}}(\Pi_{fid}(\sigma_{fname=\text{“}lecture\text{”}}(Folders)) \bowtie_{Folders.fid=Posts.folder} Posts \bowtie_{Posts.pid=Postings.post} Postings) \bowtie_{Postings.user=User.uid} Users)$

   C. $\Pi_{name}((\sigma_{fname=\text{“}lecture\text{”}}(Folders) \bowtie_{Folders.fid=Posts.folder} Posts \bowtie_{Posts.pid=Postings.post} Postings) \bowtie_{Postings.user=User.uid} \Pi_{uid}(Users))$

   D. $\Pi_{name}(\sigma_{fname=\text{“}lecture\text{”},summary=\text{“}Midterm\ 2\ grades\text{''}}((Folders \bowtie_{Folders.fid=Posts.folder} Posts \bowtie_{Posts.pid=Postings.post} Postings) \bowtie_{Postings.user=User.uid} Users))$

2. (2 points) Return summary titles of all posts with more than one user posting on that post.
   Assume $\rho(R1, Postings), \rho(R2, Postings)$.

   A. $\Pi_{summary}(\Pi_{R1.post}(\sigma_{R1.user \neq R2.user}(R1 \bowtie_{R1.post=R2.post} R2)) \bowtie_{R1.post=Posts.pid} Posts)$

   B. $\Pi_{summary}(\Pi_{R1.post}(\sigma_{R1.user \neq R2.user}(\Pi_{R1.post,R1.user,R2.user}(R1 \times R2))) \bowtie_{R1.post=Posts.pid} Posts)$

   C. $\Pi_{summary}(\Pi_{R1.post}(R1 \times R2 - (R1 \bowtie_{R1.post=R2.post} R2)) \bowtie_{R1.post=Posts.pid} Posts)$

   D. $\Pi_{summary}(\Pi_{R1.post}(\sigma_{R1.post=R2.post \wedge R1.user \neq R2.user}(R1 \times R2)) \bowtie_{R1.post=Posts.pid} Posts)$

3. (2 points) Return the name of the oldest user.
   Assume $\rho(R1, Users), \rho(R2, Users)$

   A. $\Pi_{name}(\sigma_{Users.dob=min(\Pi_{dob}(R1))}(Users))$

   B. $\Pi_{name}(\rho(1 \rightarrow dob, \gamma_{min(dob)}(R1)) \bowtie Users)$

   C. $\Pi_{name}((\Pi_{uid}(Users) - \Pi_{R1.uid}(\sigma_{R1.dob>R2.dob}(R1 \times R2))) \bowtie Users)$

   D. $\Pi_{name}(\Pi_{R1.uid}(\sigma_{R1.dob<R2.dob}(R1 \times R2)) - (\Pi_{uid}(Users)) \bowtie Users)$

For the questions below, choose the SQL queries that compute the results for the provided English queries. You may choose one or more options.

4. (2 points) Return posts with both students and instructor postings.

   A.
```
(SELECT P.pid, P.folder, P.summary
FROM Posts as P INNER JOIN Postings as Ping on P.pid = Ping.post INNER JOIN
    Users as U on Ping.user = U.uid
WHERE U.user_type = 'student')
INTERSECT
(SELECT P.pid, P.folder, P.summary
FROM Posts as P INNER JOIN Postings as Ping on P.pid = Ping.post INNER JOIN
    Users as U on Ping.user = U.uid
WHERE U.user_type = 'instructor');
```

   B.
```
(SELECT P.pid, P.folder, P.summary
FROM Posts as P INNER JOIN Postings as Ping on P.pid = Ping.post INNER JOIN
    Users as U on Ping.user = U.uid
WHERE U.user_type = 'student')
EXCEPT
(SELECT P.pid, P.folder, P.summary
FROM Posts as P INNER JOIN Postings as Ping on P.pid = Ping.post INNER JOIN
    Users as U on Ping.user = U.uid
WHERE U.user_type = 'instructor');
```

   C.
```
(SELECT P.pid, P.folder, P.summary
FROM Posts as P INNER JOIN Postings as Ping on P.pid = Ping.post INNER JOIN
    Users as U on Ping.user = U.uid
WHERE U.user_type = 'student'
OR U.user_type = 'instructor');
```

   D.
```
(SELECT P.pid, P.folder, P.summary
FROM Posts as P INNER JOIN Postings as Ping on P.pid = Ping.post INNER JOIN
    Users as U on Ping.user = U.uid
WHERE U.user_type = 'student'
AND U.user_type = 'instructor');
```

5. (2 points) Find the largest folder(s). This is the folder(s) with the largest number of postings.

A.
```
WITH R AS (
    SELECT P.folder, count(P.pid) as cnt
    FROM Posts as P
    GROUP BY P.folder
)
(SELECT folder
FROM R)
EXCEPT
(SELECT R1.folder
FROM R as R1, R as R2
WHERE R1.cnt < R2.cnt);
```

B.
```
SELECT folder FROM (
  SELECT folder, count(pid)
  FROM Posts
  GROUP BY folder
  HAVING count(pid) = (
  SELECT max(cnt) FROM (
    SELECT folder, count(pid) as cnt
    FROM Posts
    GROUP BY folder
   ) AS t1)
  ) as t2;
```

C.
```
SELECT R.folder FROM
    (SELECT folder, count(pid) as cnt
    FROM Posts
    GROUP BY folder) as R
WHERE R.cnt = (
    SELECT max(R.cnt)
    FROM R
    );
```

D.
```
WITH R as (
  SELECT folder, count(pid) as cnt
  FROM Posts
  GROUP BY folder
)
SELECT R.folder
FROM R
WHERE R.cnt = (
SELECT max(R.cnt)
FROM R
);
```

6. (2 points) How many users post more than once on the same post?

A.
```
SELECT count(distinct user)
FROM Postings
WHERE position > 0
AND user in (
    SELECT user
    FROM Postings
    WHERE position = 0
)
```

B.
```
SELECT count(DISTINCT P1.user)
FROM Postings as P1, Postings as P2
WHERE P1.post = P2.post AND P1.user = P2.user
AND P1.position <> P2.position
```

C.
```
SELECT count(DISTINCT P1.user)
FROM Postings as P1 NATURAL JOIN Postings as P2
WHERE P1.position <> P2.position
```

D.
```
SELECT count(distinct user)
FROM Postings as P1
WHERE exists
    (SELECT user
    FROM Postings as P2
    WHERE P1.post = P2.post AND P1.user = P2.user
    AND P2.position <> P1.position)
```

7. (2 points) Which users (user id) post in more than one folder?

A.
```
WITH USERFOLDER AS
(SELECT user, folder
FROM Posts INNER JOIN Postings on Posts.pid = Postings.post)
SELECT DISTINCT uf1.user
FROM USERFOLDER as uf1, USERFOLDER as uf2
WHERE uf1.user = uf2.user and uf1.post <> uf2.post;
```

B.
```
SELECT DISTINCT P1.user
FROM Postings as P1, Postings as P2
WHERE P1.user = P2.user and P1.post <> P2.post;
```

C.
```
WITH USERFOLDER AS
(SELECT user, folder
FROM Posts INNER JOIN Postings on Posts.pid = Postings.post)
SELECT user FROM
(SELECT user, COUNT(folder) AS cnt
FROM USERFOLDER
GROUP BY user
HAVING cnt > 1);
```

D.
```
WITH USERFOLDER AS
(SELECT user, folder
FROM Posts INNER JOIN Postings on Posts.pid = Postings.post)
SELECT user FROM
(SELECT user, COUNT(DISTINCT folder) AS cnt
FROM USERFOLDER
GROUP BY user
HAVING cnt > 1);
```

2 Files, Indices, and Buffers (18 points)

For this section, remember that **the height of a one level tree is defined to be 0, the height of a tree with two levels is defined to be 1, and so on.**

2.1 True and False

1. (3 points) Fill in the corresponding box on the answer sheet if True.
   A. Flash memory is preferable to disk because flash allows for fine-grained writes.
   B. In random page access patterns, seek time and rotational delays are significant overheads in I/O costs.
   C. The leaves of an ISAM are the only part of the structure that can be modified on an insert.
   D. To keep track of the amount of free space on a page, we use a page directory.
   E. Assuming we only have one copy of a table, we can only build an Alternative 1 clustered index on a single column of that table.
   F. A clustered B+ tree will always perform better than an unclustered B+ tree for equality searches.

2.2 The Lottery

To celebrate the end of the semester, the CS 186 TAs decide to enter a lottery! Each time a TA buys a ticket, a tuple containing the first letter of their name and the number of their ticket is inserted into an unordered file. Note, the each lottery ticket number is unique. At the end of the day, the file contains the following 10 tuples:

Name (Char(1))	Number (Integer)
E	2245
Y	2240
V	2235
E	2243
Y	2244
E	0007
R	2241
C	2242
S	2250
L	2249

Buffer Management

Suppose the Lottery Company stores customer profile information on a page that corresponds to the first letter of their name (e.g. Customer information on Eugene would be stored on page E).

Before the winning number is announced, the Lottery Company wants to go through all tickets that were sold that day and see if their respective purchaser has any criminal records that would disqualify them from participating in the lottery. The company goes through all tickets twice to make sure they don't make any mistakes.

This leads to the following access pattern of pages:

$$E, Y, V, E, Y, E, R, C, S, L, E, Y, V, E, Y, E, R, C, S, L$$

Note that these letters reference pages of a customer profile table and not the records listed in the above user tickets table.

For the following questions, assume we have 4 pages in the buffer pool and assume that pages are unpinned immediately.

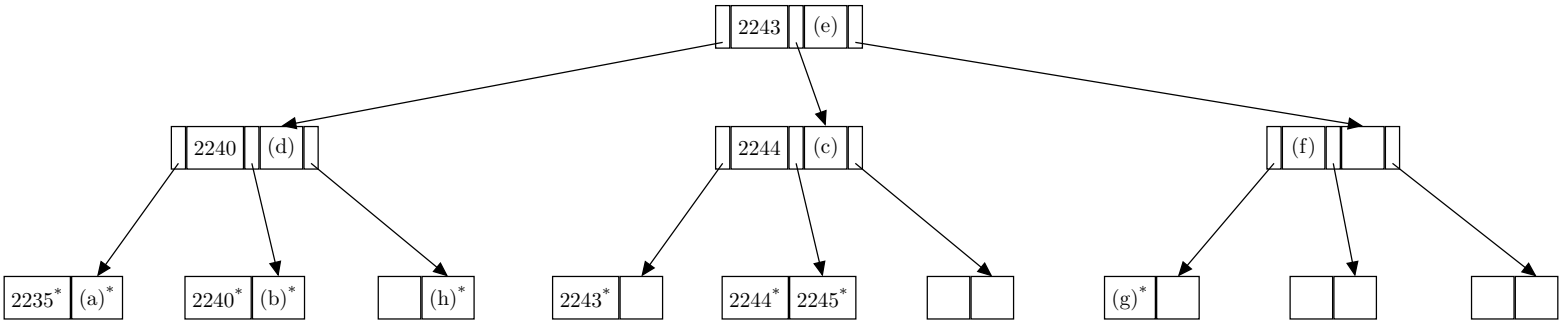
2. (2 points) Using the Most Recently Used replacement policy, which 4 pages remain in the buffer pool (in alphabetical order)? How many cache hits do we have?

3. (2 points) Using the Clock replacement policy, which 4 pages remain in the buffer pool (in alphabetical order)? How many cache hits do we have?

Indices

At the end of the day, the Lottery Company decides upon one winning number from all of the purchased tickets. Instead of traversing the entire file to find the name of the person with the winning ticket, they hire you to build an Alternative 2 B+ tree index on **Number** of order 1. To build this index, they specifically tell you to insert the tuples from the file one at a time in the order they appear (so no bulk loading).

4. (6 points) Draw the resulting B+ tree after all 10 inserts, and write the value of each specified entry on your answer sheet. If the specified entry either does not exist in the final tree or has no value in the final tree, write N/A. The first five records have already been inserted for you. Note that after a split, the new right node gets the larger number of keys or records.



Index Alternatives

Now, consider a different scenario where we have an Alternative 2 Clustered Index and an Alternative 2 Unclustered Index, both with height 3. Assume that 4 records fit on a page for the heap file.

5. (1.5 points) Suppose a range scan returns 6 records. Assuming that all leaf records were on the same leaf of the index, how many IOs does this take for the unclustered Index in the worst case?

6. (1.5 points) Suppose a range scan returns 6 records. Assuming that all leaf records were on the same leaf of the index, how many IOs does this take for the clustered Index in the worst case?

Files

After a while, the business of the Lottery Company grows immensely. On any given day, they expect over 10,000 people to buy tickets for the lottery. At the beginning of each day, the Lottery Company creates a brand new file. Each time a person buys a ticket, a new record (with schema shown earlier) is inserted into the file. At the end of each day, **one** winning number is chosen and the corresponding record is retrieved from the file.

7. (2 points) Given this specification, mark the letters of the parameters **(one from each column)** that will optimize the performance for the tasks the Lottery Company must perform:

Page Format	**File Layout**	**Index**
(a) Fixed length (Packed)	(d) Heap File	(f) Clustered Index on Number
(b) Fixed length (Unpacked)	(e) Sorted File	(g) Unclustered Index on Number
(c) Slotted Page		(h) None

3 Sorting, Hashing, and Parallel Queries (12 points)

For the following questions, we consider the following schema:

```
CREATE TABLE Player(
  pid INTEGER PRIMARY KEY,
  name TEXT
);

CREATE TABLE Relationship(
  pid1 INTEGER REFERENCES Player(pid),
  pid2 INTEGER REFERENCES Player(pid)
);
```

Because there are no uniqueness constraints on `(pid1, pid2)`, a pair of players may appear multiple times in `Relationship`. This is undesirable, and unfortunately, we have already accumulated some duplicates in this table. So, suppose we want to clean up our table by removing duplicates.

In this section, assume that every machine has B=101 pages of memory available. Assume all hash functions perfectly partition their input, and assume a fill factor of 0.5 for all in-memory hash tables. **Do not give your answer as a formula; simplify and provide your answer for each part as a single number.**

First assume we have a **single machine**, and that [`Relationship`] = 200 pages. Also, do not include the I/O cost for writing the output in your answer.

1. (1.5 points) One way to remove duplicates is to first sort the data. What is the cost of sorting `Relationship`?

2. (1.5 points) Alternatively, we can use hashing to remove duplicates. What is the disk I/O cost of removing duplicates using the standard hashing algorithm?

A common optimization is ***partial pre-aggregation***. In this, operators computing decomposable aggregations, like `DISTINCT`, can opportunistically compute the aggregate on part of the data to reduce disk I/O. For example, during a partitioning pass in hashing, we can scan through an output buffer and only append a record if it is unique. As before, we append the output buffer to its partition on disk when full or at the end of the pass.

3. (1.5 points) Suppose every unique row appears exactly 4 times in `Relationship`. What is the disk I/O cost of the partitioning pass of hashing with partial pre-aggregation?

4. (1.5 points) Assume we complete the first partitioning pass with partial pre-aggregation, as above, and maintain the separate partition runs on disk. Using these partitions, what is the disk I/O cost to check if a new record is a duplicate? Ignore the I/O cost of reading the new record, and assume that no data from the partitions is in memory.

Now suppose we have **m=100 machines**. Assume that the data is initially **distributed round robin** across all the machines. Assume further that every unique row in `Relationship` appears exactly 100 times and [`Relationship`] = 100,000 pages. The key we use for parallel hashing is `(pid1, pid2)`.

5. (1.5 points) What is the total amount of data in MBs transferred over the network when doing duplicate elimination with parallel hashing? Assume each page is 1MB.

6. (1.5 points) What is the total disk I/O cost of the partitioning pass of parallel hashing with all optimizations including partial pre-aggregation for duplicate elimination?

7. (3 points) Suppose we have completely de-duplicated and compacted the `Relationship` table, and it is hash partitioned by `(pid1, pid2)` without any skew across machines. Consider the following query:

```
SELECT *
FROM Players, Relationship
WHERE pid2 = pid;
```

   (a) (1.5 points) Suppose [`Player`] = 2,000 pages, and `Player` is hash partitioned by `pid` over all 100 machines. Using the cheapest query plan, how much data (in MBs) must be transferred over the network to compute this query? Assume each page is 1MB.

   (b) (1.5 points) Using the cheapest query plan, what is the total disk I/O cost across all machines needed to compute this query? Do not include the I/O cost for writing the output.

4 Joins and Query Optimization (20 points)

For this section, we consider a modified schema from Section 1. The database has the schema below with the corresponding statistics and indices built on the tables. Remember that the height of a one level tree is defined to be 0, the height of a tree with two levels is defined to be 1, and so on. **Assume that the buffer pool has 10 pages**, and column values are uniformly distributed unless otherwise specified.

Table Schema	Records	Pages	Indices

`CREATE TABLE Users (`
`  uid INTEGER PRIMARY KEY,`
`  name VARCHAR(32),`
`  age INTEGER,`
`  user_type VARCHAR(10) NOT NULL,`
`)`
	10,000	500	Index 0: Clustered alternative 2 B+-tree of height 3 on uid

`CREATE TABLE Posts (`
`  pid INTEGER PRIMARY KEY,`
`  summary VARCHAR(32),`
`  post_time TIMESTAMP,`
`  type VARCHAR(10)`
`)`
	30,000	300	• Index 1: Clustered alternative 2 B+ tree of height 2 on post_time
			• Index 2: Unclustered alternative 2 B+ tree of height 1 on type

`CREATE TABLE Postings (`
`  post INTEGER REFERENCES posts(pid),`
`  position INTEGER,`
`  user INTEGER REFERENCES users(uid),`
`  ptext TEXT,`
`  PRIMARY KEY(post, position)`
`)`
	100,000	1,000	Index 3: Unclustered alternative 2 B+-tree index of height 1 on position

Assume for Q1-3 that traversing the index from root to leaf and scanning the leaf pages on Index 0 incurs a cost of 50 I/Os.

1. (2 points) What is the best cost of a sort-merge join between Postings and Users on uid? Use any optimizations from lecture, if possible, and assume we stream the output.

2. (2 points) What is the best cost of a hash join between Postings and Users on uid? Assume that you have a perfect hash function and fill factor of 1.0. Use the optimization from lecture if possible. Assume we stream the output.

3. (2 points) Now assume that we have 15 buffers and a fill factor of 1.0. What is the I/O cost for a hybrid hash join between Posts and Postings on pid? Assume we stream the output.

Assume the following histogram for the distribution of Users.age. Note, each bucket's range is [min,max), exclusive of the max.

15-18	18-21	21-24	≥ 24
30%	15%	35%	20%

Figure 1: Histogram on Users.age

Consider the following query:

```
SELECT Postings.text, Posts.summary, U.name
FROM (Postings INNER JOIN Posts ON Postings.post = Posts.pid)            ---Predicate 1
      RIGHT OUTER JOIN Users U ON Postings.user = U.uid                  ---Predicate 2
WHERE Posts.type = 'final'                                               ---Predicate 3
      AND U.age < 19                                                     ---Predicate 4
      AND U.name <> 'Ryan'                                               ---Predicate 5
      AND Posts.summary != 'Final␣Exam␣Solutions'                        ---Predicate 6
      AND (Postings.position >= 1,000 OR Posts.post_time <= 2018:05:10)  ---Predicate 7
ORDER BY U.name DESC;
```

4. (1 point) What selectivity would a System R-style optimizer estimate for join Predicate 1?
   A. 1/3,000,000,000
   B. 1/100,000
   C. 1/30,000
   D. 1/10,000
   E. 1/3,000

5. (1 point) What is the best estimate for the selectivity of Predicate 4 on the `Users` table?
   A. 40%
   B. 38%
   C. 37.5%
   D. 35%
   E. 33.75%

Suppose we use a System R-style bottom-up optimization algorithm. In the following questions, you may choose more than one option.

6. (1 point) Which predicate(s) would be considered in the **first** pass of optimization? You may select none of them if no predicates apply in this pass.
   A. Predicate 1
   B. Predicate 2
   C. Predicate 3
   D. Predicate 4
   E. Predicate 5
   F. Predicate 6
   G. Predicate 7

7. (1 point) Which predicate(s) would be considered in the **second** pass of optimization? You may select none of them if no predicates apply in this pass.
    A. Predicate 1
    B. Predicate 2
    C. Predicate 3
    D. Predicate 4
    E. Predicate 5
    F. Predicate 6
    G. Predicate 7
8. (1 point) Which of the following additional indices would be the most useful in speeding up the query?
    A. An unclustered alternative 2 index on Users.uid
    B. A clustered alternative 1 index on Users.name
    C. An unclustered alternative 3 index on Users.age

Assume that the following table contains the correct costs for each access method for the Posts relation.

Relation	**Access Method**	**I/O Cost**
`Posts`	Full table scan	300
`Posts`	Index 1 on post_time	325
`Posts`	Index 2 on type	250

9. (1 point) Select all the single table access plans that are considered for the Posts table.
    A. Full table scan
    B. Index 1
    C. Index 2
10. (1 point) Select all the single table access plans for the Posts table that remain for consideration at the start of the second pass.
    A. Full table scan
    B. Index 1
    C. Index 2
11. (1 point) Now consider all the tables. After the first pass, how many single-table access plans have an interesting order?
    A. 0
    B. 1
    C. 2
    D. 3
    E. 4
    F. 5

Assume that we now have an index on Users.name and the following is the System R-style dynamic programming table after Pass 1:

Relation	Interesting Order	Output Size (in Pages)	I/O cost
Users	none	500	500
Users	name	500	750
Users	uid	500	525
Posts	none	100	100
Postings	none	1,000	1,000

Use the given pass 1 results from the table above to solve the pass 2 questions and use your pass 2 solution to solve for pass 3.

12. (1.5 points) Which of the following joins orders may be **considered** in pass 2?
    A. (Users ⋈ Posts) I/O Cost: 10,500 Output Order: none
    B. (Users ⋈ Posts) I/O Cost: 20,000 Output Order: U.name
    C. (Posts ⋈ Users) I/O Cost: 10,750 Output Order: U.uid
    D. (Postings ⋈ Users) I/O Cost: 5,000 Output Order: none
    E. (Users ⋈ Postings) I/O Cost: 10,250 Output Order: U.name
    F. (Users ⋈ Postings) I/O Cost: 20,100 Output Order: U.uid
    G. (Users ⋈ Postings) I/O Cost: 9,000 Output Order: none
    H. (Posts ⋈ Postings) I/O Cost: 10,600 Output Order: none
    I. (Postings ⋈ Posts) I/O Cost: 10,000 Output Order: none

13. (1.5 points) Which of the following join orders would be **selected** in pass 2?
    A. (Users ⋈ Posts) I/O Cost: 10,500 Output Order: none
    B. (Users ⋈ Posts) I/O Cost: 20,000 Output Order: U.name
    C. (Posts ⋈ Users) I/O Cost: 10,750 Output Order: U.uid
    D. (Postings ⋈ Users) I/O Cost: 5,000 Output Order: none
    E. (Users ⋈ Postings) I/O Cost: 10,250 Output Order: U.name
    F. (Users ⋈ Postings) I/O Cost: 20,100 Output Order: U.uid
    G. (Users ⋈ Postings) I/O Cost: 9,000 Output Order: none
    H. (Posts ⋈ Postings) I/O Cost: 10,600 Output Order: none
    I. (Postings ⋈ Posts) I/O Cost: 10,000 Output Order: none

14. (1.5 points) Which of the following join orders may be **considered** in pass 3?
    A. (Users ⋈ (Postings ⋈ Posts)) I/O Cost: 60,300 Output Order: none
    B. ((Users ⋈ Postings) ⋈ Posts) I/O Cost: 70,100 Output Order: U.name
    C. ((Postings ⋈ Posts) ⋈ Users) I/O Cost: 130,200 Output Order: none
    D. (Posts ⋈ (Users ⋈ Postings)) I/O Cost: 90,250 Output Order: none
    E. ((Postings ⋈ Users) ⋈ Posts) I/O Cost: 70,000 Output Order: none
    F. (Posts ⋈ (Users ⋈ Postings)) I/O Cost: 60,750 Output Order: U.name
    G. ((Users ⋈ Posts) ⋈ Postings) I/O Cost: 50,100 Output Order: U.uid
    H. ((Posts ⋈ Users) ⋈ Postings)) I/O Cost: 50,600 Output Order: none
    I. (Postings ⋈ (Posts ⋈ Users)) I/O Cost: 110,700 Output Order: U.uid
    J. (Postings ⋈ (Users ⋈ Posts)) I/O Cost: 80,400 Output Order: U.name

15. (1.5 points) Assume that the final output size is 500 pages. Which of the following subplans would comprise the final query plan?

    A. (Users ⋈ (Postings ⋈ Posts)) I/O Cost: 60,300 Output Order: none

    B. ((Users ⋈ Postings) ⋈ Posts) I/O Cost: 70,100 Output Order: U.name

    C. ((Postings ⋈ Posts) ⋈ Users) I/O Cost: 130,200 Output Order: none

    D. (Posts ⋈ (Users ⋈ Postings)) I/O Cost: 90,250 Output Order: none

    E. ((Postings ⋈ Users) ⋈ Posts) I/O Cost: 70,000 Output Order: none

    F. (Posts ⋈ (Users ⋈ Postings)) I/O Cost: 60,750 Output Order: U.name

    G. ((Users ⋈ Posts) ⋈ Postings) I/O Cost: 50,100 Output Order: U.uid

    H. ((Posts ⋈ Users) ⋈ Postings)) I/O Cost: 50,600 Output Order: none

    I. (Postings ⋈ (Posts ⋈ Users)) I/O Cost: 110,700 Output Order: U.uid

    J. (Postings ⋈ (Users ⋈ Posts)) I/O Cost: 80,400 Output Order: U.name

5 Transactions and Concurrency Control (16 points)

1. (3.5 points) Fill in the corresponding box on the answer sheet if True.
   A. SIX locks cannot be escalated.
   B. IX locks are compatible with X locks.
   C. All conflict serializable schedules guarantee no cascading aborts if they are created using Strict 2PL.
   D. All conflict serializable schedules are also view serializable.
   E. In deadlock avoidance, later transaction timestamps always map to higher transaction priorities.
   F. Multi-granularity locking allows for higher concurrency.
   G. In multi-granularity locking, we should acquire the locks from bottom to top hierarchy (e.g. from record-level to page-level) and release the locks from top to bottom hierarchy (e.g. from page-level to record-level).

2. (4 points) Consider the following schedule. (For each of the questions below, you may mark zero, one or more than one of the choices.)

	T1	T2	T3	T4
1	R(A)			
2		R(A)		
3			R(C)	
4			W(C)	
5		R(B)		
6		W(B)		
7	R(B)			
8				R(B)
9	R(C)			
10				R(C)
11				W(B)
12		COM		
13			COM	
14	COM			
15				COM

   (a) (0.5 points) What transactions is T1 pointing to in the conflict graph for the schedule above?
      A. T1
      B. T2
      C. T3
      D. T4

(b) (0.5 points) What transactions is T2 pointing to in the conflict graph for the schedule above?

A. T1
B. T2
C. T3
D. T4

(c) (0.5 points) What transactions is T3 pointing to in the conflict graph for the schedule above?

A. T1
B. T2
C. T3
D. T4

(d) (0.5 points) What transactions is T4 pointing to in the conflict graph for the schedule above?

A. T1
B. T2
C. T3
D. T4

(e) (1 point) Which of the following locking disciplines could have produced the above schedule?

A. 2 phase locking
B. Strict 2 phase locking

(f) (1 point) Which of the following schedules below are conflict equivalent to the schedule above?

A. T3, T1, T2, T4
B. T2, T3, T1, T4
C. T4, T3, T1, T2
D. T1, T2, T3, T4
E. T3, T2, T1, T4

3. (8.5 points) A Kitchener high school, Huron Heights, is trying to figure out class assignments for its students! Suppose their database has 2 tables, Students and Classes. Students consists of 2 pages, A and B. Classes consists of another 2 pages, C, and D. They have 3 transactions currently running. Consider the following lock table and wait list of requested locks for each object. Wait list requests are ordered from left to right with the leftmost to be granted next. (For each of the questions below, you may mark zero, one or more than one of the choices.)

Object	Granted
Database	T1: IX, T2:IX, T3: IX
Students	T1: IX, T3: IS
Classes	T2: SIX, T3: IS
A	T1: S
B	T1: X
C	T2: X
D	T3: S

Object	Waitlist
Database	
Students	
Classes	T3: SIX, T1: S
A	
B	T3: S
C	
D	

(a) (0.5 points) What transactions is T1 pointing to in the waits-for graph for the table above?

A. T1
B. T2
C. T3

(b) (0.5 points) What transactions is T2 pointing to in the waits-for graph for the table above?

A. T1
B. T2
C. T3

(c) (0.5 points) What transactions is T3 pointing to in the waits-for graph for the table above?

A. T1
B. T2
C. T3

(d) (1 point) True or False? Suppose T3 asks to escalate to an IX lock on Classes and subsequently requests an X lock on page D. Its requests can be immediately granted.

(e) (1 point) Suppose T2 commits in the next time step. After T2 commits, which locks are granted?

A. T3: S(B)
B. T3: SIX(Classes)
C. T1: S(Classes)

(f) (1.5 points) Starting from the original lock table, assume we are using Wait-Die deadlock avoidance and that the transactions were started in the following chronological order: T2, T3, T1. If T2 requests X lock on Students, which of the following happen as a result of deadlock avoidance?

A. T1 dies
B. T2 dies
C. T3 dies
D. T2 waits

(g) (1.5 points) Starting from the original lock table, assume we are using Wound-Wait deadlock avoidance and that the transactions were started in the following chronological order: T2, T3, T1. If T2 requests X lock on Students, which of the following happen as a result of deadlock avoidance?

A. T1 dies
B. T2 dies
C. T3 dies
D. T2 waits

(h) (1 point) Starting from the original lock table, suppose T2 requests an X lock on Classes. Which transaction(s) will be in the waitlist for Classes?

A. T1
B. T2
C. T3

(i) (1 point) Support further that T3 subsequently aborts. Which transaction(s) will remain in the waitlist for Classes?

A. T1
B. T2
C. T3

6 Recovery (18 points)

1. (3.5 points) Fill in the corresponding box on the answer sheet if True.
   A. If we do not allow buffer-pool frames with uncommitted updates to overwrite committed data on DB disk (No Steal), then we do not need UNDO logging.
   B. Write-Ahead Logging (WAL) guarantees that a transactions log records are flushed to disk before the transaction commit.
   C. If PageLSN is greater than the max LSN flushed so far (flushedLSN), we can safely write this page to disk.
   D. In the REDO phase, the PageLSN is used to identify and skip updates that have been already applied.
   E. The recovery REDO process does not force any updates to disk.
   F. If we are forced to flush pages in the dirty page table (DPT) to disk upon checkpointing, then we can skip the analysis phase whenever we have a checkpoint available.
   G. When we perform checkpointing in a distributed database system, nodes have to communicate synchronously to agree on a checkpoint timestamp to ensure correctness.

2. (3 points) Consider the following log. If the **flushedLSN is 50**, under WAL, which of the following scenario(s) are possible?

LSN	**Record**	**prevLSN**
10	UPDATE: T1 writes P1	null
20	UPDATE: T2 writes P2	null
30	Begin Checkpoint	-
40	End Checkpoint	-
50	UPDATE: T2 writes P2	20
60	UPDATE: T1 writes P3	10

   A. No dirty pages have been flushed to disk.
   B. The page updated at LSN 50 has been flushed to disk but the page updated at LSN 10 has not.
   C. The page updated at LSN 50 has been flushed to disk but the page updated at LSN 20 has not.
   D. All dirty pages have been flushed to disk.

Consider the following log, recovered after a crash. T1, T2, and T3 are the only transactions. All pages were flushed to disk at and including LSN 50, so the log record has been truncated to start at LSN 60. For the following questions, assume we have not flushed any pages to disk after LSN 50, that is, all pages on disk have pageLSN < 60.

LSN	**Record**	**prevLSN**
...	...	...
60	UPDATE: T1 writes P2	null
70	UPDATE: T2 writes P3	40
80	Begin Checkpoint	-
90	End Checkpoint	-
100	UPDATE: T2 writes P1	70
110	UPDATE: T1 writes P3	60
120	UPDATE: T3 writes P1	null
130	T1 ABORT	110
140	CLR: T1 LSN 110, undoNextLSN: 60	130
150	T2 COMMIT	100
	CRASH	

3. (4 points) Fill out the transaction table (**order by TID**) and dirty page table (**order by PID**) as recorded in the end checkpoint record. You may not need all rows.

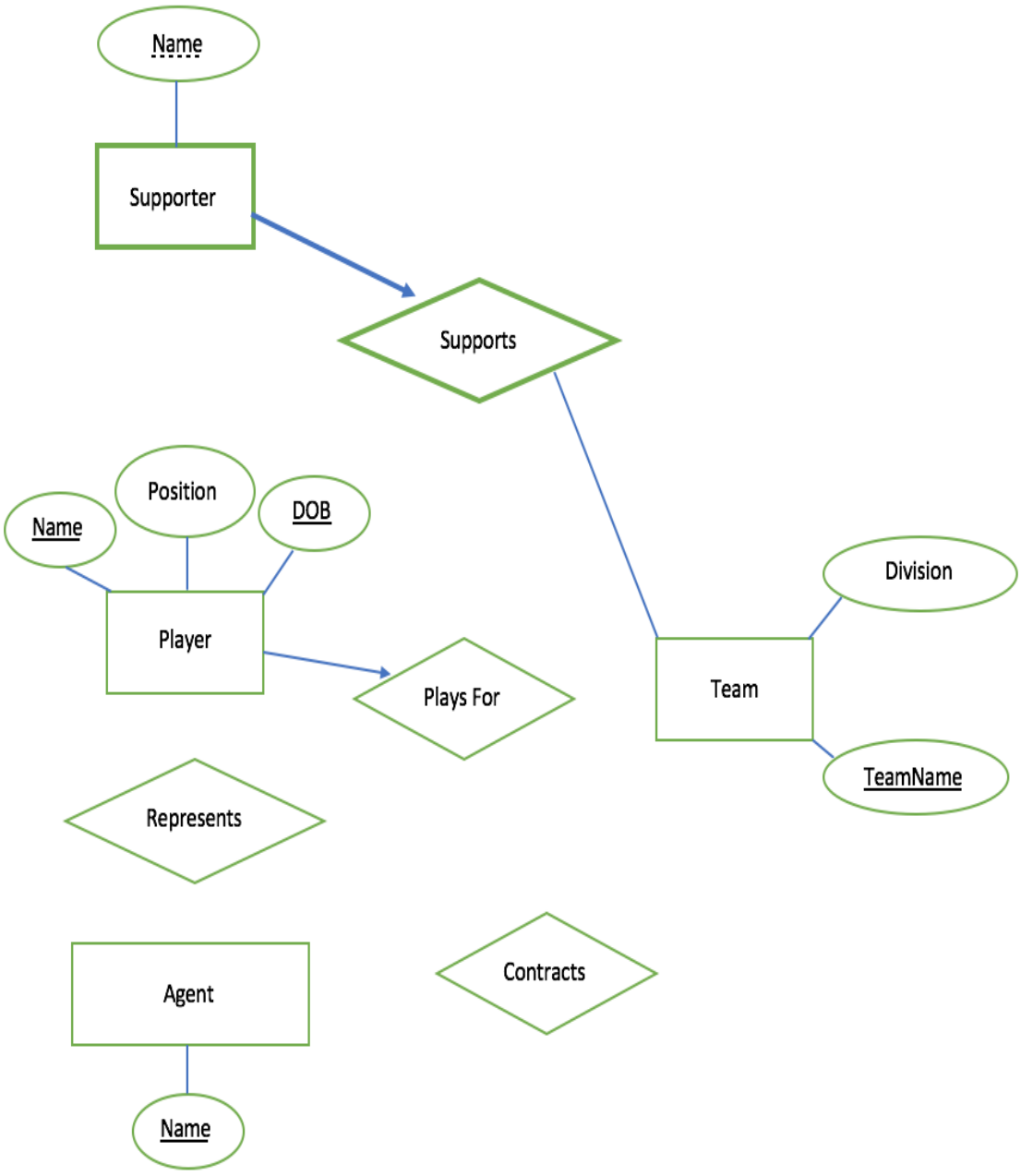
4. (4 points) Now, fill out the transaction table (**order by TID**) and dirty page table (**order by PID**) for the same log after the analysis phase. You may not need all rows.

5. (2 points) After the UNDO phase is complete, which of the record(s) are added to log?
   A. a CLR for T1's update at LSN 60
   B. a CLR for T2's update at LSN 100
   C. a CLR for T2's update at LSN 70
   D. a CLR for T3's update at LSN 120

6. (1.5 points) Consider a slightly different scenario, in which the records at LSN 90 and 100 are in opposite order (that is, the UPDATE occurs before the End Checkpoint). Of the following, choose the most appropriate answer.
   A. The dirty page table stored in the checkpoint must be different than in the original scenario.
   B. The dirty page table stored in the checkpoint could be different than in the original scenario.
   C. The dirty page table stored in the checkpoint must be the same as in the original scenario.

7 Database Design (10 points)

In this question, we will choose to model elements of a baseball league using ER-Diagrams. We have 4 entities: supporters, teams, players, and agents. Refer to the diagram below for the following question. Some elements of the ER-diagram are missing; we will ask you to fill them in.



1. (1 point) Each team must have at least 30 players on its roster. Which edge should we draw to connect the Team entity set with the Plays For relationship set?
    A. Thin line
    B. Thick line
    C. Thin Arrow
    D. Thick Arrow

2. (2 points) Each player must have exactly one agent, and **each agent can represent as many players as he or she wants to, or none at all**. Select all choices indicating the appropriate edges we should draw to indicate this relationship.
    A. Thin Line (Agent to Represents)
    B. Thick Line (Agent to Represents)
    C. Thick Arrow (Agent to Represents)
    D. Thin Arrow (Agent to Represents)
    E. Thin Line (Player to Represents)
    F. Thick Line (Player to Represents)
    G. Thick Arrow (Player to Represents)
    H. Thin Arrow (Player to Represents)

3. (1 point) True or False: In the diagram above, a supporter may **not** support multiple baseball teams.
    A. True
    B. False

4. (2 points) Select the proper SQL table corresponding to the **Supporter** Entity set:
    A. `CREATE TABLE Supporter(Name CHAR(20), TeamName CHAR(20) NOT NULL, PRIMARY KEY (Name, TeamName), FOREIGN KEY (TeamName) REFERENCES TEAM ON DELETE CASCADE);`
    B. `CREATE TABLE Supporter(Name CHAR(20), TeamName CHAR(20) REFERENCES TEAM, PRIMARY KEY (Name, TeamName));`
    C. `CREATE TABLE Supporter(Name CHAR(20), TeamName CHAR(20) REFERENCES TEAM NOT NULL, PRIMARY KEY (Name));`
    D. `CREATE TABLE Supporter(Name CHAR(20), TeamName CHAR(20) REFERENCES TEAM ON DELETE CASCADE, PRIMARY KEY (Name, TeamName));`

5. (2 points) Select the SQL tables (there may be more than one appropriate one) that can represent the **PlaysFor** relationship for Players:
    A. `CREATE TABLE PLAYER_PLAYSFOR (Name CHAR(20), Position CHAR(20), DOB DATE, TeamName CHAR(20), PRIMARY KEY (Name, DOB), FOREIGN KEY (TeamName) REFERENCES Team);`
    B. `CREATE TABLE PLAYSFOR (Name CHAR(20), DOB DATE, Division CHAR(10), TeamName CHAR(20), PRIMARY KEY (Name, TeamName));`
    C. `CREATE TABLE PLAYSFOR (Name CHAR(20), DOB DATE, TeamName CHAR(20), PRIMARY KEY (Name, DOB), FOREIGN KEY (Name, DOB) REFERENCES Player, FOREIGN KEY (TeamName) REFERENCES Team);`
    D. `CREATE TABLE PLAYSFOR (Name CHAR(20), DOB DATE, TeamName CHAR(20), PRIMARY KEY (Name, DOB, TeamName), FOREIGN KEY (Name, DOB) REFERENCES Player;`
    E. `CREATE TABLE PLAYSFOR (Name CHAR(20), DOB DATE, TeamName CHAR(20), PRIMARY KEY (Name, DOB, TeamName);`

6. (1 point) Let us now add in another set: the relation **Contract**, which involves a player, his agent, and his team. Let us model this using a ternary relationship. Assuming that each player can have at most one contract, but that the other assumptions from previous parts hold, what should be the edge connecting the **Player** entity and **Contracts** relation?

    A. Thin Line (Player to Contracts)

    B. Thick Line (Player to Contracts)

    C. Thick Arrow (Player to Contracts)

    D. Thin Arrow (Player to Contracts)

7. (1 point) Assuming that each player can have at most one contract, but that the other assumptions from previous parts hold, what should be the edge connecting the **Agent** entity and **Contracts** relation? Remember that agents can choose to represent as many players as they want, or none at all.

    A. Thin Line (Agent to Contracts)

    B. Thick Line (Agent to Contracts)

    C. Thick Arrow (Agent to Contracts)

    D. Thin Arrow (Agent to Contracts)