# CS W186 Fall 2018 Final

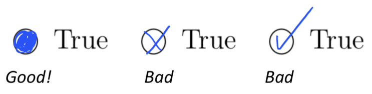
**Do not turn this page until instructed to start the exam.**

## Contents:

- You should receive one *double-sided answer sheet* and a 33-page *exam packet.*
- The final has *10 questions*, each with multiple parts.
- The final is worth a total of *126.5 points.*

## Taking the exam:

- You have *170 minutes* to complete the final.
- All answers should be written on the answer sheet. The exam packet will be collected but not graded.
- 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* as shown on the left below. *Do not mark the box with an X or checkmark.*



- Use the blank spaces in your exam for scratch paper.

## Aids:

- You are allowed **two** 8.5" × 11" double-sided pages of notes.
- The **only** electronic devices allowed are basic scientific calculators with simple numeric readout. No graphing calculators, tablets, cellphones, smartwatches, laptops, etc. are allowed.

## 1 FDs & BCNF (12 points)

1. (5 points) Decompose R = ABCDEFGH into BCNF in the order of the given functional dependencies: F = {C → A, B → EF, H → BCG, F → CD, G → B}.

   Which of the following tables are included in the final decomposition?

   **A. AC**

   B. BCDGH

   C. BCGH

   **D. BG**

   **E. DH**

   **Solution:** C → A: ABCDEFGH decomposes into BCDEFGH and **AC**

   B → EF: BCDEFGH decomposes into BCDGH and **BEF**

   H → BCG: BCDGH decomposes into **DH** and BCGH

   F → CD: Skip

   G → B: BCGH decomposes into **CGH** and **BG**

   Final decomposition: AC, BEF, BG, CGH, DH

2. (3 points) Two students are writing a decomposition of tables based on the attribute set R = ABCDEF and the functional dependency set F = {C → D, A → B, B → EF, F → A}.

   A Stanford 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 → ABCDEF to the functional dependency set F

   **B. Adding E → C to the functional dependency set F**

   **C. Adding B → D to the functional dependency set F**

   **Solution:** Intersection is AB: AB → ABEF. To be lossless, either AB → ABDE or AB → ABCF.

3. (4 points) Please mark the functional dependencies that are consistent with the following table.

| A | B | C | D |
|---|---|---|---|
| 1 | 3 | 6 | 2 |
| 2 | 4 | 1 | 1 |
| 1 | 3 | 5 | 2 |
| 2 | 2 | 1 | 2 |

   A. A →D

   **B. AB → D**

C. A $\rightarrow$ CD

**D. ABC $\rightarrow$ D**

**Solution:** A $\rightarrow$ D: False because when A is 2, D could be either 1 or 2

AB $\rightarrow$ D: True because when AB is (1,3), D can only be 2. When AB is (2, 4), D can only be 1. When AB is (2, 2), D can only be 2.

A $\rightarrow$ CD: False because when A is 1, CD could be either (6, 2) or (5, 2)

ABC $\rightarrow$ D: True because ABC has unique rows in the table. So, you can never have a conflict where one row implies two different things.

## 2 Transactions/2PL (17 points)

1. (3 points) Consider the following schedule:

| $T_1$ | $T_2$ |
|---|---|
| Read(A) | |
| Read(B) | |
| | Read(A) |
| | Read(B) |
| A = A + B | |
| | c = MIN(A, B) |
| | d = MAX(A, B) |
| B = A - B | |
| A = A - B | |
| | A = c |
| | B = d |
| Write(A) | |
| | Write(A) |
| Write(B) | |
| | Write(B) |

Which of the following describe this schedule?

**A. Serializable**

B. View Serializable

C. Conflict Serializable

**Solution:** This schedule is equivalent to the serial schedule $T_1, T_2$ (where A is assigned the smaller value, and B is assigned the larger value).

It is not view serializable (and therefore not conflict serializable). Written another way, the schedule looks like:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $T_1$ | R(A) | R(B) | | | W(A) | | W(B) | |
| $T_2$ | | | R(A) | R(B) | | W(A) | | W(B) |

This cannot have the same initial reads and same winning writes as a serial schedule: the only serial schedule with the same winning writes is $T_1, T_2$, but then $T_2$ would not read the initial value of A and B.

2. (2.5 points) For each of the following, mark either true or false.

A. Two Phase Locking must be used, or we cannot guarantee conflict serializability.

B. Two Phase Locking must be used, or we cannot guarantee serializability.

C. Two Phase Locking ensures that we do not have cascading aborts.

D. Strict Two Phase Locking ensures that we do not have deadlocks.

E. Strict Two Phase Locking requires that transactions acquire locks only at the start of the transaction, and release all locks only at the end of the transaction.

**Solution:**

The first choice is false: 2PL is not required to guarantee conflict serializability, it is just one way to enforce it.

The second choice is false, for a similar reason to the first choice.

The third choice is false: Strict 2PL is required to avoid cascading aborts.

The fourth choice is false: Strict 2PL does not help us avoid deadlocks: if $T_1$ wants X(A), X(B), and $T_2$ wants X(B), X(A), we can get into a deadlock if $T_1$ acquires X(A) and $T_2$ acquires X(B).

The fifth choice is false: there is no requirement for all locks to be acquired at the very start.

3. (2 points) In our new database, we implement lock promotion similarly to Homework 5: if we need to block the transaction, we place the request at the front of the queue. Unlike Homework 5, however, we release the lock a transaction originally held before blocking the transaction and adding the promotion request to the queue. We wish to test our implementation to make sure that the request was indeed placed at the front, rather than the back, of the queue, and write the following test:

```
Transaction t1, t2, t3;
t1 requests __(1)__ lock on database
t2 requests __(2)__ lock on database
t3 requests __(3)__ lock on database

t2 requests a promotion to __(4)__
check that t2 is blocked

t1 releases lock on database
check that t2 has a __(4)__ lock on database
```

Our new database does not support multigranularity locking, so transactions can only have an S or X lock. What lock types do we need for this test to properly test that our implementation indeed places the request at the front of the queue (in other words, the test should pass if and only if we placed the promotion request at the front of the queue)? If there are multiple answers, choose any one.

| (1) | (2) | (3) | (4) |
|---|---|---|---|
| [✓ A. S] | [✓ A. S] | A. S | A. S |
| B. X | B. X | [✓ B. X] | [✓ B. X] |

**Solution:** Our answer must satisfy the following conditions:

1. (1) must be compatible with (2) (or else we block t2 on the first acquire and cannot test the promotion)
2. (3) must not be compatible with (1) (or else t3 is not blocked, and the queue has size 1: we cannot test queue order)
3. (4) must be stronger than (2) (or else this is not a promotion)

The first condition requires (1) and (2) both be S.

The second condition requires (3) be X.

The third condition requires (4) be X (since (2) must be S).

Therefore, the only correct answer is S/S/X/X.

4. (1.5 points) In this question, assume that the levels of granularity we have are: Database, Table, Page, Record. For each of the following, mark either true or false.
   A. To write to a page, we must have an X lock the page, the table, and the database.
   B. A transaction that has only one lock—an IX lock on the database—has enough to request an X lock on a page.
   C. There is no need to check for a SIX and IX conflict at the table level, because the conflict can be resolved at a lower level (page or record) when one transaction requests an X lock.

**Solution:** The first choice is false: we must have an IX, not X, lock on the table and database.

The second choice is false: we must also have an IX lock on the table.

The third choice is false: the conflict between the X lock at the lower level and the S part of SIX cannot be checked at a lower level, since the S is implicit.

5. (3 points) Mark all the transactions that are involved in a deadlock. If there is no deadlock, mark No Deadlock.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $T_1$ | X(A) | | | | | | | | | S(D) |
| $T_2$ | | S(A) | | | | | | | | |
| $T_3$ | | | X(B) | | | | | S(C) | X(A) | |
| $T_4$ | | | | S(C) | | | X(A) | | | |
| $T_5$ | | | | | S(D) | X(B) | | | | |

A. $T_1$

B. $T_2$

C. $T_3$

D. $T_4$

E. $T_5$

**F. No Deadlock**

**Solution:** There is no deadlock. The "waits-for" graph has the following edges:

- $T_2$ to $T_1$ (S(A) conflicts with X(A))
- $T_5$ to $T_3$ (X(B) conflicts with X(B))
- $T_4$ to $T_1$ (X(A) conflicts with X(A))
- $T_3$ to $T_1$ (X(A) conflicts with X(A))

This graph has no cycles, so there are no deadlocks.

6. (1 point) Which of the following applies for the "wait-die" deadlock avoidance strategy?

**A. This deadlock avoidance strategy avoids all deadlocks.**

B. This deadlock avoidance strategy never aborts unnecessarily.

**Solution:** The wait-die deadlock avoidance strategy avoids all deadlocks (for a deadlock to form, we must have a cycle of waiting transactions, but here, a transaction can only wait if it has higher priority: the highest priority transaction can never be waited on).

The wait-die deadlock avoidance strategy can abort unnecessarily: if $T_i$ has lower priority than $T_j$ and wants a lock that $T_j$ has, it will abort even if $T_j$ never waits on another transaction.

7. (1 point) Which of the following applies for the "wound-wait" deadlock avoidance strategy?

**A. This deadlock avoidance strategy avoids all deadlocks.**

B. This deadlock avoidance strategy never aborts unnecessarily.

**Solution:** The wound-wait deadlock avoidance strategy avoids all deadlocks (for a deadlock to form, we must have a cycle of waiting transactions, but here, a transaction can only wait if it has lower priority: the lowest priority transaction can never be waited on).

The wound-wait deadlock avoidance strategy can abort unnecessarily: if $T_i$ has higher priority than $T_j$ and wants a lock that $T_j$ has, $T_j$ will abort even if $T_j$ never waits on another transaction.

8. (3 points) Consider a "wait-wait-die" deadlock avoidance strategy, where when $T_i$ requests a lock that $T_j$ holds, it follows the following protocol:

   - if $T_j$ has higher priority, $T_i$ waits,
   - else if no transaction is waiting on $T_i$, $T_i$ waits,
   - else, $T_i$ aborts.

   Which of the following applies for the "wait-wait-die" deadlock avoidance strategy?

   A. This deadlock avoidance strategy avoids all deadlocks.

   B. This deadlock avoidance strategy never aborts unnecessarily.

   **C. This deadlock avoidance strategy aborts less frequently than the wait-die strategy.**

**Solution:** The "wait-wait-die" deadlock avoidance strategy does not avoid all deadlocks. If we have $T_1 > T_2$ (i.e. $T_1$ has higher priority, $T_2$ has lower), and we request locks in the following order:

- $T_1$: X(A)
- $T_2$: X(B)
- $T_1$: S(B) (waits on $T_2$, allowed because no one is waiting on $T_1$)
- $T_2$: S(A) (waits on $T_1$, allowed because $T_1$ has higher priority)

then $T_1$ and $T_2$ wait on each other.

This deadlock avoidance strategy can abort unnecessarily: if $T_i$ has lower priority than $T_j$ and wants a lock that $T_j$ has, and $T_k$ is waiting on $T_i$, $T_i$ will abort even if $T_j$ never waits on another transaction.

This deadlock avoidance strategy does abort less frequently than the wait-die strategy: the wait-die strategy aborts whenever the wait-wait-die strategy aborts, while additionally aborting when no transaction waits on $T_i$ and $T_i$ has lower priority.

## 3 Recovery (14 points)

1. (4 points) For each of the following four questions, mark True or False.
   - A. A NO STEAL policy is useful for achieving atomicity without REDO logging.
   - **B. A FORCE policy is useful for achieving durability without REDO logging.**
   - C. An implementation of the REDO phase that only checks pageLSN $\geq$ LSN and not the other 2 criteria is an incorrect implementation (here correctness means that an operation is redone in this implementation if and only if it is redone in the original implementation from lecture).
   - D. Assume that a system only allows at most $n$ active transactions. In the worst case, there is no ACID-preserving mechanism to bound the number of operations undone in the undo phase by some constant $k$.

**Solution:** a. False - NO STEAL provides atomicity without UNDO logging
b. True
c. False - this is still correct, just inefficient because we do unnecessary IOs
d. False - we can force the system to abort transactions that are longer than $k/n$ operations long.

Consider the following log recovered after a crash. Assume the buffer pool flushes to disk all pages referenced up to (and including) LSN 60, and does not flush any pages after that.

| LSN | Record | prevLSN |
|---|---|---|
| 70 | UPDATE T2 W(P3) | null |
| 80 | UPDATE T1 W(P1) | null |
| 90 | Begin Checkpoint | — |
| 100 | UPDATE T3 W(P2) | null |
| 110 | UPDATE T1 W(P2) | 80 |
| 120 | End Checkpoint | — |
| 130 | COMMIT T2 | 70 |
| 140 | COMMIT T3 | 100 |
| 150 | END T2 | 130 |
| 160 | UPDATE T4 W(P1) | null |
| 170 | UPDATE T1 W(P1) | 110 |
| 180 | ABORT T1 | 170 |

2. (5.5 points) Fill out what the DPT and Xact table look like at the beginning of the REDO phase (after ANALYSIS ends). Please write **None** for any fields that should not be in the tables. For the Xact table Status, the possible options are: running, committing, aborting.

| **PID** | **recLSN** |
|---|---|
| 1 | (a) |
| 2 | (b) |
| 3 | (c) |

| **TID** | **lastLSN** | **Status** |
|---|---|---|
| 1 | (d) | (e) |
| 2 | (f) | (g) |
| 3 | (h) | (i) |
| 4 | (j) | (k) |

**Solution:**

| PID | recLSN |
|---|---|
| 1 | 80 |
| 2 | 100 |
| 3 | 70 |

| TID | lastLSN | Status |
|---|---|---|
| 1 | 180 | Aborting |
| 2 | None | None |
| 3 | None | None |
| 4 | 160 | Running |

3. (3 points) For each of the following questions, mark True if the record is added to the log after recovery, and False otherwise.

   A. A CLR for T2's update at LSN 70

   **B. A CLR for T1's update at LSN 80**

   C. A CLR for T3's update at LSN 100

   **D. A CLR for T1's update at LSN 110**

   **E. A CLR for T4's update at LSN 160**

   **F. A CLR for T1's update at LSN 170**

**Solution:** The correct choices are B, D, E, F. Check the solution for the next part to see what the log looks like after recovery.

4. (1.5 points) Mark the box corresponding to the order of transactions (ascending by LSN) in which the CLR records are added. For example, if we had a log that looked like this

| LSN | Record | prevLSN |
|---|---|---|
| 190 | CLR T5 ... | ... |
| 200 | CLR T6 ... | ... |
| 210 | CLR T7 ... | ... |

The ordering would be

T5, T6, T7

A. T1, T1, T1, T4, T3, T2

B. T1, T4, T1, T3, T1, T2

C. T1, T1, T1, T4

**D. T1, T4, T1, T1**

E. T1, T1, T1, T3, T2

F. T1, T1, T3, T1, T2

**Solution:** D is correct. Here is what the log looks like (ignoring ABORT and END records).

| LSN | Record | prevLSN |
|---|---|---|
| 190 | CLR T1 LSN 170, undoNextLSN 110 | 180 |
| 200 | CLR T4 LSN 160, undoNextLSN null | 160 |
| 210 | CLR T1 LSN 110, undoNextLSN 80 | 190 |
| 220 | CLR T1 LSN 80, undoNextLSN null | 210 |
| | | |

## 4 Text Search Incl Ranking (TF-IDF) (12 points)

1. (6 points) For each assertion, fill in the corresponding bubble True or False.

   **A. In the "Bag of Words" model, we might choose to ignore the word "of" because it doesn't contain much information. This is an example of a stop word.**

   B. To allow fast text searches, we build inverted files, which contain maps from documents to words.

   C. The vector space of an IR system is enormous because it has one "dimension" per document in the corpus.

   **D. When calculating distance to find similarity between docs, it is beneficial to normalize the vectors.**

   E. Consider three docs: Doc1 has length 10,000, Doc2 has length 9,900, Doc3 has length 100. If we use Cosine Similarity Distance Metric, Doc1 is very likely to be more similar to Doc2 than to Doc3.

   F. TF-IDF value increases as the number of times a word appears in the document increases.

**Solution:** A, D

Explanation:

B is wrong. Inverted files map words to documents.

C is wrong. Should be one dimension per term.

E is wrong. Length does not matter since we normalize the vectors in this metric, so we don't know which one is more likely to be similar.

F was origially said to be true, but techically it should be false because if every doc contains the word, then the log part is 0 and thus the TF-IDF value is 0 and doesn't increase. Since this is pretty tricky, the majority of people chose true and the staff didn't catch it at first, we've given point for choosing either true or false.

2. (1 point) When doing IR Ranking, what types of words do we favor? **There may be zero, one, or more than one correct answer.**

   **A. repeated words**

   B. long words

   **C. unusual words**

   D. capitalized words

**Solution:** A, C

Explanation:

Directly from lecture that we favor repeated and unusual words.

3. (2 points) Consider 500 documents with docID $1, \ldots, 500$. The following table contains the number of occurrences of the word "database" in each document:

| docID | number of "database" |
|---|---|
| 1 | 4 |
| 2 | 17 |
| 3 | 12 |
| 4 | 7 |
| 5 | 9 |
| 6-500 | 0 |

What is the "DocTermRank" for term = "database" and docID = 2? (Hint: consider TF and IDF). *Assume base-10 for any logarithms.*

**Solution:** 34

Explanation:

$17 \cdot \log_{10}(500/5) = 34$

4. (1 point) We run a text search query and get the top 100 answers. We find that 95 of them are correct. We say that this is a good answer because 95% of them are correct. Which metric of answer quality is this?

    A. Accuracy

    **B. Precision**

    C. Retrieval

    D. Recall

**Solution:** B

Explanation:

Definition of precision.

For the next two problems, consider the following expressions:

A. (# of true positives) / ((# of true positives) + (# of false positives))

B. (# of true positives) / ((# of true positives) + (# of true negatives))

C. (# of true positives) / ((# of true positives) + (# of false negatives))

D. (# of true positives) / (# of true negatives)

5. (1 point) Which of the above 4 expressions above is a definition of precision?

**Solution:** A (from lecture)

6. (1 point) Which of the above 4 expressions above is a definition of recall?

**Solution:** C (from lecture)

## 5 Distributed Transactions with Two Phase Commit (2PC) (10 points)

1. (4 points) For each of the following four questions, mark True or False.

   A. Distributed deadlock occurs only if the waits-for graph at a node forms a cycle.

   B. In 2PC, we need all participant nodes to agree on abort (vote NO) to abort a transaction.

   **C. A participant machine that does not flush its prepare records to the log before responding to the coordinator could violate the durability property.**

   D. In 2PC phase 2, after the coordinator sends commit message to all participants, participants first respond with Ack, and then generate and flush commit record.

**Solution:**

A. False: Distributed deadlock occurs whenever the union of waits-for graphs across different nodes forms a cycle.

B. False: Unanimity is only required for commit.

C. True

D. False: Participants respond with Ack after generate and flush commit record.

2. (2 points) Suppose you are you are a machine that has just recovered from a crash. You discover only an abort record in your log for transaction T. Under proper 2PC and logging protocols, mark True or False for each of the following two questions.

   A. You are a participant.

   **B. You crashed during phase 2.**

**Solution:** You discover only an abort record in your log for transaction T, so you are a coordinator who finished phase 1 but crashed during phase 2.

A. False. You are a coordinator.

B. True.

3. (2 points) Suppose you are a participant machine that has just recovered from a crash. There is only one record in your log for transaction T. Under proper 2PC and logging protocols, mark True or False for each of the following two questions.

   A. You know all the messages you received before you crashed.

   **B. You need to decide if you need to commit or abort transaction T.**

**Solution:** You discover only one record in your log for transaction T and you are a participant, so the record must be ”prepare”. You could possibly crash anytime after flushing prepare message and before flushing commit/abort message.

A. False. You cannot tell from the prepare message.

B. True.

4. (2 points) Suppose in 2PC with logging, we have one coordinator and three participants. It takes 30ms for a coordinator to send messages to all participants; 5, 10, and 15ms for participant 1, 2, and 3 to send a message to the coordinator respectively; and 10ms for each machine to generate and flush a record. Assume for the same message, each participant receives it from the coordinator at the same time.

   Under proper 2PC and logging protocols, how long does the whole 2PC process (from the beginning to the coordinator's final log flush) take for a successful commit in the best case?

**Solution:**

130ms

explanation:

phase 1: $30 + 10 + 15 + 10 = 65$ms

phase 2: $30 + 10 + 15 + 10 = 65$ms

total: 130ms

## 6 SQL/Relational Algebra (10.5 points)

Use your knowledge of SQL to deduce what is happening during the Battle of Hogwarts. You may 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);
```

**Disclaimer**: 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. (1.5 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. (1.5 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. (1.5 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;
```

For the following questions: fill in the 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 that **exactly one** answer will be correct.

```
WITH count_evil_good_spells(evil, num_offensive) AS
    (SELECT evil, count(*)
        FROM attacks, wizards, spells
        WHERE ___(4)___ and spell = sid
        AND offensive = ____(5)___
        GROUP BY ____(6)___)
SELECT evil
    FROM count_evil_good_spells
    WHERE num_offensive >= ALL
        (SELECT num_offensive
        FROM ___(7)___);
```

4. (0.5 points) Fill in the blank labeled (4)
   A. `wizid=attackid`
   B. `wizid=attacked`
   **C.** `wizid=attacker`

5. (0.5 points) Fill in the blank labeled (5)
   **A.** `true`
   B. `false`

6. (0.5 points) Fill in the blank labeled (6)
   **A.** `evil`
   B. `spell`
   C. `attacker`

7. (0.5 points) Fill in the blank labeled (7)
   A. `Wizards`
   **B.** `count_evil_good_spells`
   C. `Attacks`

8. (2 points) Select all of the following answers that return the **wizid** of all of the wizards who have not attacked anybody.

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

   **B. $\pi_{\mathbf{wizid}}(\pi_{\mathbf{wizid}}(\textbf{Wizards}) - \pi_{\mathbf{attacker}}(\textbf{Attacks}))$**

   **C. $\pi_{\mathbf{wizid}}(\textbf{Wizards}) - \pi_{\mathbf{attacker}}(\textbf{Spells} \bowtie_{\textbf{sid = spell}} \textbf{Attacks} \bowtie_{\textbf{attacker = wizid}} \textbf{Wizards})$**

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

9. (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. $\pi_{\mathbf{attacked}}(\textbf{Spells} \bowtie_{\textbf{sid = spell}} \textbf{Attacks} \bowtie_{\textbf{attacker = wizid}} \sigma_{\textbf{house ='Gryffindor'}}(\textbf{Wizards}))$**

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

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

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

## 7 B+ Trees (15 points)

1. (5 points) For each of the following five questions, mark True or False.

   A. You cannot build a B+ tree on a variable length column such as a string.

   B. Indexes are used to optimize tables that are written to much more frequently than they are read from.

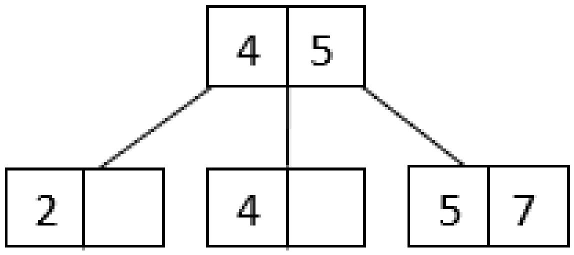
   C. For alternative 2 trees, clustered indexes will not save any IOs over unclustered indexes when you are only doing equality search.

   **D. Alternative 1 indexes will usually be able to store fewer records in each leaf than Alternative 2 indexes.**

   E. With a series of carefully constructed inserts, an adversary can cause a B+ tree to be unbalanced, so that some lookups visit more nodes than others.

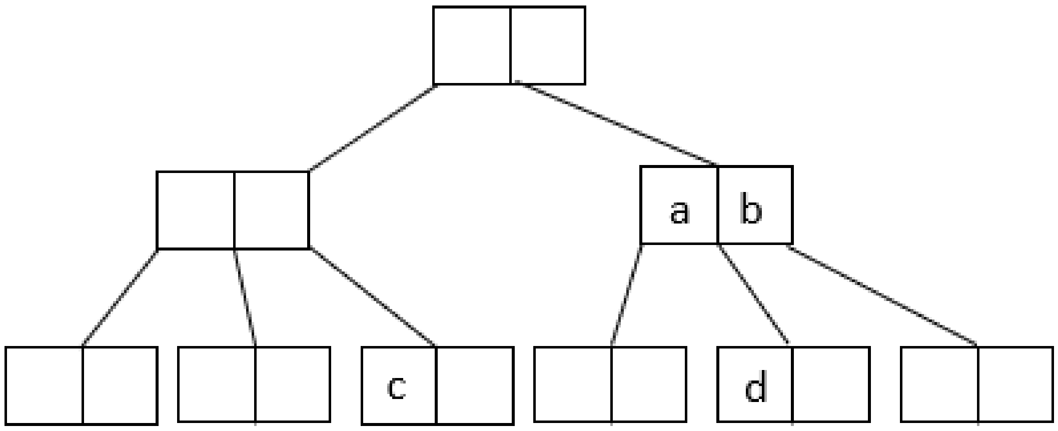
**Solution:** a. False - there is no such constraint
b. False - indexes speed up reads but writes are often more costly
c. False - for duplicates, having clustering is beneficial
d. True - a record is most likely going to be much larger than the recordid
e. False - B+ are self balancing so they will always perform well

2. (4 points) Given the following B+ tree:



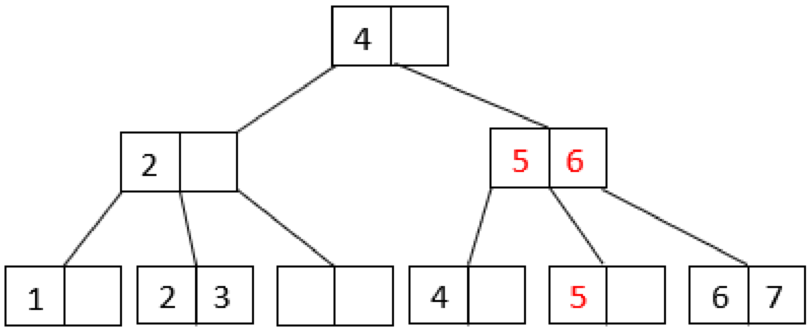
What will the B+ tree look like after inserting 1, 3, and 6 in that order? Fill in the answer sheet with what number goes in boxes a-d using the following template. Put none if the box is empty. Some pages in the template may not have any entries on in them.

**Reminder**: when a node splits and there is an odd number of entries, the majority goes to the right node.



**Solution:** a. 5 b. 6 c. none d. 5

Full tree:



For the following problems consider an alternative 2 unclustered B+ tree of height 3 on the sid column, which is a primary key. As a reminder, we define the height of a tree with only a root as 0. Assume that no tree pages or data pages are initially in the buffer pool.

3. (2 points) In the best case, how many IOs (index and heap) will it take to insert a tuple? Assume no tree nodes split, and that modifications to any page will immediately be flushed to disk.

**Solution:** 4 IOs to reach leaf + 1 to read data page + 1 to write data page + 1 to write leaf = **7 IOs**

4. (2 points) If we want to do a range search on this tree and we estimate that about half the tuples will fall in this range, should we do an index scan, or should we simply do a full table scan?

   **Solution: Table scan -** unclustered index will produce an IO for every tuple so you will end up doing way more IOs than the number of pages

5. (2 points) How many IOs (index and heap) will an index scan on the condition: sid $>$ 10 and sid $<$ 15 take? Assume that only two leaves have records matching the conditions, and that each of those leaves have three records matching the condition.

   **Solution:** 4 IOs to reach leaf + 3 to read data pages + 1 to reach next leaf + 3 to read data pages = **11 IOs**

## 8 Query Processing and Parallel DBMS (10 points)

1. (2 points) For each of the following three questions, mark True or False.

   A. Performing an index nested loop join with a clustered index on the inner relation always requires more I/Os than an index nested loop join with an unclustered index on the inner relation (for the same tables).

   B. Pipeline parallelism scales up with the amount of data.

   **C. Performing key lookup over hash partitioned data requires fewer I/Os than if the data had been round-robin partitioned.**

> **Solution:** A. is incorrect because a clustered index will always be equally or less costly than at an index nested loop join on an unclustered index B. is incorrect because it scales up to the pipeline depth C. is correct.

We want to join together the Owners and Pets table on petID so we can later on compare the owner's happiness to the pet's happiness. Suppose we have the following tables and variables:

`Owners: (ownerID, name, happiness, petID)` (ownerID underlined)

`Pets: (petID, happiness, ownerID)` (petID underlined)

| **variable** | **symbol** | **value** |
|---|---|---|
| pages of `Owners` table | $[O]$ | 30 |
| tuples per `Owners` page | $p_O$ | 5 |
| pages of `Pets` table | $[P]$ | 45 |
| tuples per `Pets` page | $p_P$ | 5 |
| pages in memory per machine to perform the join | B | 6 |
| Number of machines | M | 3 |
| Size of Page | s | 2KB |

**Note:** For these questions, do **NOT** include the cost of writing matching output, but **DO** include the cost of scanning the tables.

Assume that we have **three** machines. We want to join `Owners` and `Pets` on `Owners.petID = Pets.petID`.

2. (2 points) If the data starts out round-robin partitioned, what is the total amount of data **in KB** transmitted over the network during repartitioning in order to perform **grace hash join**?

> **Solution:** On expectation, each node keeps $\frac{1}{3}$ of its data local and ships the other $\frac{2}{3}$ during repartitioning, for a total of $\frac{2}{3} * 75 * 2 = 100KB$

3. (2 points) Assume that we can partition and perform the initial pass of hashing at the same time. Assuming that the data starts out round-robin partitioned across the machines, how many (disk) I/Os are needed **in total across all machines** to perform **grace hash join**?

**Solution:** Since we have 6 pages in memory per machine, we can split the incoming pages into B - 1 = 5 partitions. Size of pets table on one machine: 15/5 = 3 pages per partition Size of owners table on one machine: 10 / 5 = 2 pages per partition The partitions are now small enough to fit in the in memory hash table so we will finish in two passes. The cost is therefore 3([O] + [P]) = 3(75) = 225 I/Os.

4. (2 points) Assume that the data starts out partitioned perfectly across the machines for a **sort-merge join**, and no network communication is required. How many (disk) I/Os are needed **per machine** to perform sort merge join, using the optimization if possible? (Note: your answer should be per machine, in contrast to the previous question.)

**Solution:** We will have 10 pages of the Owners table and 15 pages of the Pets table per machine.

First pass: # sorted runs Owners = $ceil(10/6) = 2$, # sorted runs Pets = $ceil(15/6) = 3$

We can perform the optimization because the total # of sorted runs for Owners and Pets < 6.

Second pass: Merge and join Owners and Pest at the same time

The cost is therefore 3([P] + [O]) = 3(75) = 225 I/Os for all the data There are 3 computers so 225 / 3 = 75

5. (2 points) How many (disk) I/Os would be needed to perform sort merge join using the optimization if possible if we only had **one** machine and **6 buffer pages** for this one machine instead?

**Solution:** First pass: Pets: ceil(45/ 6) = 8 partitions Owners: 30 / 6 = 5 partitions We will therefore have 8 partitions of 6 pages each for the Pets table and 5 partitions of 6 pages each for the Owners table.

Second pass (Merge 5 partitions together at once): Pets: ceil(8 / 5) = 2 sorted runs Owners: ceil (5 / 6) = 1 sorted runs

We will now perform the optimization so the total cost is 5([P] + [O]) = 5 * 75 = 375 I/Os

## 9 Query Optimization (16 points)

A relational database has the table:

```
CREATE TABLE Employee(PNR text, SALARY integer, SEX text, DEPT text, PRIMARY KEY(PNR))
```

The cost-based optimizer uses the following statistics stored as meta-data in the DBMS's system catalogue:

- There are 10000 rows in the table.
- There are 20 departments in the company.
- SEX is either 'Male' or 'Female'.
- The lowest salary is 20000 and the highest is 100000.

The query optimizer make the following assumptions on statistics:

- The query optimizer assumes even distributions of data values in columns
- The query optimizer assumes independence between values in different columns

For questions of selectivity, you may answer either with a fraction or a decimal.

1. (2 points) What is the selectivity of the condition: SALARY $<$ 25000?

   **Solution:** 0.06250 or 5000/80001

   Explanation:

   (25000-20000)/(100000-20000+1)

2. (2 points) What is the selectivity of the condition: (SALARY $>$ 100000 OR SEX != 'Male') And DEPT='Toys'?

   **Solution:** 0.025 or 1/40

   Explanation:

   There is no salary $>$ 100000. So it's (0+0.5)*1/20=0.025

3. (2 points) What is the selectivity of the condition: (SEX = 'Male' OR SALARY $>$ 80000)?

   **Solution:** 0.62500 or 100001/160002

   $1/2 + 20000/80001 - (1/2 * 20000/80001) = 100001/160002$.

4. (2 points) What is the selectivity of the condition: ((SEX= 'Male' AND DEPT = 'Toys') OR (SEX= 'Female' AND DEPT = 'Tools')) AND (SALARY > 50000 OR SALARY < 25000)?

**Solution:** 0.03437 or 2750/80001

Explanation:

s1 = 0.5 * 1/20 = 0.025 (SEX= 'Male' AND DEPT = 'Toys')

s2 = 0.5 * 1/20 = 0.025 (SEX= 'Female' AND DEPT = 'Tools')

s3 = (100000-50000)/(100000-20000+1) = 50000/80001 (SALARY > 50000 )

s4 = (25000-20000)/(100000-20000+1) = 5000/80001 (SALARY <25000)

combined: (s1 + s2)*(s3 + s4) = 2750/80001

Consider the following SQL query that finds all applicants who want to major in CS, live in Berkeley, and go to a school ranked better than 10 (i.e., rank < 10).

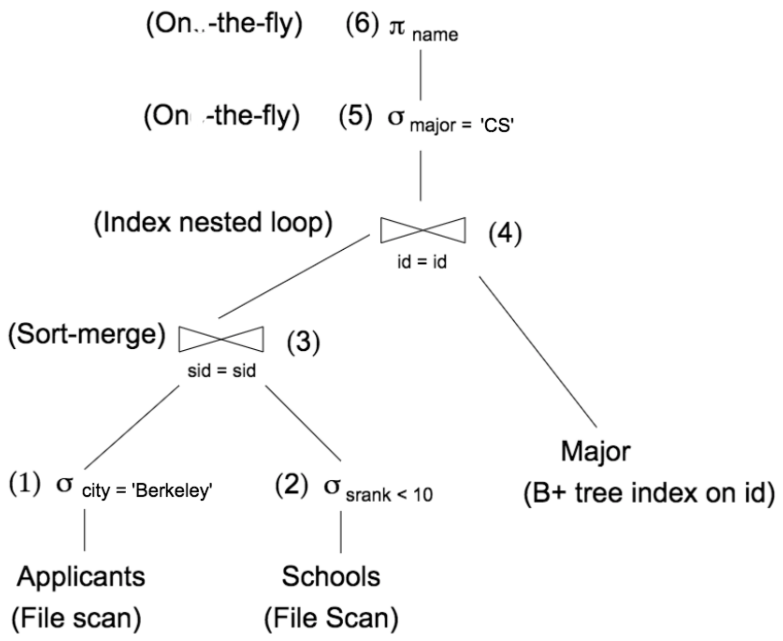
| Relation | Cardinality | Number of pages | Primary key |
|---|---|---|---|
| `Applicants (id, name, city, sid)` | 2,000 | 100 | `id` |
| `Schools (sid, sname, srank)` | 100 | 10 | `sid` |
| `Major (id, major)` | 3,000 | 200 | `(id,major)` |

```
SELECT A.name
    FROM Applicants A, Schools S, Major M
    WHERE A.sid = S.sid AND A.id = M.id
    AND A.city = 'Berkeley' AND S.srank < 10 AND M.major = 'CS'
```

Assume that:

- Each school has a unique rank number (srank value) between 1 and 100 (both inclusive).
- There are 20 different cities.
- Applicants.sid is a foreign key that references Schools.sid.
- Major.id is a foreign key that references Applicants.id.
- There is an unclustered, alternative 2 B+ tree index on Major.id and all index pages (a total of 5) are already in memory.
- For Major table, all tuples belonging to same student are stored in one physical page.
- The buffer size is 150 pages.

The query optimizer is currently considering the below query plan.



5. (3 points) What is the cardinality of the output of operator 1 in unit of tuples?

**Solution:** 100

Explanation: 2000/20 = 100 tuples.

6. (5 points) Suppose the output cardinality of operator 3 is known to be 9 tuples and they have different Applicant ids. What is the total I/O cost of this whole plan? Do not include the cost of writing the final result.

**Solution:** 119

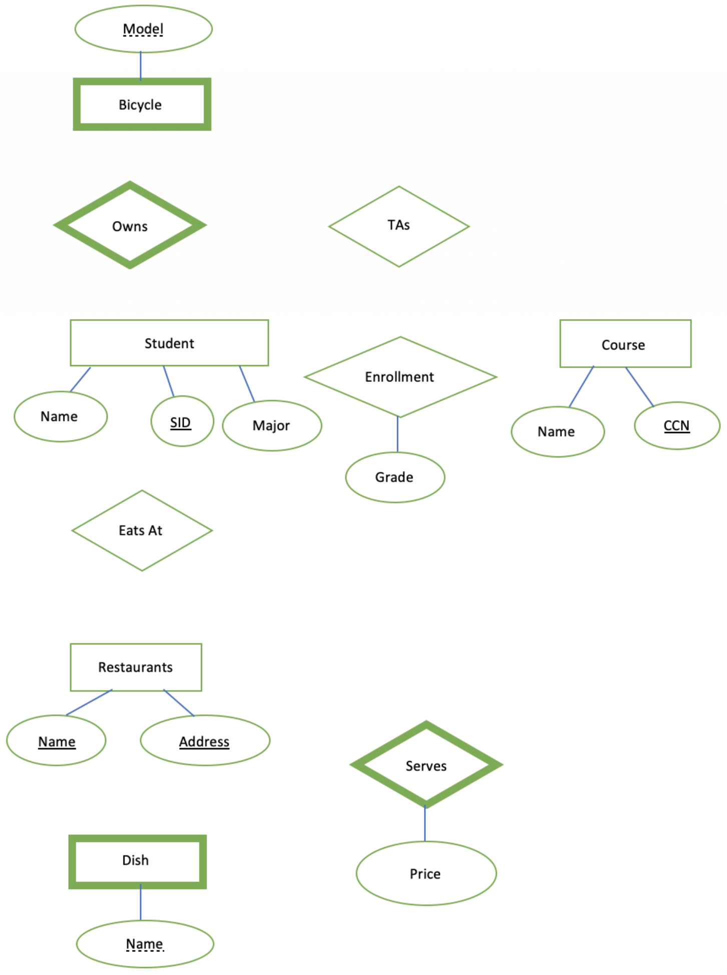
Explanation:

(1) The cost of scanning Applicants is 100 I/Os. (2) The cost of scanning Schools is 10 I/Os. (3) Given that the input to this operator is will be smaller than the buffer pool size in any circumstance, we can do an in-memory sort-merge join. No additional I/O cost. (4) The index-nested loop join must perform one look-up for each input tuple in the outer relation. We assume that each student only declares a handful of majors, so all the matches fit in one page. The cost of this operator is thus 9 I/Os. (5) and (6) are done on-the-fly, so there are no I/Os associated with these operators.

## 10 ER Diagrams (10 points)

For questions 1-10, you will fill in the following ER Diagram, which models parts of a student's routine in a University. (Hint: You might want to fill the diagram while you read these requirements here). The constraints are listed below.

- Students own at least one bicycle. Each bicycle is uniquely identified by its owner and the bicycle model.
- Students must take at least one course, and every course must have at least one student.
- Students may TA for at most one course, but are under no obligation to teach at all. Every course can have as many TAs as it needs, or none at all.
- Students can choose to eat at restaurants, but are under no obligation to do so. Every restaurant must have some students eating at it, however.
- Each restaurant, identified by its address and name, has to serve multiple dishes. Each dish is uniquely identified by its name and the restaurant serving it.



For each of the following questions, mark **one** of the following choices:

A. Thin Arrow

B. Thick Arrow

C. Thin Line

D. Thick Line

1. (1 point) Which edge should we draw to connect the **Students** entity with the **Enrollment** relationship set?

**Solution:** Thick Line

2. (1 point) Which edge should we draw to connect the **Course** entity with the **Enrollment** relationship set?

**Solution:** Thick Line

3. (1 point) Which edge should we draw to connect the **Students** entity with the **TAs** relationship set?

**Solution:** Thin Arrow

4. (1 point) Which edge should we draw to connect the **Course** entity with the **TAs** relationship set?

**Solution:** Thin Line

5. (1 point) Which edge should we draw to connect the **Student** entity with the **Owns** relationship set?

**Solution:** Thick Line

6. (1 point) Which edge should we draw to connect the **Bicycle** entity with the **Owns** relationship set?

**Solution:** Thick Arrow

7. (1 point) Which edge should we draw to connect the **Restaurants** entity with the **Eats At** relationship set?

**Solution:** Thick Line

8. (1 point) Which edge should we draw to connect the **Student** entity with the **Eats At** relationship set?

**Solution:** Thin Line

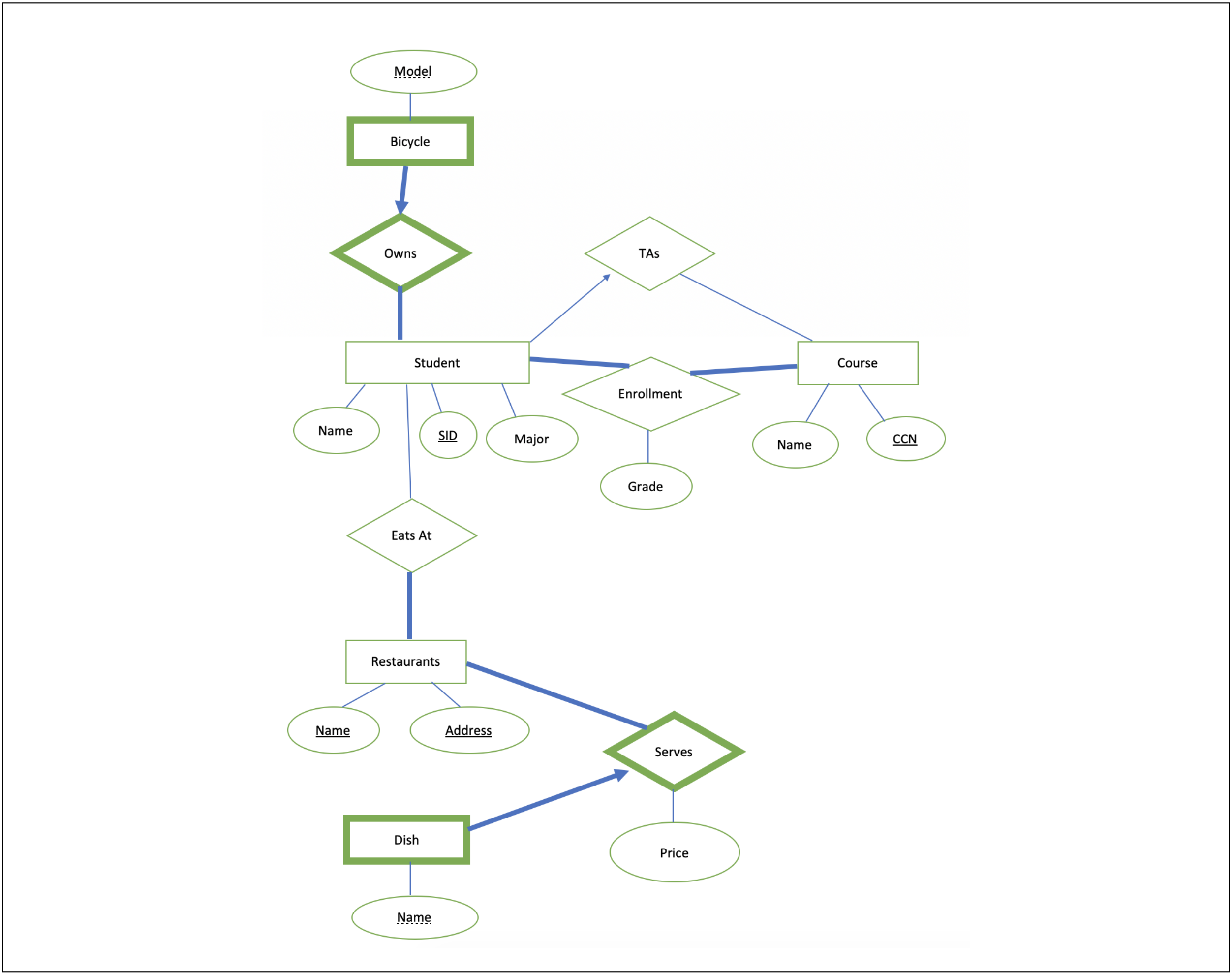
9. (1 point) Which edge should we draw to connect the **Restaurants** entity with the **Serves** relationship set?

**Solution:** Thick Line

10. (1 point) Which edge should we draw to connect the **Dish** entity with the **Serves** relationship set?

**Solution:** Thick Arrow

**Solution:** See below for the completed ER Diagram.


Model
Bicycle
Owns
TAs
Student
Enrollment
Course
Name
SID
Major
Grade
Name
CCN
Eats At
Restaurants
Name
Address
Serves
Dish
Price
Name