My Name: _____

My Course Account: _____

Midterm 2: CS186, Spring 2015

Prof. J. Hellerstein

You should receive a double-sided answer sheet and an 8-page exam.

Mark your name and login on both sides of the answer sheet, and in the blanks above.

For each question, **place only your final answer on the answer sheet**—do not show work or formulas there.

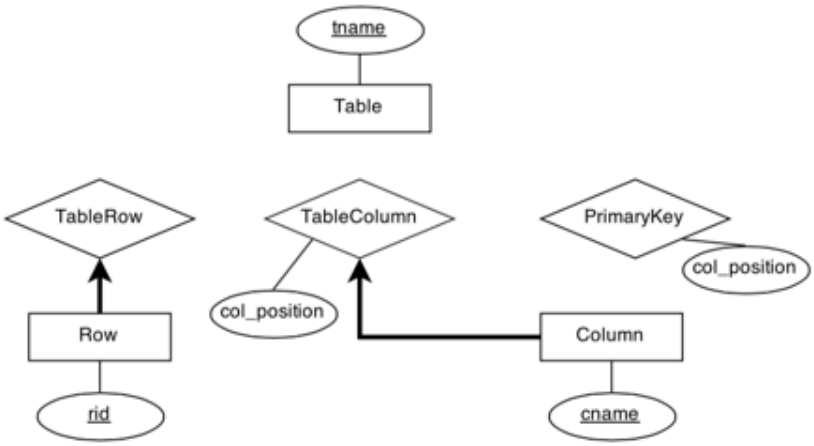
You may use the backs of the questions for scratch paper, but do not tear off any pages.

We will ask you to turn in your question sheets as well as your answers.

I. ER Diagrams [18 points]

1. [4 points] **True or False:**
   a. ER diagrams are used to model the physical schema of a database.
   b. An entity set E can be associated with a relationship set R more than once.
   c. At most 2 distinct entity sets can be associated with any relationship set R.
   d. A weak entity set must have total participation in its identifying relationship set.

Chancellor Dirks wants you to explain how a **relational database management system**, (e.g. Postgres) works. Dirks is familiar with reading ER diagrams, so we've created the diagram below, but it is missing a few edges.



Answer questions 2 to 5 choosing the best option from the following multiple choice options:

A. partial participation, non-key
B. partial participation, key
C. total participation, non-key
D. total participation, key
E. none of the above

2. [2 point] What is the correct edge between TableRow and Table?
3. [2 point] What is the correct edge between TableColumn and Table?
4. [2 point] What is the correct edge between Column and PrimaryKey?
5. [2 point] What is the correct edge between Table and PrimaryKey?

ER to SQL

6. [6 points] On your answer sheet, fill in the blanks to convert the **Row entity set**, and the **TableColumn relationship set** in the ER diagram above into SQL tables. You may not need all

the blanks.

II. FDs & Normalization [16 points]

1. [4 points] Consider the attribute set R = ABCDEF and the functional dependency set F = {AD → B, A → E, C → E, DEF → A, F → D}. Find a candidate key of R.

2. [6 points] Given the attribute set R = ABCDEFGH and the functional dependency set F = {BC → GH, AD → E, A → H, E → BCF, G → H}, decompose R into BCNF by decomposing *in the order of the given functional dependencies*.

3. [6 points] Given the attribute set R = ABCDEF and the functional dependency set F = {B → D, E → F, D → E, D → B, F → BD}.
    a. Is the decomposition ABDE, BCDF lossless?
    b. If not, what functional dependency could you add to make it lossless?

III. Transactions/CC [17 points]

T1	R(A)						R(C)	W(B)	COM		
T2				R(A)	W(A)	COM					
T3		R(A)	R(C)							W(B)	COM

1. [4 points] Consider the preceding schedule for three transactions. R indicates a read of a page, W indicates a write of a page, and COM indicates a commit.
    a. Draw the arrows for the dependency graph for this schedule.
    b. Is the schedule conflict serializable?

2. [5 points] Which of the following changes to the above schedule will result in a schedule that is possible using strict two-phase locking? (**Mark all that apply**)
    A. Make T1 abort instead of commit
    B. Make T2 abort instead of commit
    C. Remove R(A) from T1 and T3
    D. Remove T2 from the schedule
    E. Make T3 write to a page D instead of page B

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

3. [3 points] Consider the preceding schedule. A, B, and C are positive integers. Each transaction reads in two integers, adds them, and writes the result. Which of the following statements is/are true? (**Mark all that apply**)
    A. The schedule avoids cascading aborts
    B. The schedule is conflict serializable
    C. The schedule is equivalent to some serial schedule

T1	RS(D)							RX(D)	
T2		RS(A)							RS(C)
T3			RS(D)	RX(B)	RX(A)				
T4						RX(C)	RS(B)		

4. [5 points] Assume that we are using strict two-phase locking, and the objects being locked are pages. RS indicates a request for a shared lock on a page, and RX indicates a request for an exclusive lock on a page. No transactions commit or abort during this time.
    a. Draw the arrows for the waits-for graph at the end of this schedule.
    b. Does the schedule lead to deadlock?

IV. Text Search [14 points]

We have a search engine with a corpus containing **only** the documents
Document ID 1: “A time to plant and a time to reap”
Document ID 2: “Time for you and time for me”
Document ID 3: “Time flies”

1a. [2 points] Given that the stemmed version of the word “flies” is the term “fly”, what is the TF-IDF of “fly” in document 3?

1b. [2 points] What is the TF-IDF of “time” in document 1?

2. [4 points] We want to perform cosine similarity ranking on these documents, so we represent each document with a vector containing the TF-IDFs of different terms in our documents. Which of the following lists of terms would allow us to correctly compute cosine similarity? Assume that standard stop words include “a”, "and", "to", and "for". (**Mark all that apply**)

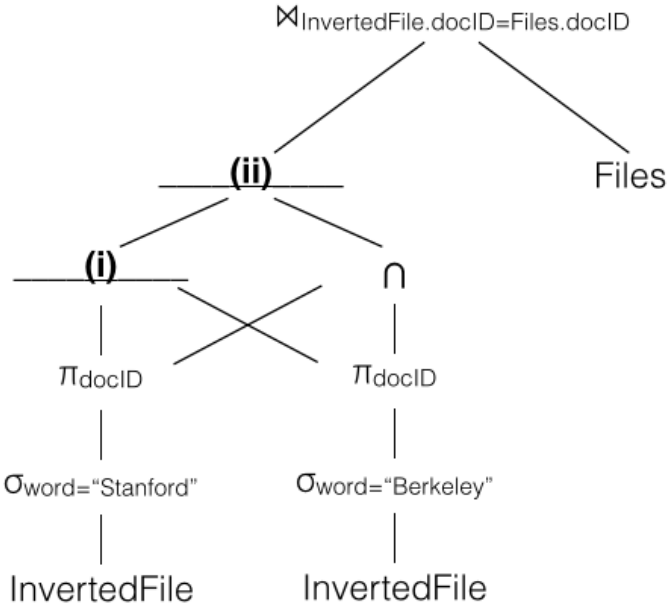
A. (“plant”, “you”, “fly”, “me”, “reap”, “time”)
B. (“time”, “plant”, “reap”, “you”, me”, “fly”)
C. (“time”, “plant”, “you”, “me”, “fly”)
D. (“plant”, “reap”, “you”, “me”, “fly”)

3. [4 points] Suppose we issue our engine the query “plant OR fly”. In the spaces provided, write **only** the document IDs which are returned by this query, in order of relevance **as determined by cosine similarity**. (You may want to leave some spaces **blank**.) Break ties with smaller document IDs first, and then larger document IDs.

4a. [2 point] For this question, assume that we have a text index with the following schema:

```
Files(docID text, content text)
InvertedFile(word text, docID text)
B+-tree Alternative 2 on Files
B+-tree w/"postings list" at leaves on InvertedFile.word
```

If we have two terms A and B, the exclusive OR operator "XOR" returns all documents which contain *either* term "A" or term "B", but not *both*. Suppose we run the query "Berkeley XOR Stanford" on our search engine. Fill in the spots on your answer sheet corresponding to the missing parts of the generated query plan below (indicated by (i) and (ii))



4b. [2 points] One of the following join algorithms would be ideal for evaluating the logical operator at the top of our query plan $\bowtie_{\text{InvertedFile.docID=Files.docID}}$. Which one should we choose? (You can ignore issues of parallelism when answering this question, though they don't really affect the answer.)

A. Block Nested Loops join
B. Index Nested Loops join
C. Hash join
D. Sort Merge join

V. Query Optimization [17 points]

Suppose the "System R" assumptions about uniformity and independence from lecture hold. Assume that costs are estimated as a number of I/Os, without differentiating random and sequential I/O cost.

Consider the following relational schema and statistics:

```
Students   (sid, name, age, gpa, year)
Courses    (cid, name, professor)
Enrollment (sid, cid, credits)
```

	Tuples	Tuples / Page	Ranges	Distinct Values	Indexes
Students	40,000	20	SID: 0 to 40,000 age: 15 to 44	gpa: 500 age: 30	unclustered B+-tree on age (800 pages, d= 4) clustered B+-tree on sid (400 pages, d = 3)
Enrollment	60,000	50	-	-	clustered B+-tree on cid (200 pages, d = 2)
Courses	800	40	-	professor: 100 name: 200	unclustered B+-tree on prof (100 pages, d = 3)

1. Suppose we run the following query. (<> is the SQL syntax for "not equals").

```
SELECT *
  FROM Courses
 WHERE name <> "CS 186"
   AND professor = "Hellerstein";
```

a. [2 points] What is the selectivity of each conjunct in the WHERE clause?

b. [2 points] Which single table access plan would we choose?

A) Sequential/File Scan on Courses
B) Sequential/File Scan on Enrollment
C) Index scan of unclustered B+Tree on Courses.professor
D) Index scan of clustered B+Tree on Enrollment.cid

2. [6 points] Suppose we run the following query. On the line labeled “considered”, mark the letters of *all* 2-table subplans that will be considered. On the line labeled “chosen”, mark the letters of *all* 2-table subplans that are chosen. Ignore interesting orders.

```
SELECT *
  FROM Students S, Courses C, Enrollment E
 WHERE S.age > 40
   AND C.name LIKE "CS%"
   AND S.sid = E.sid
   AND E.cid = C.cid;
```

A) S ⋈ C (200,000 IOs)

B) C ⋈ S (300,000 IOs)

C) S ⋈ E (400,000,000 IOs)

D) E ⋈ S (500,000,000 IOs)

E) C ⋈ E (300,000 IOs)

F) E ⋈ C (400,000 IOs)

3. [3 points] Now, mark the letters for *all* three-table access plans that would be considered, and write down the letter of the final query plan that would be chosen. (Cost given are cumulative for the full query. Again, ignore interesting orders in formulating your answer.)

A) E ⋈ (S ⋈ C) (3,900,000,000 IOs)

B) E ⋈ (C ⋈ S) (6,000,000,000 IOs)

C) (C ⋈ S) ⋈ E (5,000,000,000 IOs)

D) (S ⋈ C) ⋈ E (8,000,000,000 IOs)

E) C ⋈ (S ⋈ E) (7,000,000,000 IOs)

F) C ⋈ (E ⋈ S) (9,000,000,000 IOs)

G) (S ⋈ E) ⋈ C (7,500,000,000 IOs)

H) (E ⋈ S) ⋈ C (8,000,000,000 IOs)

I) S ⋈ (C ⋈ E) (3,500,000,000 IOs)

J) S ⋈ (E ⋈ C) (6,000,000,000 IOs)

K) (C ⋈ E) ⋈ S (4,000,000,000 IOs)

L) (E ⋈ C) ⋈ S (10,000,000,000 IOs)

4. [4 points] **True or False:**

a. If we had considered interesting orders in the query of Question 3, it would have further pruned the number of plans chosen.
b. In Question 3, it would be beneficial to consider interesting orders on C.name during optimization.
c. In Question 3, it would be beneficial to consider interesting orders on C.cid during optimization.

d. A modern optimizer should consider “interesting hashes”, since the output of a hash-based operator (e.g. hash join) is organized in a way that would decrease the cost of performing a subsequent hash-based operator (e.g. group-by).

Name:_____

Class Login (e.g. cs186-aa): _____

Class Login – person on your left:_____

Class Login – person on your right:_____

I. ER Diagrams

1a. True / False

1b. True / False

1c. True / False

1d. True / False

2. A / B / C / D / E

3. A / B / C / D / E

4. A / B / C / D / E

5. A / B / C / D / E

6.
```
CREATE TABLE Row (
   _____
   _____
   PRIMARY KEY (_____)
);

  CREATE TABLE TableColumn (
  _____ STRING,

  _____,
  col_position INTEGER,

  FOREIGN KEY (_____)
      REFERENCES Table
  FOREIGN KEY (_____)

      REFERENCES _____

  PRIMARY KEY (_____)
);
```

II. FDs

1. _____

2. _____

3a. Yes / No

3b. _____

III. Transactions/CC

1a.

1b. Yes / No

2. A / B / C / D / E

3. A / B / C

4a.

4b. Yes / No

.

IV. Text Search

1a. _____ log (_____)

1b. _____ log (_____)

2. A / B / C / D

3. _____, _____, _____,

4a, i. _____

4a, ii. _____

4b. A / B / C / D

V. Query Optimization

1a. name: _____

professor: _____

1b. A / B / C / D

2.

Considered:

A / B / C / D / E / F

Chosen:

A / B / C / D / E / F

3.

Considered:

A / B / C / D / E / F /

G / H / I / J / K / L

Final Plan:

_____

4a. True / False

4b. True / False

4c. True / False

4d. True / False