

Name \_\_\_\_\_

Class Account: \_\_\_\_\_

UNIVERSITY OF CALIFORNIA  
Department of EECS, Computer Science Division

CS186  
Fall 2006

Hellerstein/Olston  
Final Exam

### Final Exam: Introduction to Database Systems

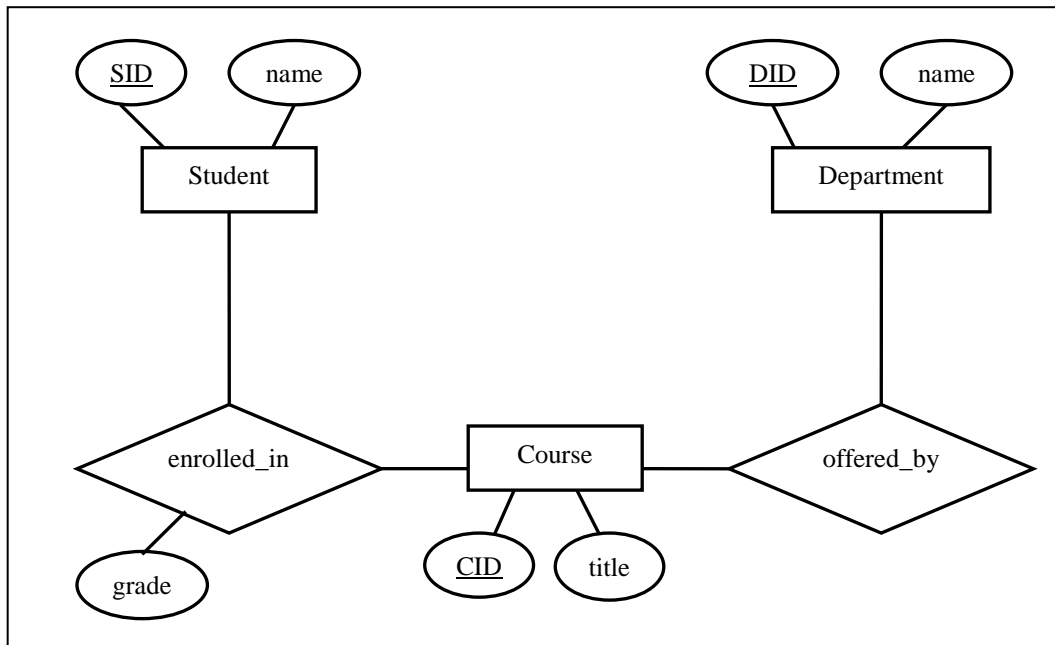
This exam has six problems, worth different amounts of points each. Each problem is made up of multiple questions. You should read through the exam quickly and plan your time-management accordingly. Before beginning to answer a problem, be sure to read it carefully and to *answer all parts of every problem!*

You **must** write your answers on the exam. If you need scratch paper, you can use the back sides of the pages. ***Do not tear pages off of your exam!***

Good luck!

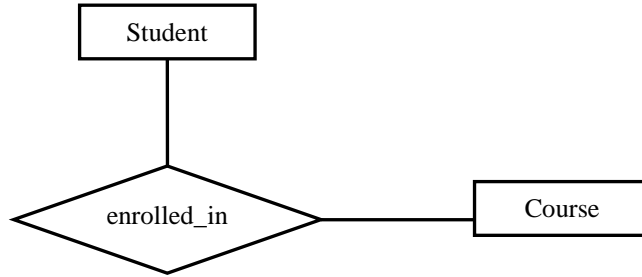
#### 1. ER Diagrams [12 points]

In this problem you will be asked to modify various fragments of the E-R diagram below. Be sure to make very clear when you draw a **dark** line rather than a normal line! The questions will be based on the diagram below, but *do not draw on this page* – you will be given places to draw on the following pages.

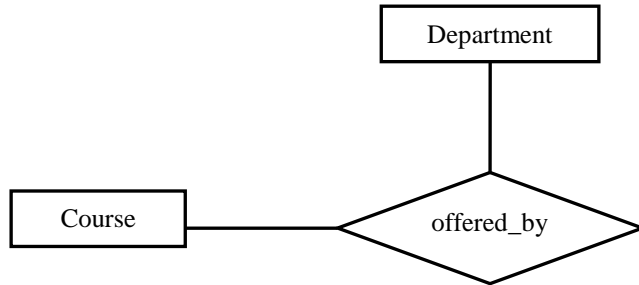


A. [4 points] For each of the following constraints, make the necessary modifications on the given portion of the diagrams below, so that the constraint is captured.

- i. Each Course has to have **at least one** Student enrolled.

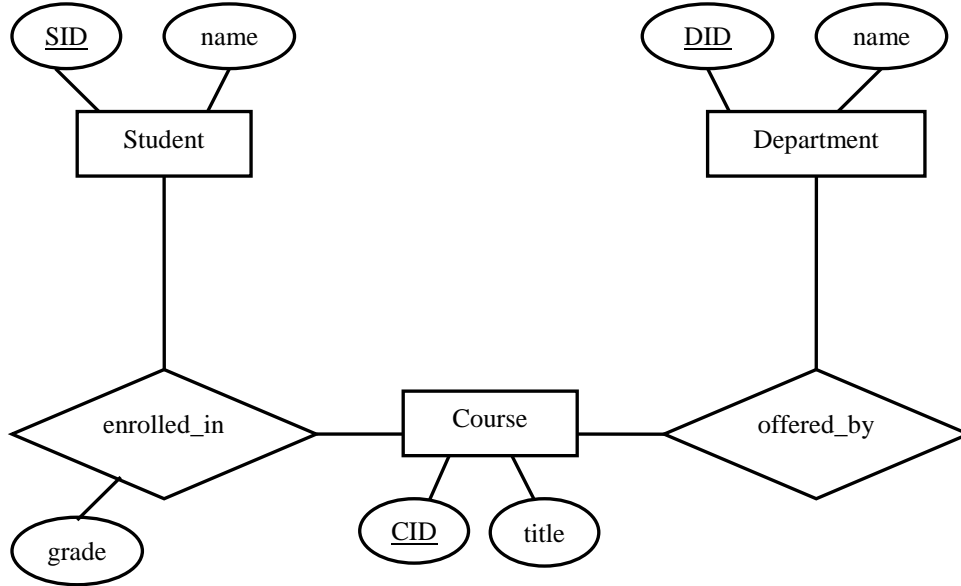


- ii. Each Course must be offered by **exactly one** Department.

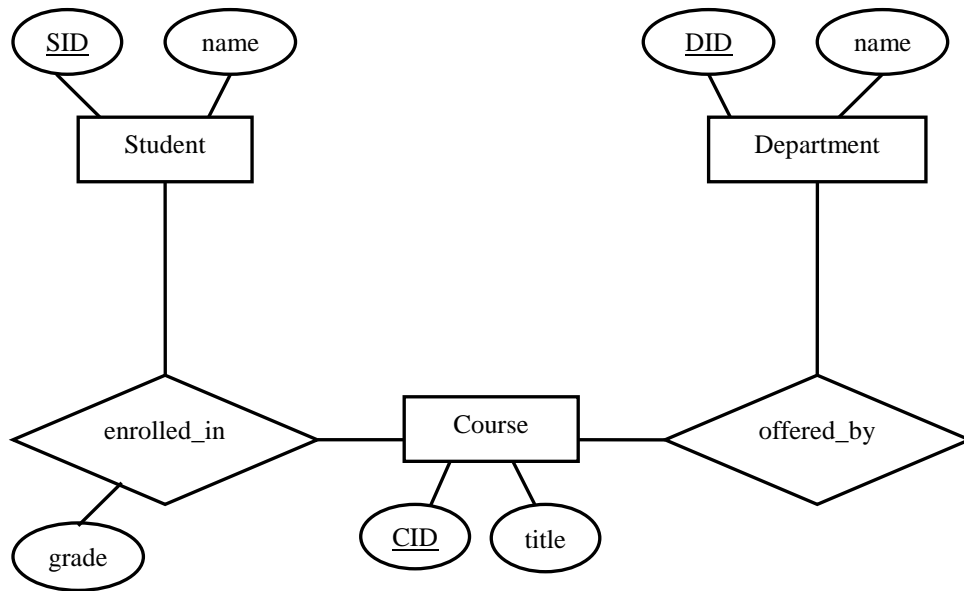


B. [4 points] For each of the following additions, add whatever is necessary to the given diagram.  
*Pay attention to the specified constraints!*

i. Each Course is allowed to hire **at most one** Student to be a GSI.



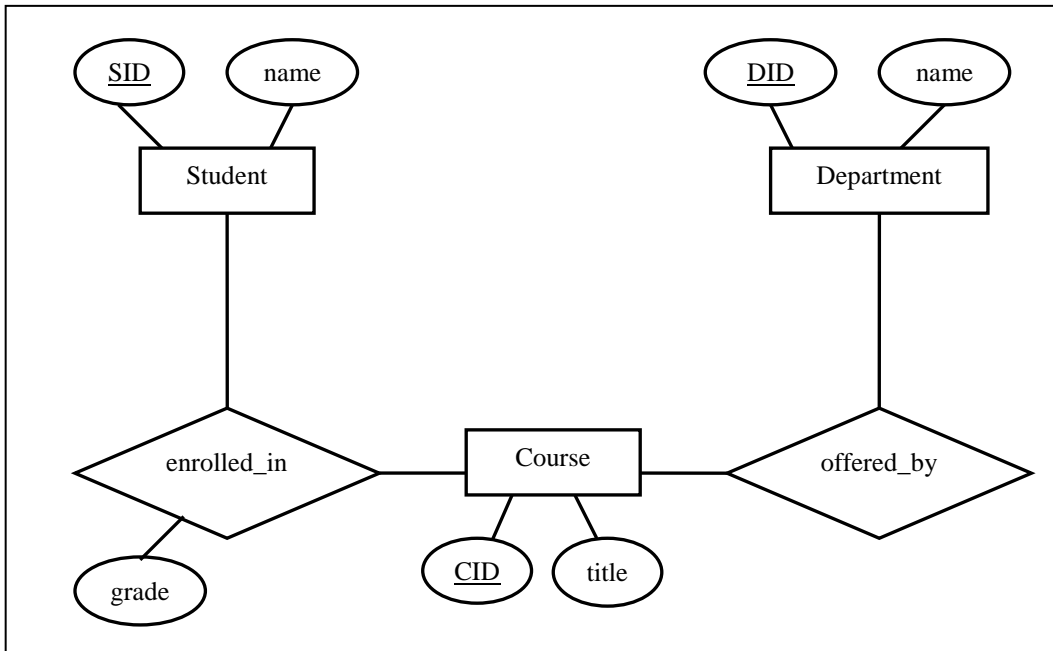
ii. We want to keep track of which Diplomas are earned by Students. For each Diploma, record the degree type (BA, MS, PhD, etc). A Student can **not** earn more than one diploma of a given degree type. Also, a particular Diploma should **only** exist if a Student receives it!



C. [4 points] Now consider the following relational schema, derived from the original ER diagram of page 1 that is reproduced for your reference at the bottom of the page.

Student (SID, name)  
Department (DID, name)  
Course (CID, title)  
Enrolled (SID, CID)

What is *the simplest* way to fix this schema to include **both** the `offered_by` relationship **and** the constraint of part Aii (each course offered by exactly one department)? You should be able to do this without introducing any new tables.



## 2. Query Execution [16 points, 2 per question]

Suppose you have the following table:

```
People (PID, name, age)
Cities (CID, name, population)
```

And you try to execute the following query:

```
SELECT P.name
FROM People P, Cities C
WHERE P.name = C.name
```

But you're using a DBMS with a query optimizer that can't make a decision on its own. Instead, you are given full control of the query plan. So, it's up to you to decide: sort-merge join or hash join?

- A. Suppose you know that the tuples in People are already sorted by name: does that affect your decision? If so, which one of the two algorithms would most likely be best?
- a. No effect
  - b. Sort-merge
  - c. Hash

Answer: \_\_\_\_\_

- B. Ignoring the information in part (A), suppose you know that the tuples in People are already sorted by age: does that affect your decision? If so, which one of the two algorithms would most likely be best?
- a. No effect
  - b. Sort-merge
  - c. Hash

Answer: \_\_\_\_\_

- C. Ignoring the information in parts (A) and (B), suppose you know that over 90% of the People tuples are named "John": does that affect your decision? If so, which one of the two algorithms would most likely be best?
- a. No effect
  - b. Sort-merge
  - c. Hash

Answer: \_\_\_\_\_

Name \_\_\_\_\_

Say you have 100 pages of People tuples and 20 pages of Cities tuples, with a total of 500 values evenly distributed for the name column in each table. Assume that you have 10 frames in the buffer pool. For any I/O computations, ignore the cost of writing out the results.

- D. For each table, how many passes will it take to sort it with external sorting? Assume that the first pass uses basic quicksort.

People: \_\_\_\_\_ Cities: \_\_\_\_\_

- E. How many I/O's will it take to execute the basic sort-merge join algorithm (without the "refinement" that performs the join during the last merging pass)?
- F. If the data happens to be sorted by name, would changing the sorting method used in Pass 0 of sort-merge join have an impact on the total number of passes? Circle one; no explanation is necessary.

YES

NO

- G. Assuming you write perfect hash functions that split up the data evenly into buckets, how many I/O's will it take to execute the query using the basic hash join algorithm with Cities as the "inner" relation (i.e. during the matching phase, partitions of Cities are loaded into hash tables)?
- H. Assume that you have a habit of *always* writing functions that hash half of the values to 0 and distribute the rest of the values evenly over the other possible outputs. Each of your hash functions chooses a different random set of values to hash to 0. In that case, how many I/O's will the scenario from part (G) take?

**3. ARIES Logging and Recovery [22 points]**

The log below is taken from a database system that has crashed. Assume that the dark grey line in the log labeled CRASH denotes the point at which the system crashed. Assume that the Dirty Page Table and Transaction Table were empty prior to LSN 00 and that there were no checkpoints.

LSN	LOG	prevLSN	undoNextLSN
00	T1 writes P1		
10	T2 writes P2		
20	T3 writes P3		
30	T2 Commit	10	
40	T3 Abort	20	
50	CLR: Undo T3 LSN 20	40	
60	T2 End	30	
70	T1 writes P2	00	
<b>CRASH</b>			

A. [6 points] Perform the *analysis* phase on the above log and write the resulting Transaction Table and Dirty Page Table as they exist at the end of analysis below.

Dirty Page Table		Transaction Table		
Page ID	RecLSN	Transaction ID	Status	LastLSN

B. [2 point] At what LSN would the REDO phase begin?

C. [6 points] Perform the *undo* phase of ARIES, writing any additional log records **on the log on the previous page**, in the spaces below the grey CRASH line.

D. [2 point] Let's consider a more general case. Suppose that a bug in a database system running ARIES causes the system to crash at some random point during recovery during the entire month of December. (Assume that the bug does not corrupt the log, it just causes the system to halt during recovery.) On the first of January, when the database finally has a chance to completely recover, *at most* how many CLR's will be written to the log?

We want a symbolic answer, **not a number**. Use only (but not necessarily all of) the following symbols and functions in your answer:

$T_i$ : transaction  $i$  that was uncommitted at the time of the initial crash. Assume there are  $U$  of these.

$C_j$ : transaction  $j$  that was committed at the time of the initial crash. Assume there are  $N$  of these.

$W(X)$ : the number of update log records for transaction  $X$

**4. Functional Dependencies and Normalization [14 points]**

Given a relation with attributes ABCDEF and the following functional dependencies:

- AB  $\rightarrow$  F
- BE  $\rightarrow$  A
- DF  $\rightarrow$  C
- F  $\rightarrow$  E
- D  $\rightarrow$  A

A. [2 points] What is the attribute closure of AB?

B. [2 points] Of the following FDs, circle the ones that are implied by the functional dependencies given above:

- i. AB  $\rightarrow$  E
- ii. BD  $\rightarrow$  F
- iii. DF  $\rightarrow$  BA
- iv. DF  $\rightarrow$  CE

C. [2 points] Name a candidate key of ABCDEF.

D. [2 points] Write down a functional dependency that causes this relation to violate 3NF.

E. [2 points] Which of the following could occur with the table? (Circle ALL THAT APPLY.)

- i. Update anomaly
- ii. Insertion anomaly
- ii. Deletion anomaly

F. [4 points] For each of the following decompositions of the relation, circle whether it is lossless or not.

- (a) BCDEF, AD                      lossless                      not lossless
- (b) ABCDEF, CDF                      lossless                      not lossless
- (c) ACDE, BEF                      lossless                      not lossless
- (d) ABEF, ACDF                      lossless                      not lossless

**5. Concurrency Control [16 points, 4 per question]**

Consider these three transactions:

T1: INSERT INTO A (SELECT \* FROM B)

T2: INSERT INTO B (SELECT \* FROM A)

T3: PRINT [(SELECT COUNT(\*) FROM A) + (SELECT COUNT(\*) FROM B)]

Suppose that at the outset,  $|A| = 7$  and  $|B| = 3$ , i.e. A has 7 tuples and B has 3 tuples. Assume the DBMS does not enforce any uniqueness of tuples (i.e., duplicate tuples are allowed in the same relation).

Give all the possible outputs of T3's print statement, running T1, T2 and T3 concurrently, under the following options. *Note*: there may be multiple possible outputs for each option, corresponding to all the schedules that are possible under that option. You should produce them all!

1. View serializability

\_\_\_\_\_

2. Strict 2-phase locking with **table-level** locks

\_\_\_\_\_

3. Strict 2-phase locking with **row-level** locks

\_\_\_\_\_

4. Timestamp concurrency control, with table-level timestamps, and transaction timestamps assigned such that  $\text{timestamp}(T2) < \text{timestamp}(T3) < \text{timestamp}(T1)$

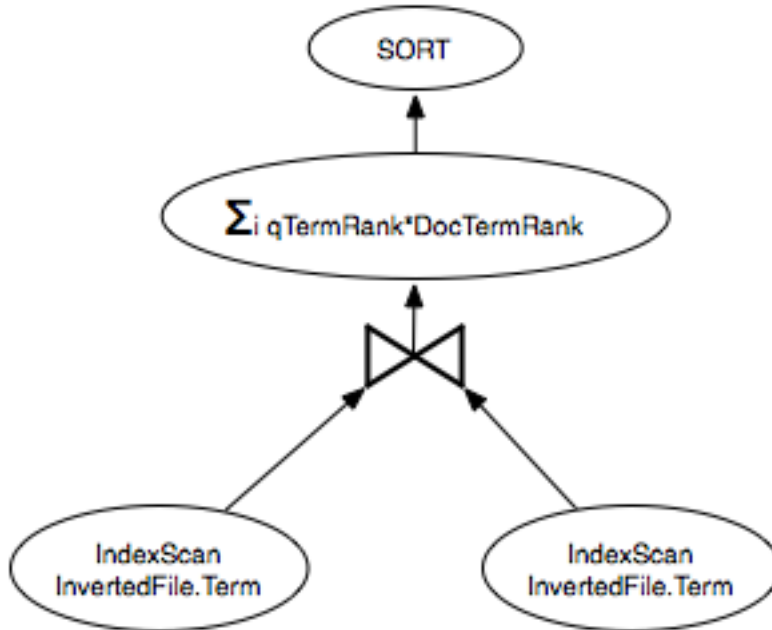
\_\_\_\_\_

**6. Search [6 points]**

Recall the query plan for text search that we discussed in class, pictured below for a two-term query. As you may remember, the schema for the InvertedFile table was:

`InvertedFile(term string, DocID int64, DocTermRank float).`

Recall also that the `qTermRank` in the SUM aggregate is the `TFxIDF` score of the term in the query.



A. [1 point] What is the best choice of a join algorithm for this query?

B. [1 point] According to Google, here are the Document Frequencies on the web for three different terms:

**Britney: 69,500,000**  
**Spears: 61,700,000**  
**Serializabilty: 207,000**

Consider a query that consists of those three terms. What is the `TFxIDF` score of “Serializability” in that query?

- i.. 1/207000
- ii. 1/(69500000\*61700000)
- iii. 3/(207000\*69500000\*61700000)
- iv. Not enough information to compute

Answer: \_\_\_\_\_

Name \_\_\_\_\_

- C. [2 points] Your mother made you a shopping list of 10 items before sending you to the grocery store. Unfortunately, you forgot the list when you went, and had to shop by memory. You are, unfortunately a little absentminded. (You may make a good professor some day!) You returned with 8 items, only 5 of which were on the original list.

Your mom wants you to keep track of your absentmindedness scientifically. Please fill in today's statistics:

Precision: \_\_\_\_\_

Recall: \_\_\_\_\_

- D. [2 points] Consider the letter-oriented bigram approach to matching terms with misspellings. Recall that a bigram is a  $q$ -gram with  $q=2$ .)

i. Write down the bigrams from the word "BERKELEY".

ii. Compute the TF cosine similarity of bigrams in "STANFORD" and "KURDISTAN".