Do not write in this space

**Name**_____

**Class Account** _____

UNIVERISTY OF CALIFORNIA, BERKELEY
College of Engineering
Department of EECS, Computer Science Division

CS186 J. Hellerstein
Spring 2001 Final Exam

**Final Exam: Introduction to Database Systems**

This exam has seven problems, each worth a different amount of points. 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 question, be sure to read it carefully and to *answer all parts of every question!*

You **must** write your answers on the exam. You also must **write your name** at the top of **every page**, and you must turn in all the pages of the exam. **Do not** remove pages from the stapled exam! Extra answer space is provided in case you run out of space while answering. If you run out of space, be sure to make a "forward reference" to the page number where your answer continues.

Good luck!

## Transaction Management and Locking

1. [15 Points] You have been hired as a consultant to the US Government, to decide which database system to buy for the National Institute of Health. You are considering products from three database companies: *Andacle*, *Notacle*, and *JCN*. Each company has sent you their top technical architect to tell you about their system. All three use fancy terminology, but that does not mean they know what they are talking about! Your job is to determine which companies clearly do not know what they are talking about.

   Some of the following statements that you hear are wrong, some are right. For each one, **circle whether it is wrong or right**. **If it is wrong, give a counterexample. If it is right, explain briefly**.

(over →)

**Name**_____

a. "Because we use the Two-Phase Locking protocol (2PL) in *JCN*, deadlocks can never occur in our system."

   Wrong  Right

b. "*Notacle* uses Strict Two Phase Locking Protocol, and therefore we can ensure there will be no cascading aborts."

   Wrong  Right

c. "Our lock manager in *Andacle* uses shared and exclusive locks, which guarantees that all schedules in our system are serializable."

   Wrong  Right

**Name**_____

2. [15 points] Commercial SQL systems actually let you "relax" serializability, by controlling the "locking mode" that is used by a transaction. By default, transactions run with Strict Two-Phase Locking (2PL), but you can explicitly set up a transaction to run in a relaxed mode.

   In this question you will be given a brief description of different relaxed locking modes, and asked about conflicts. Recall that conflicts occur between two transactions that are running at the same time and touch the same data objects. **There may be zero, one, two or three correct answers in each part!**

   a. Transactions running in "Read Committed" mode obtain all locks in the same way as Strict 2PL, and release exclusive locks at the end-of-transaction just like Strict 2PL. However, these transactions release *shared* locks immediately after finishing a read. Said differently, Read Uncommitted transactions use **Short-Term S-Locks, and Strict 2-Phase X-Locks**. Which of the following conflicts **can occur** between a Read Committed transaction, and a transaction following the Strict 2PL locking protocol:

      - Write-Write
      - Read-Write
      - Write-Read

   b. Transactions running in "Read Uncommitted" mode always read data without setting any locks in the lock table. These transactions always set exclusive locks before writing as usual, and hold them until end-of-transaction. Said differently, Read Uncommitted transactions use **No S-Locks, and Strict 2-Phase X-Locks**. Which of the following conflicts **can occur** between a Read Uncommitted transaction, and a transaction following the Strict 2PL locking protocol:

      - Write-Write
      - Read-Write
      - Write-Read

   c. All SQL lock modes – even the relaxed modes – use Strict 2-Phase X-Locks. Consider what would happen if a transaction released a shared lock immediately after a read, and released an exclusive lock immediately after a write (**Short-Term S-Locks and Short-Term X-Locks**). Which of the following **could happen** in those cases?

      - Write-Write conflicts.
      - Inconsistent data in the database.
      - Deadlocks

**Name**_____

**Functional Dependencies and Decomposition**

3. [10 points] Consider the relation **R**(A, B, C, D), with a **legal** instance of R as shown below. You have not been told what functional dependencies hold on R, you have only been shown this legal instance. Each statement below can be categorized either as True, False, or Not Enough Information to be sure. **Circle one** of those 3 categories for each statement, and **justify your answers briefly.**

| **A** | **B** | **C** | **D** |
|---|---|---|---|
| 2 | 2 | 2 | 4 |
| 2 | 2 | 2 | 2 |
| 4 | 2 | 2 | 2 |
| 2 | 4 | 2 | 3 |
| 4 | 4 | 4 | 8 |

   a. Relation R needs to be decomposed to achieve Third Normal Form.

      True False NotEnoughInfo

   b. The functional dependency A → C holds for R.

      True False NotEnoughInfo

   c. CD is a not a key for R.

      True False NotEnoughInfo

**Name**_____

4. [15 points] Consider the relation **Student**(SSN, Id, Email, NumericGPA, LetterGPA), also known as SIENL. The relation has the following Functional Dependencies (FDs): {S → I, I → E, E → S, N → L}.

   a. Which of the following is the most efficient, correct way to find the keys?
      - Apply Armstrong's Axioms repeatedly until you cannot derive any more FDs. For each FD with right-hand-side equal to SIENL, the corresponding left-hand-side is a key.

      - Find the attribute closures for each of S,I,E,N and L. The keys are those sets of attributes for which the union of their closures is equal to SIELN.

      - Construct a dependency graph and test it for cycles. All nodes on a cycle correspond to a key.

   b. How many (minimal) keys are there for Student? Explain either by listing the keys, or by a brief explanation.

   c. Decompose R into BCNF, showing a tree of decompositions. For each step, note the violation you resolve.

**Name**_____

**Logging and Recovery**

5. [10 points] A sales representative from *Debacle* corporation claims that they provide a STEAL/NO-FORCE buffer management policy for efficiency, and do not need any logging in their system to guarantee the ACID properties. As a database expert, you have serious doubts about this.

   a. What potential problem can happen due to the STEAL policy in *Debacle*? Briefly explain a scenario in which this happens.

   b. What potential problem can happen due to the NO-FORCE policy in *Debacle*? Briefly explain a scenario in which this happens.

**Name**_____

6. [20 points] Below is the state of a DBMS after a crash. Your job will be to perform recovery according to ARIES (Algorithm for Recovery and Isolation Exploiting Semantics). Follow the procedures on the next page.

| Database: (PageNo: pageLSN) | | |
|---|---|---|
| A: 2 | B: 3 | C: 30 |
| D: 5 | E: 40 | F: 7 |

| Transaction Table | | |
|---|---|---|
| Xact ID | status | lastLSN |
| 1 | running | 20 |
| 2 | running | 30 |
| | | |

| Dirty Page Table | |
|---|---|
| page | recLSN |
| A | 10 |
| B | 20 |
| C | 30 |
| | |
| | |
| | |

| Master Log Record |
|---|
| LSN of last checkpoint = 35 |

Log:

| LSN | xid | page | prevLSN | Type | UndoNextLSN | Indicate if redone: |
|---|---|---|---|---|---|---|
| 10 | 1 | A | (null) | Update | (null) | |
| 20 | 1 | B | 10 | Update | (null) | |
| 30 | 2 | C | (null) | Update | (null) | |
| 35 | (null) | (null) | (null) | Begin checkpoint | (null) | [redacted] |
| 40 | 3 | E | (null) | Update | (null) | |
| 50 | 2 | (null) | 30 | Commit | (null) | [redacted] |
| 60 | 3 | D | 40 | Update | (null) | |
| 70 | 2 | (null) | 50 | End | (null) | [redacted] |
| 80 | 1 | C | 20 | Update | (null) | |
| 90 | 1 | (null) | 80 | Commit | (null) | [redacted] |
| 100 | 3 | E | 60 | Update | (null) | |
| 105 | See tables above | | | End checkpoint | (null) | [redacted] |
| 110 | | | | | | [redacted] |
| 120 | | | | | | [redacted] |
| 130 | | | | | | [redacted] |
| 140 | | | | | | [redacted] |
| 150 | | | | | | [redacted] |
| 160 | | | | | | [redacted] |
| 170 | | | | | | [redacted] |
| 180 | | | | | | [redacted] |
| 190 | | | | | | [redacted] |
| 200 | | | | | | [redacted] |

(over →)

**Name**_____

a. **Analysis**. Do analysis to correctly fill out the Transaction and Dirty Page Tables. You may find that you're scratching things out as you go…please try to be neat so we can read the final state of the tables.

b. **Redo**. Perform redo to repeat history. Put a mark in the rightmost column of the log next to each update log record that you must redo in the database during this phase.

c. **Undo**. Perform undo. For your answer, please add the appropriate log entires while undo is being performed, but **do not** modify the tables you filled in for Analysis (so that we can see what you did in part (a)).

d. At the end of Undo, what would the real transaction table have in it?

**Name**_____

7. [15 points] You are building an e-commerce application. Suppose you have four tables:

   - **ShoppingCarts**(CartID, CustomerID, ExpirationTime): 10,000 rows, 500 pages. CustomerID is a foreign key to Customers.
   - **CartContents**(CartID, ProductID): 50,000 rows, 2,000 pages. CartID is a foreign key to ShoppingCarts, ProductID is a foreign key to Products.
   - **Products**(ProductID, name, description, manufacturer, price): 100,000 rows, 10,000 pages
   - **Customers**(CustomerID, name, address, phone): 200,000 rows, 10,000 pages

   There are a number of common queries in your application. One produces a "checkout page" for a customer (whose ID is represented by the variable CUSTVAR):

```
SELECT *
  FROM ShoppingCarts INNER JOIN CartContents
       INNER JOIN Products INNER JOIN Customers C
 WHERE C.CustomerID = CUSTVAR
```

   Another common query is to create a shopping cart (the Values are filled in by the application depending on what the customer ID is, what the time of day is, etc.)

```
INSERT INTO ShoppingCarts Values ( … )
```

   A third common query is to add a product to a cart (again, the Values are filled in by the app):

```
INSERT INTO CartContents Values ( … )
```

   The final common query deletes a shopping cart (whose ID is represented by the variable CARTVAR):

```
DELETE FROM ShoppingCarts WHERE CartID = CARTVAR
```

   The computer you're running on is also running a number of other processes, so you can only support a buffer pool that is 200 pages large. Hence you cannot expect any of your tables to fit into the buffer pool.

(over → )

**Name**_____

a) Provide CREATE TABLE statements for the tables ShoppingCarts and CartContents. You may assign types as you see fit. Be sure that integrity constraints are maintained. Also, ensure that all the common transactions are able to run to completion (i.e. they don't have to be aborted)!

(over →)

b) Assume that each transaction works as follows: an existing customer creates a cart, adds five existing products to the cart, generates a checkout page, and deletes the cart. Given this workload, for each of the following indexes say whether they should be built (YES/NO), and **justify why or why not**. Be sure to justify all the details of the indexes you think should be built!

- A clustered B+-tree index on CartContents.ProductID

- A linear hash index on Customers.CustomerID

- A linear hash index on ShoppingCarts.CartID

- A clustered B+-tree index on ShoppingCarts.CustomerID

- A clustered B+-tree index on ShoppingCarts.CustomerID

- A linear hash index on Products.ProductID

(over →)

c) You are trying to decide between MySQL and Postgres, two different open source DBMSs. One advantage of Postgres over MySQL is that Postgres supports hash joins, sort-merge joins as well as block and index nested-loops joins; MySQL only supports block and index nested-loops joins. Assuming both systems were the same in every other way, would Postgres be faster than MySQL for your workload? Justify your answer.

(over →)

d) In fact, Postgres has other advantages over MySQL as well. One additional advantage is that Postgres supports row-level locking (i.e. its transaction manager sets locks on individual rows), while MySQL only supports table-level locking (it sets locks on entire tables). Hence in principle Postgres can provide higher concurrency than MySQL in many scenarios.

Your friend loves MySQL, and tries to convince you that its locking technique is not a problem for your workload. One of his main arguments is that your "checkout page" query is a "killer query" that will effectively force both MySQL and Postgres to run one user at a time to guarantee the ACID properties. Explain why your friend is wrong.

**Name**_____

**Name**_____

**Name**_____