B561 Final Exam
Fall 2001

This exam comprises 3 pages. Ensure you hand in answers to **9** questions. Point counts are given for each question: ensure you take these into account when planning your time. **Total Points: 135**.

Part I: Query Formulation (50 points)

Consider the following schemas:

ZONES(ZONEID NUMBER, TYPE CHAR(1))
ROADS(ROADID NUMBER, SRCZONE NUMBER,
ENDZONE NUMBER, DIST NUMBER)

ROADID is the primary key for relation ROADS and ZONEID is the primary key for relation ZONES. As in assignments 4 and 5, ROADS indicates directed routes between zones in a particular city. Note that the existence of a road from a zone z_1 to a zone z_2 says nothing about the existence (or lack thereof) of a road from z_2 to z_1.

Given a zone, z, the *immediate neighbors* of z are all zones, z', reachable from z via a road in ROADS, i.e. $\{z' | \exists r, d. \langle r, z, z', d\rangle \in \text{ROADS}\}$.

You may wish to abbreviate ROADS by R and ZONES by Z in order to save writing.

1. Formulate the following queries in the *Relational Algebra*, using only the operators Π, σ, $\times$, $\cup$, $-$ and $\bowtie$ (natural join).

   (a) **(5 points)** Find those zones having no roads leaving them.

   (b) **(10 points)** Find pairs of zones $\langle z, z'\rangle$ who have the same set of immediate neighbors.

   (c) **(5 points)** Find zone types, t, such that all zones of that particular type have the same set of intermediate neighbors.

2. Formulate the following queries in the domain relational calculus (DRC). Make sure you conform to a consistent syntax.

   (a) **(5 points)** Find those roads that link zones having the same type.

   (b) **(10 points)** Find those zones, z, that have no zone of the same type among their immediate neighbors, i.e. for all z' that is an immediate neighbor of z, the type of z' is not the same as the type of z.

3. Formulate the following queries in SQL. Conform to *either* the SQL '92 *or* the Oracle syntax (but not a mixture of both). You may use aggregate functions if you wish.

   (a) **(5 points)** Find those zones that have exactly three zones of type 'I' among their immediate neighbors.

(b) **(10 points)** Find zone pairs $\langle z, z' \rangle$ such that the shortest road between z and z' has a distance of greater than 50 units.

Part II: PL/SQL (25 points)

A two-element *list* is an ADT (Abstract Data Type) having an element distinguished as the *head* and another distinguished as the *tail*. An n-element list is an ADT having an element distinguished as the *head* and having an $n-1$-element list as its *tail*. Note the inherrently recursive structure of the list ADT. The following operations are useful for ADT *list*:

length(): returns the length of a list (i.e. the number of elements in it).

reverse(): reverses the order of elements in the list.

In your answers to the following questions, try to conform to the SQL '92 or the PL/SQL syntax as much as possible. Minor syntax errors will not cause you to lose points, but do *not* simply state your answers in pseudo-code.

3. **(5 points)** Suggest an appropriate relational data type (that can be manipulated easily in PL/SQL) for ADT *list*. Assume that data elements will have the type NUMBER. Give an appropriate SQL '92 `CREATE` statement for your list data type.

4. **(5 points)** Assume that a list A_LIST exists in the database, created in accordance with question 3 (above). Write a PL/SQL procedure *length()*, that returns the length of A_LIST.

5. **(15 points)** Assume that a list A_LIST exists in the database, created in accordance with question 3 (above). Write a PL/SQL procedure *reverse()*, that reverses A_LIST. Assume an appropriate list A_REV_LIST exists to contain the result.

Part III: Query Optimization (35 points)

6. Consider the following query.

```
SELECT Z.ZONEID
FROM   ROADS R1, ROADS R2,
       ZONES Z, ZONES Z1, ZONES Z2
WHERE  R1.ROADID != R2.ROADID AND R1.SRCZONE = Z.ZONEID AND
       R2.SRCZONE = Z.ZONEID AND R1.ENDZONE = Z1.ZONEID AND
       R2.ENDZONE = Z2.ZONEID AND Z1.TYPE = 'R' AND Z2.TYPE = 'R'
```

(a) **(5 points)** Translate the query into an RA expression using the naieve method developed in class. Give the query plan form (tree form) of this naieve RA query.

(b) **(10 points)** Optimize the query plan obtained in (a) (above) using the rule-based optimization method developed in class. Ensure the join ordering you choose is a *left-deep* ordering. If you feel you need any additional assumptions, state these briefly and clearly.

7. Given relation schemas $R(A, B, C, D)$ and $S(A, B, C, D)$, prove or disprove the following statements.

   (a) **(10 points)**

   $$\big(\Pi_{A,B}(R) \cap \Pi_{A,B}(S)\big) \cup \Pi_{A,B}(R \bowtie S) = \big(\Pi_{A,B}(R) \cup \Pi_{A,B}(S)\big) \cap \Pi_{A,B}(R \bowtie S).$$

   (b) **(10 points)**

   $$\sigma_{A=B}(R - S) = \sigma_{A=B}(R) - \sigma_{A=B}(S).$$

Part IV. Transaction Management, Concurrency Control, and Recovery (25 points)

Questions 8 and 9 concern the schedule, S, given in figure 1. After step 10, all schedules either abort or commit (and these are the only further actions that occur).

	T_1	T_2	T_3	T_4
1			R(X)	
2				R(Y)
3		R(Z)		
4	R(X)			
5			W(X)	
6				W(X)
7			R(X)	
8	W(X)			
9		W(Z)		
10		R(X)		
⋮				

Figure 1: Schedule S for questions 8 and 9.

8. (a) **(5 points)** Draw the (conflict) serializability graph for S. Is S conflict serializable? Justify your answer.

   (b) **(5 points)** Suppose each of the transactions T_1 through T_4 commit. In which order would the transactions have to commit in order that S is recoverable?

   (c) **(5 points)** Is there an order in which the transactions could commit that would make S strict? Explain your answer.

9. **(10 points)** Suppose transaction T_3 commits at step 11, and there is a crash directly afterward. Describe in detail the actions an UNDO/REDO recovery manager would take to restore consistency to the database. You may wish to display the contents of the log at the point of the crash to assist with your explanation. In this case, you should use appropriate constants (e.g. $a, b, c, ...$) to indicate the before and after images of the variables X, Y, Z according to the log.