Umeå University
Department of Computing Science
TDBC86 – Database Concepts
Examination: January 15, 1999

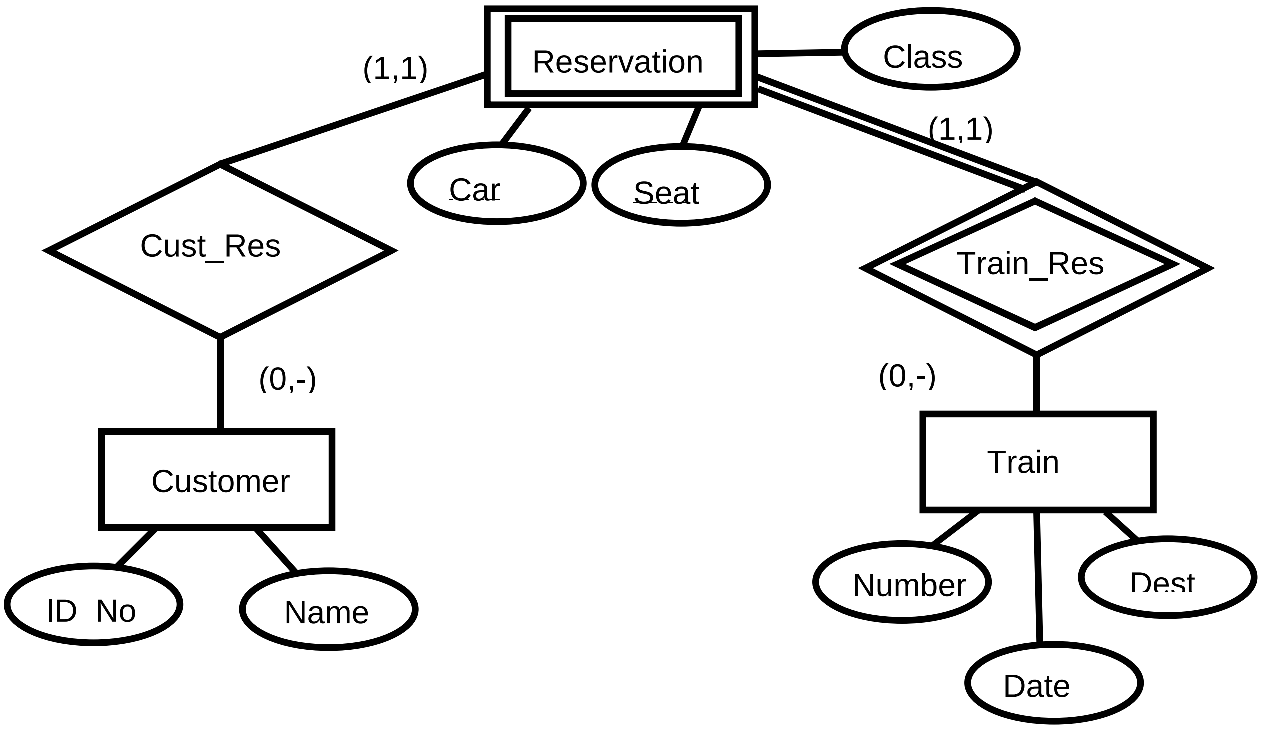
Name and ID number: ________________________________

Signature:____

1. All answers must be written in English.
2. A pocket engineering calculator and English/X – X/English dictionary may be used. No other help materials are allowed.
3. Solutions to all problems must be written on the special examination paper, which is provided. Write your name, ID number, and the problem number on each solution sheet. Collate the sheets in numerical order of the problems. Please write on only one side of the paper.
4. The examination has a total of 900 points. Your point total will be divided by 10, and then your average points from the obligatory exercises will be added on, to obtain a final score between 0 and 100, upon which your grade will be based.
5. In the table below, place an X in the position for any problem for which you have attempted a solution, and which you wish to have graded. **It is extremely important that you fill in this table properly**, because of the following option. For any box which is left blank, the associated question will not be graded, and you will instead be awarded 20% of the points for that question. Your decision to leave a box blank is definitive, so be very careful. For example, If you leave box 3(b) blank, your answer to that question will not be graded, even if it is completely correct. Similarly, if you place an X in box 3(b), but provide no answer whatsoever to that question, you will not receive 20% of the points for that question. It is strongly recommended that you use a pencil, in case you change your mind!

Prob.	1	2	3	4	5	6	7	8	9	10	11	12
(a)												
(b)	■											■
(c)	■									■	■	■
(d)	■	■	■	■						■	■	■
(e)	■	■	■	■	■			■	■	■	■	■
(f)	■	■	■	■	■		■	■	■	■	■	■
(g)	■	■	■	■	■		■	■	■	■	■	■
(h)	■	■	■	■	■		■	■	■	■	■	■

1. (70 points) Shown below is an ER-diagram for seat reservations on trains. Using the techniques developed in the course, map this diagram into an equivalent relational schema. Show all keys, primary and foreign.



Notes:

- Dest = destination.
- The (x,y) notation gives the minimum and maximum number of participants in a relationship. Thus, (1,1) means exactly one, and (0,-) means any number.

The following simple “banking” relational database schema,

Customer

Cust_ID	Name	Residence_City

Account

Acct_No	Branch_ID	Cust_ID	Type	Balance

Branch

Branch_City	Branch_ID	Branch_type

together with the following three queries, are the common context in which Problems 2, 3, and 4, below, should be solved.

(a) Find the name and ID of those customers who have an account in a branch located in Vännäs, and an account in a branch located in Umeå.

(b) Find the name and ID of those customers who have an account in every branch which is located in their city of residence.

(c) Find the name and ID of those customers who have an account in exactly one branch which is located in their city of residence. (These customers may have accounts in any number of branches, but exactly one of the branches must be in the customer’s city of residence.)

2. (75 points total; 25 for each part) Solve each of the three queries (a) – (c) in the relational algebra. Functional operators, such as count and average, may not be used.

3. (75 points total; 25 for each part) Solve each of the three queries (a) – (c) in the tuple relational calculus. Functional operators, such as count and average, may not be used.

4. (75 points total; 25 for each part) Solve each of the three queries (a) – (c) using SQL. For this part only, you may use SQL functional operations, such as count and average.

5. (80 points total) Using the database schema of Problem 2, provide solutions, in SQL, to the following additional queries. All SQL operations may be used, including aggregation operations.

(a: 20 points) For each city, find the average of the balances of all accounts which are housed in a branch in that city. The query must list both the city and the average balance.

(b: 20 points) Find the name and ID of each customer who has an account with a balance > 100000.

(c: 20 points) For each branch, find the average balance of all accounts owned by customers who live in the city housing that branch.

(d: 20 points) For each branch type, find the maximum balance over all accounts housed in branches of that type.

6. (80 points total; 10 points for each part) Let **A** be a finite set of attributes, and let **R** be a relational database schema with a single relation R[**A**]. Assume that R is constrained by a set $\mathfrak{F}$ of functional dependencies. Within this context, define the following concepts. Your definitions should be formal enough to convey precise meaning.

(a) Superkey

(b) Candidate key

(c) Primary key

(d) Third normal form (3NF)

(e) Boyce-Codd normal form (BCNF)

Now assume further that $\mathbf{A}_1, \mathbf{A}_2, .., \mathbf{A}_k \subseteq \mathbf{A}$, and that $\{R_1[\mathbf{A}_1], R_2[\mathbf{A}_2], .., R_k[\mathbf{A}_k]\}$ is a decomposition of R[**A**] into projections. Define the following properties for the decomposition.

(f) Dependency preserving decomposition

(g) Lossless decomposition

Finally, let **A** and **B** be finite sets of attributes, and let R[**A**] and S[**B**] be relations in a database schema.

(h) Define what it means for a set of attributes of R[**A**] to be a foreign key of S[**B**].

7. (75 points total) Consider the relational schema **R** with relation R(A,B,C,D,E,F) and functional dependencies $\{A \rightarrow B, C \rightarrow EF, B \rightarrow C, CD \rightarrow AB\}$.

(a) (15 points) Find all candidate keys for **R**.

(b) (15 points) Give a minimal cover for the set of functional dependencies.

(c) (15 points) Give a lossless and dependency-preserving decomposition of **R** which is in 2NF but not in 3NF. Explain why your decomposition is not in 3NF.

(d) (15 points) Give a lossless and dependency-preserving decomposition of **R** which is in 3NF.

(e) (15 points) Determine whether or not your decomposition of (d) is in BCNF. Justify your answer with an explanation.

The following data apply to problems 8 and 9. A B$^+$-tree is used as the data structure for the storage of a relation. Assume that each node, interior or leaf, is stored in one page, and that each tuple is stored as one record. The following parameters are given:

Number of tuples in the relation	800000000 (8×10^8)
Size of one tuple	512 bytes
B$^+$-tree pointer size	4 bytes
B$^+$-tree internal key size	24 bytes
Page size	4K (4096 bytes)
Sequential-access pointers in the leaf nodes	8 bytes total

8. (80 points total) Answer the following.

(a) (20 points) Compute the maximum height that such a B-tree may have. (The height is defined as the length of a path from the root to a leaf.)

(b) (20 points) For such a tree of maximum height, assuming a uniform distribution of records in the nodes (other than possibly the root), compute the average number of records per (non-root) interior node. Assume that the root has the minimum number of records possible.

(c) (20 points) Compute the number of leaf nodes in the B$^+$-tree. Assume that a leaf node contains only records and the sequential-access pointers.

(d) (20 points) Compute the number of interior (index) nodes in the B$^+$-tree. Assume that such a node contains only pointers.

9. (80 points total) Answer the following.

(a) (20 points) Compute the minimum height that such a B-tree may have. (The height is defined as the length of a path from the root to a leaf.)

(b) (20 points) For such a tree of minimum height, assuming a uniform distribution of records in the nodes (other than possibly the root), compute the average number of records per (non-root) interior node. Assume that the root has the maximum number of records possible.

(c) (20 points) Compute the number of leaf nodes in the B$^+$-tree. Assume that a leaf node contains only records and the sequential-access pointers.

(d) (20 points) Compute the number of interior (index) nodes in the B$^+$-tree. Assume that such a node contains only pointers.

10. (a: 40 points) Give a precise definition for each of the following concepts.

(i) Transaction
(ii) Serial schedule
(iii) View serializable schedule
(iv) Conflict serializable schedule

(b: 30 points) A simple database system has three data objects, named x, y, and z. The following three transactions are given (r = read; w = write).

$T_1 =$ $r_1(x)$ $w_1(x)$
$T_2 =$ $r_2(y)$ $w_2(x)$ $w_2(y)$
$T_3 =$ $r_3(z)$ $w_3(x)$ $w_3(z)$

For each point listed below, give an example schedule (using the three transactions listed above) which satisfies the constraints indicated. If such an example is impossible, so state and explain why.

(i) A serial schedule.
(ii) A view serializable schedule which is not conflict serializable.
(iii) A conflict serializable schedule which is not view serializable.

11. (a: 30 points) Give precise definitions of the following notions:

(i) Two phase locking protocol (2PL).
(ii) Strict two-phase locking protocol.
(iii) Conservative two-phase locking protocol.

(b: 40 points) Given are the following three transactions on a database system with four data objects: x, y, z, and w.

T_1 = Read(T_1,x) Write(T_1,x) Read(T_1,y) Write(T_1,y)
T_2 = Read(T_2,x) Write(T_2,x) Read(T_2,w) Write(T_2,w)
T_3 = Read(T_3,z) Write(T_3,z) Read(T_3,x) Write(T_3,x)

For each of the following schedule types, give a *serial* schedule, with appropriate lock and unlock statements inserted, for the above transactions which has precisely the properties stated. If no such schedule is possible, so state and explain why. (The order of the reads and writes must be the same as for some serial schedule. The lock and unlock statements may be distributed, as needed.)

(i) A schedule which is not 2PL.
(ii) A schedule which is 2PL, but neither strict nor conservative.
(iii) A schedule which is strict 2PL but not conservative 2PL.
(iv) A schedule which is conservative 2PL but not strict 2PL.
(v) A schedule which is both conservative 2PL and strict 2PL.

12. (70 points) Shown below is the initial segment for seven transactions, labelled T_1 through T_7. This sequence of operations is run on a database system with immediate update; that is, the database is updated immediately upon execution of a write statement. Further, a log is maintained which records the sequence of previous values for each updated database record. Assume that transaction T_1 crashes after the indicated Write statement. Identify the following, assuming that database integrity is to be maintained.

(i) Those transactions which must be rolled back and re-run.

(ii) For each data object in {x, y, z} which is restored by the rollback process, the Write statement whose *before* value the data object is restored to.

(iii) Those completed transactions which need not be rolled back or re-done. (A completed transaction is one for which an End statement has been executed.)

(iv) Those transactions which are not completed, but which may be continued without rollback. (Note: T_1 must be restarted, since it has crashed!)

Begin(T_1)
Read(T_1,x)
Begin(T_2)
Read(T_2,y)
Write(T_2,y)
End(T_2)
Write(T_1,x)
Begin(T_4)
Read(T_4,y)
Read(T_4,x)
Write(T_4,y)
End(T_4)
Begin(T_3)
Read(T_3,z)
Read(T_3,y)
Write(T_3,y)
End(T_3)
Begin(T_5)
Read(T_5,z)
Write(T_5,z)
End(T_5)
Begin(T_6)
Read(T_6,z)
Write(T_6,z)
Begin(T_7)
Read(T_7,x)
Write(T_7,x)
Read(T_1,y)
Write(T_1,y) $\leftarrow$ T_1 crashes *after* this statement has executed.
⋮

Appendix:

A useful formula for uniform (m,r,d) B-trees:

$$R(m,r,d) = (m+1) \cdot (r+1)^{d} - 1$$

m = the total number of indices at the root node.
r= the total number of records in each non-root node.
d = the depth of the tree; *i.e.,* the length of a path from the root to a(ny) leaf node.
R(m,r,d) = the total number of records in the tree.

A useful formula for uniform (m,q,r,d) B^{+}-trees:

$$R(m,q,r,d) = (m+1) \cdot (q+1)^{d-1} \cdot r$$

m = the total number of indices at the root node.
q = the total number of indices in each non-root, non-leaf node.
r= the total number of records in each leaf node.
d = the depth of the tree; *i.e.,* the length of a path from the root to a(ny) leaf node.
R(m,q,r,d) = the total number of records in the tree.